



开发人员指南

Amazon Simple Storage Service



API 版本 2006-03-01

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

Amazon Simple Storage Service: 开发人员指南

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆或者贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

什么是 Amazon S3 ?	1
API 调用建议	2
直接调用 REST API	2
权限	3
使用 Amazon S3 进行开发	4
提出请求	4
关于访问密钥	4
请求终端节点	6
通过 IPv6 发出请求	6
使用 AWS 开发工具包提出请求	16
使用 REST API 提出请求	52
使用 AWS CLI	62
了解有关 AWS CLI 的更多信息	63
使用 AWS SDK 进行开发	63
SDK 编程接口	65
在请求身份验证中指定签名版本	65
获取支持的 Amazon S3 请求 ID	73
使用 AWS CLI 获得请求 ID	73
使用 Windows PowerShell 获取请求 ID	74
使用 AWS CloudTrail 数据事件获取请求 ID	74
使用 S3 服务器访问日志记录获取请求 ID	74
使用 HTTP 获得请求 ID	74
使用 Web 浏览器获得请求 ID	75
使用 AWS SDK 获得请求 ID	75
S3 表类数据存储服务支持的 S3 对象级 API 操作	79
代码示例	85
Amazon S3	91
基本功能	96
场景	782
无服务器示例	1232
Amazon S3 控件	1244
基本功能	1245
S3 目录存储桶	1325
基本功能	1326

场景	1450
对请求进行身份验证 (AWS 签名版本 4)	1453
身份验证方法	1454
对请求进行签名简介	1455
使用 Authorization 标头	1456
概述	1456
签名计算：在单个分块中传输有效载荷	1459
签名计算：在多个分块中传输有效载荷	1477
签名计算：包括尾随标头	1489
使用查询参数	1495
计算签名	1498
一个示例	1501
示例 2	1502
示例：签名计算	1503
使用 Java 的签名计算示例	1504
使用 C# 的签名计算示例	1505
对 HTTP POST 请求进行身份验证	1506
计算签名	1508
特定于 Amazon S3 签名版本 4 身份验证的策略键	1509
使用与签名版本 4 有关的条件键的存储桶策略示例	1511
使用 POST 的基于浏览器的上传	1514
POST Object	1515
说明	1515
版本控制	1516
请求	1516
示例	1531
相关资源	1532
POST Object restore	1533
说明	1533
使用 Select 请求查询归档	1533
还原归档	1534
请求	1535
响应	1550
示例	1550
更多信息	1553
使用 HTTP POST 的基于浏览器的上传	1553

计算签名	1555
创建 HTML 表单	1556
HTML 表单声明	1557
HTML 表单字段	1557
POST 策略	1563
过期	1564
条件匹配	1564
条件	1565
字符转义	1569
POST 上传示例	1571
使用 HTTP POST 将文件上传到 Amazon S3	1571
使用 AWS Amplify 进行基于浏览器的上传	1574
使用 AWS Amplify JavaScript 库将文件上传到 Amazon S3	1574
更多信息	1575
通用请求标头	1576
通用响应标头	1579
错误响应	1583
REST 错误响应	1584
错误代码列表	1585
SELECT 对象内容错误代码列表	1609
与复制相关的错误代码列表	1619
与标记相关的错误代码列表	1621
与 Amazon S3 注释相关的错误代码列表	1622
Amazon S3 on Outposts 错误代码列表	1625
Amazon S3 Storage Lens 存储统计管理工具错误代码列表	1626
Amazon S3 对象 Lambda 错误代码列表	1633
Amazon S3 异步错误代码列表	1636
Amazon S3 访问权限管控错误代码列表	1637
Amazon FSx 相关的错误代码列表	1640
Amazon S3 表类数据存储服务错误代码列表	1641
Amazon S3 排错最佳实践	1641
重试 InternalErrors	1642
针对重复的 SlowDown 错误调整应用程序	1642
隔离错误	1642
AWS 术语表	1644
资源	1645

附录	1647
附录：SelectObjectContent 响应	1648
说明	1648
响应	1648
相关资源	1657
附录：OPTIONS 对象	1658
说明	1658
请求	1658
响应	1659
示例	1660
相关资源	1661
附录：SOAP API	1661
服务上的操作（SOAP API）	1662
对存储桶的操作（SOAP API）	1664
针对对象的操作（SOAP API）	1676
对 SOAP 请求进行身份验证	1699
使用 SOAP 设置访问策略	1701
常见元素	1702
SOAP 错误响应	1702
附录：对请求进行身份验证（AWS 版本签名 2）	1704
使用 REST API 对请求进行身份验证（AWS 版本签名 2）	1706
对 REST 请求进行签名和身份验证（AWS 签名版本 2）	1709
使用 POST (AWS Signature Version 2) 的基于浏览器的上传	1722
附录：生命周期配置 API（已弃用）	1745
PUT 存储桶生命周期（已弃用）	1746
GET 存储桶生命周期（已弃用）	1762
文档历史记录	1773

什么是 Amazon S3 ?

Amazon Simple Storage Service (Amazon S3) 是一种对象存储服务，提供行业领先的可扩展性、数据可用性、安全性和性能。各种规模和行业的客户都可以使用 Amazon S3 存储和保护任意数量的数据，用于数据湖、网站、移动应用程序、备份和恢复、归档、企业应用程序、IoT 设备和大数据分析。Amazon S3 提供了管理功能，使您可以优化、组织和配置对数据的访问，以满足您的特定业务、组织和合规性要求。

本指南提供以开发人员为中心的有关使用 Amazon S3 REST API 来构建在云中存储和检索数据的应用程序的信息。

您可以借助任何支持 HTTP 的工具包来使用 REST API。只要对象是匿名可读的，您甚至可以使用浏览器来提取它们。

REST API 使用标准的 HTTP 标头和状态代码，以使标准的浏览器和工具包按预期工作。在某些区域中，我们向 HTTP 添加了功能 (例如，我们添加了标头来支持访问控制)。在这些情况下，我们已尽最大努力使添加的新功能与标准的 HTTP 使用样式相匹配。

Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。但是，不支持将新的 Amazon S3 功能用于 SOAP。我们建议您使用此 REST API 或以下链接中的 AWS SDK：

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

本开发人员指南包括：

- [通用请求标头和通用响应标头](#)
- [错误响应](#)
- [使用 POST 的基于浏览器的上传 \(AWS 签名版本 4 \)](#)

Important

在阅读特定 API 主题之前，请先阅读以下有关身份验证和访问控制的内容。

对 Amazon S3 的请求可以是经过身份验证的，也可以是匿名的。经过身份验证的访问需要 AWS 可用来对请求进行身份验证的凭证。

API 调用建议

直接从代码中进行 REST API 调用可能会比较繁琐。它要求您编写必要的代码来计算有效的签名，以便对请求进行身份验证。我们建议改用以下替代方法：

- 使用 [AWS SDK](#) 发送您的请求。

另请参阅[示例代码和库](#)。

如果您使用 SDK，则无需编写代码来计算用于请求身份验证的签名，因为 SDK 客户端会使用您提供的访问密钥对请求进行身份验证。您应始终使用 AWS SDK，除非您有充分的理由不这样做。

- 使用 AWS CLI 进行 Amazon S3 API 调用。有关设置 AWS CLI 和示例 Amazon S3 命令的信息，请参阅以下主题：

《Amazon Simple Storage Service 用户指南》中的[设置 AWS CLI](#)。

《AWS Command Line Interface 用户指南》中的[在 AWS 命令行界面中使用 Amazon S3](#)。

直接调用 REST API

Note

PUT 请求标头的大小限制为 8KB。在 PUT 请求标头中，系统定义的元数据的大小限制为 2 KB。通过计算每个键和值的 US-ASCII 编码中的字节总数来测量系统定义的元数据的大小。

如果您想进行自己的 REST API 调用，而不是使用上面的替代方法之一，那么需要记住一些事项。

- 要从您的代码直接调用 REST API，请使用有效的凭证创建签名，并在请求中包含签名。有关各种身份验证方法和签名计算的信息，请参阅[对请求进行身份验证 \(AWS 签名版本 4\)](#)。
- REST API 使用标准的 HTTP 标头和状态代码，以便标准的浏览器和工具包按预期工作。在某些区域中，我们向 HTTP 添加了功能 (例如，我们添加了标头来支持访问控制)。在这些情况下，我们已尽最大努力使添加的新功能与标准的 HTTP 使用样式相匹配。有关发出请求的更多信息，请参阅[提出请求](#)。

权限

您可以使用有效的凭证来对您的请求进行身份验证，但除非您拥有账户所有者或存储桶拥有者的 S3 权限，否则您无法创建或访问 Amazon S3 资源。这些权限通常通过 AWS Identity and Access Management (IAM) [策略](#)授予，例如存储桶策略。例如，您必须拥有创建 S3 存储桶或在存储桶中获取对象的权限。有关 S3 权限的完整列表，请参阅 [Amazon S3 的操作、资源和条件键](#)。

有关按 S3 资源类型划分的对于 S3 API 操作的权限的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [Amazon S3 API 操作所需的权限](#)。

如果您使用您的 AWS 账户的根用户凭证，则您拥有所有权限。但是，不建议使用根用户凭证。相反，我们建议您在账户中创建 AWS Identity and Access Management (IAM) 角色并管理用户权限。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [访问管理](#)。

使用 Amazon S3 进行开发

本部分介绍了与开发人员相关的使用 Amazon S3 的主题。有关更多信息，请查看以下主题。

主题

- [提出请求](#)
- [使用 AWS CLI 进行 Amazon S3 开发](#)
- [使用 AWS SDK 通过 Amazon S3 进行开发](#)
- [获取 AWS 支持的 Amazon S3 请求 ID](#)
- [S3 表类数据存储服务支持的 Amazon S3 对象级 API 操作](#)

提出请求

Amazon S3 是一项 REST 服务。可以使用 REST API 或打包底层 Amazon S3 REST API 以简化编程任务的 AWS 开发工具包（请参阅 [示例代码和库](#)）包装程序库，向 Amazon S3 发送请求。

与 Amazon S3 的每一次交互都是经身份验证的或匿名的。身份验证是对尝试访问 (Amazon Web Services) (AWS) 产品的请求者身份进行验证的过程。经身份验证的请求必须包含可验证请求发送者的签名值。签名值的一部分是从请求者 AWS 的访问密钥（访问密钥 ID 和秘密访问密钥）生成的。有关获取访问密钥的更多信息，请参阅《AWS 一般参考》中的[如何获取安全凭证？](#)

如果您使用 AWS 开发工具包，则库将通过您提供的密钥计算签名。然而，如果您在应用程序中直接调用 REST API，您必须编写代码来计算签名并将它添加到请求中。

主题

- [关于访问密钥](#)
- [请求终端节点](#)
- [通过 IPv6 向 Amazon S3 发出请求](#)
- [使用 AWS 开发工具包提出请求](#)
- [使用 REST API 提出请求](#)

关于访问密钥

下面各部分将回顾您可以用于进行经身份验证的请求的访问密钥类型。

AWS 账户 访问密钥

账户访问密钥提供对账户拥有的 AWS 资源的完全访问权限。以下是访问密钥示例：

- 访问密钥 ID (20 个字符的字母数字字符串)。例如：AKIAIOSFODNN7EXAMPLE
- 秘密访问密钥 (40 个字符的字符串)。例如：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

访问密钥 ID 可唯一标识 AWS 账户。可以使用这些访问密钥向 Amazon S3 发送经身份验证的请求。

IAM 用户访问密钥

您可以为公司创建一个 AWS 账户；但组织中可能有多个员工需要访问组织的 AWS 资源。共享您的 AWS 账户 访问密钥会降低安全性，但为每个员工创建单独的 AWS 账户 可能也不太实际。此外，您可能无法轻松地共享资源 (例如，存储桶和对象)，因为它们由不同账户所拥有。要共享资源，您必须授予许可，这意味着需要额外的工作。

在这些情况中，您可以使用 AWS Identity and Access Management (IAM) 在您的 AWS 账户 下创建具有其自己的访问密钥的用户，然后附加 IAM 用户策略以授予这些用户适当的资源访问权限。为了更好地管理这些用户，IAM 允许您创建用户组并授予适用于该组中所有用户的组级权限。

这些用户被称为 IAM 用户，您可以在内创建和管理这些用户。AWS 父账户将控制用户访问的权限。AWS 父 AWS 账户 负责控制和支付 IAM 用户创建的任何资源。这些 IAM 用户可以使用他们自己的安全证书，向 Amazon S3 发送经身份验证的请求。有关在 AWS 账户 下创建和管理用户的更多信息，请参阅 [AWS Identity and Access Management 产品详细信息页面](#)。

临时安全凭证

除了创建具有自己的访问密钥的 IAM 用户之外，IAM 还可让您向任何 IAM 用户授予临时安全凭证（临时访问密钥和安全令牌），以便这些用户可以访问 AWS 服务和资源。您也可以在之外的系统中管理用户。AWS 这些用户称为联合身份用户。此外，用户还可以是您创建的能访问您的 AWS 资源的应用程序。

IAM 提供了 AWS Security Token Service API，供您用于请求临时安全证书。您可以使用 AWS STS API 或 AWS 开发工具包来请求这些凭证。API 将返回临时安全凭证（访问密钥 ID 和秘密访问密钥）和安全令牌。这些凭证仅在您请求它们时指定的持续时间内有效。使用访问密钥 ID 和私有密钥的方法与您在使用 AWS 账户 或 IAM 用户访问密钥发送请求时使用它们的方法相同。此外，您必须在每个请求中包含您发送至 Amazon S3 的令牌。

IAM 用户可以请求这些临时安全证书以供自己使用，也可以分发给联合用户或应用程序使用。为联合用户请求临时安全证书时，您必须提供用户名称和 IAM 策略（定义您想要与这些临时安全证书关联的许可）。联合用户可获取的许可不能超过请求临时证书的父 IAM 用户的许可。

您还可以使用这些临时安全证书向 Amazon S3 发出请求。API 库会使用这些凭证计算必要的签名值，以便对您的请求进行身份验证。如果您使用过期的证书发送请求，Amazon S3 会拒绝请求。

有关在您的 REST API 请求中使用临时安全凭证对请求进行签名的信息，请参阅 [对 REST 请求进行签名和身份验证 \(AWS 签名版本 2\)](#)。有关使用 AWS 开发工具包发送请求的信息，请参阅 [使用 AWS 开发工具包提出请求](#)。

有关 IAM 对临时安全凭证的支持的更多信息，请参阅《IAM 用户指南》中的[临时安全凭证](#)。

为了提高安全性，您可以通过配置存储桶策略，要求在访问您的 Amazon S3 资源时进行多重验证 (MFA)。有关信息，请参阅[存储桶策略示例：需要 MFA](#)。在要求进行 MFA 才能访问您的 Amazon S3 资源之后，访问这些资源的唯一方式是提供使用 MFA 密钥创建的临时凭证。有关更多信息，请参阅 [AWS Multi-Factor Authentication](#) 详细信息页面和《IAM 用户指南》中的[配置受 MFA 保护的 API 访问](#)。

请求终端节点

您可以向服务的预定义终端节点发送 REST 请求。有关所有 AWS 服务及其相应端点的列表，请参阅《AWS 一般参考》中的[区域和端点](#)。

通过 IPv6 向 Amazon S3 发出请求

除了 IPv4 协议之外，Amazon Simple Storage Service (Amazon S3) 还支持使用 Internet 协议版本 6 (IPv6) 访问 S3 存储桶的功能。Amazon S3 双堆栈终端节点支持通过 IPv6 和 IPv4 向 S3 存储桶发出请求。通过 IPv6 访问 Amazon S3 不额外收费。有关定价的更多信息，请参阅 [Amazon S3 定价](#)。

Note

以下信息适用于公有 Amazon S3 IPv6 端点。有关 AWS 虚拟私有云 (VPC) 端点的信息，请参阅[适用于 Amazon S3 的 AWS PrivateLink](#)。

主题

- [通过 IPv6 发出请求入门](#)
- [在 IAM 策略中使用 IPv6 地址](#)

- [测试 IP 地址兼容性](#)
- [使用 Amazon S3 双堆栈端点](#)

通过 IPv6 发出请求入门

要通过 IPv6 向 S3 存储桶发出请求，您需要使用双堆栈终端节点。下一节介绍如何使用双堆栈终端节点通过 IPv6 发出请求。

下面是在尝试通过 IPv6 访问存储桶之前应了解的部分事项：

- 访问存储桶的客户端和网络必须支持使用 IPv6。
- 虚拟托管类型和路径类型请求必须都受支持，以便进行 IPv6 访问。有关更多信息，请参阅 [Amazon S3 双堆栈端点](#)。
- 如果您 AWS Identity and Access Management (IAM) 用户或存储桶策略中使用源 IP 地址筛选，则需要更新策略以包括 IPv6 地址范围。有关更多信息，请参阅 [在 IAM 策略中使用 IPv6 地址](#)。
- 如果使用 IPv6，服务器访问日志文件以 IPv6 格式输出 IP 地址。您需要对用于分析 Amazon S3 日志文件的现有工具、脚本和软件进行更新，以便它们能够分析 IPv6 格式的 Remote IP 地址。有关更多信息，请参阅[使用服务器访问日志记录来记录请求](#)。

Note

如果在日志文件中遇到与 IPv6 地址相关的问题，请联系 [AWS 支持](#)。

使用双堆栈终端节点通过 IPv6 发出请求

您可以使用双堆栈终端节点和 Amazon S3 API 调用通过 IPv6 发出请求。无论您是通过 IPv6 还是通过 IPv4 访问 Amazon S3，Amazon S3 API 操作的工作方式都相同。性能也应该是相同的。

如果使用 REST API，您是直接访问双堆栈终端节点。有关更多信息，请参阅 [双堆栈端点](#)。

如果使用 AWS Command Line Interface (AWS CLI) 和 AWS 开发工具包，可使用参数或标志以更改为双堆栈终端节点。您还可以在配置文件中直接将双堆栈终端节点指定为覆盖 Amazon S3 终端节点。

您可以通过以下任一方式，使用双堆栈终端节点通过 IPv6 访问存储桶：

- AWS CLI，请参阅[从 AWS CLI 使用双堆栈端点](#)。
- 有关 AWS 开发工具包，请参阅 [从 AWS SDK 使用双堆栈端点](#)。

- REST API，请参阅[通过使用 REST API 向双堆栈终端节点发出请求](#)。

不可通过 IPv6 使用的功能

当通过来自 S3 存储桶的 IPv6: Static 网站托管访问 S3 存储桶时，以下功能目前不受支持。

在 IAM 策略中使用 IPv6 地址

在尝试使用 IPv6 访问存储桶之前，您必须确保用于 IP 地址筛选的所有 IAM 用户或 S3 存储桶策略已更新为包括 IPv6 地址范围。在开始使用 IPv6 时，未更新为处理 IPv6 地址的 IP 地址筛选策略可能导致客户端错误地丢失或获得存储桶访问权。有关使用 IAM 管理访问权限的更多信息，请参阅 [Amazon S3 的身份和访问管理](#)。

筛选 IP 地址的 IAM 策略使用 [IP 地址条件运算符](#)。下面的存储桶策略通过使用 IP 地址条件运算符确定允许的 IPv4 地址范围为 54.240.143.*。此范围之外的所有 IP 地址对存储桶 (examplebucket) 的访问都会被拒绝。由于所有 IPv6 地址都在允许范围之外，此策略会阻止 IPv6 地址访问 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"}
      }
    }
  ]
}
```

您可以将存储桶策略的 Condition 元素修改为同时允许 IPv4 (54.240.143.0/24) 和 IPv6 (2001:DB8:1234:5678::/64) 地址范围，如以下示例所示。您可以使用示例中所示的相同类型的 Condition 块来更新 IAM 用户和存储桶策略。

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

在使用 IPv6 之前，必须对使用 IP 地址筛选的所有相关 IAM 用户和存储桶策略进行更新。我们建议不要在存储桶策略中使用 IP 地址筛选。

您可以在 <https://console.aws.amazon.com/iam/> 上使用 IAM 控制台查看 IAM 用户策略。有关 IAM 的更多信息，请参阅 [IAM 用户指南](#)。有关编辑 S3 存储桶策略的信息，请参阅[添加存储桶策略](#)。

测试 IP 地址兼容性

如果使用 Linux/Unix 或 Mac OS X，则通过使用下列所示的 curl 命令，可以测试您能否通过 IPv6 访问双堆栈终端节点：

Example

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

您获得的信息如下例所示。如果通过 IPv6 连接，则连接的 IP 地址将会是 IPv6 地址。

```
* 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
```

如果使用 Microsoft 7 或 10，则通过使用下列所示的 ping 命令，可以测试您能否通过 IPv6 或 IPv4 访问双堆栈端点。

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

使用 Amazon S3 双堆栈端点

Amazon S3 双堆栈端点支持通过 IPv6 和 IPv4 向 S3 存储桶发出请求。本节介绍如何使用双堆栈端点。

主题

- [Amazon S3 双堆栈端点](#)
- [从 AWS CLI 使用双堆栈端点](#)
- [从 AWS SDK 使用双堆栈端点](#)
- [从 REST API 使用双堆栈端点](#)

Amazon S3 双堆栈端点

当您向双堆栈端点发出请求时，存储桶 URL 解析为 IPv6 或 IPv4 地址。有关如何通过 IPv6 访问存储桶的更多信息，请参阅[通过 IPv6 向 Amazon S3 发出请求](#)。

当使用 REST API 时，您通过使用端点名称 (URI) 直接访问 Amazon S3 端点。通过使用虚拟托管类型或路径类型端点名称，您可以通过双堆栈端点访问 S3 存储桶。Amazon S3 只支持区域双堆栈端点名称，这意味着，您必须在名称中指定区域。

双堆栈虚拟托管类型和路径类型端点名称使用以下命名惯例：

- 虚拟托管型双堆栈端点：

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

- 路径型双堆栈端点：

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

有关端点名称样式的更多信息，请参阅[访问和列出 Amazon S3 存储桶](#)。有关 Amazon S3 端点的列表，请参阅《AWS 一般参考》中的[区域和端点](#)。

Important

您可对双堆栈端点使用传输加速。有关更多信息，请参阅[开始使用 Amazon S3 Transfer Acceleration](#)。

Note

访问 Amazon S3 的两种类型的虚拟私有云 (VPC) 端点 (接口 VPC 端点和网关 VPC 端点) 现在支持双堆栈。有关 Amazon S3 的 VPC 端点的更多信息，请参阅[适用于 Amazon S3 的 AWS PrivateLink](#)。

如果使用 AWS Command Line Interface (AWS CLI) 和 AWS SDK，可使用参数或标志以更改为双堆栈端点。您还可以在配置文件中直接将双堆栈端点指定为覆盖 Amazon S3 端点。下面几节介绍如何从 AWS CLI 和 AWS SDK 使用双堆栈端点。

从 AWS CLI 使用双堆栈端点

本节提供用于向双堆栈端点发出请求的 AWS CLI 命令示例。有关设置 AWS CLI 的说明，请参阅[使用 AWS CLI 进行 Amazon S3 开发](#)。

在 AWS Config 文件中的配置文件中将配置值 `use_dualstack_endpoint` 设置为 `true`，使 `s3` 和 `s3api` AWS CLI 命令发出的所有 Amazon S3 请求定向到指定区域的双堆栈端点。您可以在配置文件或命令中使用 `--region` 选项指定区域。

通过 AWS CLI 使用双堆栈端点时，支持 `path` 和 `virtual` 寻址类型。配置文件中设置的寻址类型控制着主机名或 URL 中是否包含存储桶名称。默认情况下，CLI 尝试使用虚拟类型，但是如果需要，会改为使用路径类型。有关更多信息，请参阅[AWS CLI Amazon S3 配置](#)。

您也可以使用命令进行配置更改，如下例所示，在默认配置文件中将 `use_dualstack_endpoint` 设置为 `true`，将 `addressing_style` 设置为 `virtual`。

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

如果需要只对指定的 AWS CLI 命令 (并非所有命令) 使用双堆栈端点，可以使用以下任一方法：

- 您可以通过将任何 `--endpoint-url` 或 `https://s3.dualstack.aws-region.amazonaws.com` 命令的 `http://s3.dualstack.aws-region.amazonaws.com` 参数设置为 `s3` 或 `s3api` 来对每条命令使用双堆栈端点。

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

- 您可以在 AWS Config 文件中设置单独的配置文件。例如，创建一个将 `use_dualstack_endpoint` 设置为 `true` 的配置文件和一个不设置 `use_dualstack_endpoint` 的配置文件。在运行命令时，根据是否需要使用双堆栈端点来指定要使用的配置文件。

Note

当使用 AWS CLI 时，当前不能对双堆栈端点使用传输加速。但是，即将推出对 AWS CLI 的支持。有关更多信息，请参阅[启用和使用 S3 Transfer Acceleration](#)。

从 AWS SDK 使用双堆栈端点

本节提供一些示例，介绍如何使用 AWS SDK 来访问双堆栈端点。

适用于 Java 的 AWS SDK 双堆栈端点示例

以下示例演示如何使用 适用于 Java 的 AWS SDK 在创建 Amazon S3 客户端时启用双堆栈端点。

有关创建和测试有效 Java 示例的说明，请参阅《适用于 Java 的 AWS SDK 开发人员指南》中的[入门](#)。

```
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();
}
}
```

如果要在 Windows 上使用适用于 Java 的 AWS SDK，可能必须设置以下 Java 虚拟机 (JVM) 属性：

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

AWS .NET SDK 双堆栈端点示例

在使用 AWS SDK for .NET 时，请使用 `AmazonS3Config` 类来启用双堆栈端点，如下例所示。

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



```
}
```

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的[适用于 .NET 的 AWS SDK 入门](#)。

从 REST API 使用双堆栈端点

有关如何使用 REST API 向双堆栈端点发出请求的信息，请参阅[通过使用 REST API 向双堆栈终端节点发出请求](#)。

使用 AWS 开发工具包提出请求

主题

- [使用 AWS 账户 或 IAM 用户凭证发出请求](#)
- [使用 IAM 用户临时凭证发出请求](#)
- [使用联合身份用户临时凭证提出请求](#)

您可以使用 AWS 开发工具包或通过直接在应用程序中进行 REST API 调用，将经身份验证的请求发送到 Amazon S3。AWS 开发工具包 API 使用您提供的凭证来计算用于身份验证的签名。若您直接在应用程序中使用 REST API，您必须编写所需的代码来计算用于对您的请求进行身份验证的签名。有关可用 AWS 开发工具包的列表，请转到 [示例代码和库](#)。

使用 AWS 账户 或 IAM 用户凭证发出请求

您可以使用 AWS 账户 或 IAM 用户安全凭证向 Amazon S3 发送经身份验证的请求。本部分提供的示例演示了如何使用 适用于 Java 的 AWS SDK、适用于 .NET 的 AWS SDK 和 适用于 PHP 的 AWS SDK 发送经身份验证的请求。有关 AWS 可用开发工具包的列表，请转到 [示例代码和库](#)。

每个 AWS 开发工具包都使用开发工具包特有的凭证提供程序链来查找和使用凭证，并代表凭证所有者执行操作。所有这些凭证提供程序链的共同点是，它们都会寻找您的本地 AWS 凭证文件。

有关更多信息，请参阅以下主题：

主题

- [创建本地 AWS 凭证文件](#)
- [使用 AWS 开发工具包发送经身份验证的请求](#)
- [相关资源](#)

创建本地 AWS 凭证文件

为 AWS 开发工具包配置凭证的最简单方法是使用 AWS 凭证文件。如果您使用 AWS Command Line Interface (AWS CLI)，那么您可能已经配置了本地 AWS 凭证文件。否则，请按照以下步骤设置凭证文件：

1. 登录 AWS 管理控制台，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 创建一个新用户，其权限仅限于您希望您的代码有权访问的服务和操作。有关创建新用户的更多信息，请参阅[创建 IAM 用户（控制台）](#)，并按照直至步骤 8 的说明进行操作。

3. 选择 Download .csv (下载 .csv) 以保存 AWS 凭证的本地副本。
4. 在您的计算机上，导航至主目录，并创建 .aws 目录。在基于 Unix 的系统 (例如 Linux 或 OS X) 上，它在以下位置：

```
~/ .aws
```

在 Windows 上，它在以下位置：

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

5. 在 .aws 目录中，创建名为 credentials 的新文件。
6. 打开您从 IAM 控制台中下载的凭证 .csv 文件，并使用以下格式将其内容复制到 credentials 文件：

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

7. 保存 credentials 文件，并删除在步骤 3 中下载的 .csv 文件。

您的共享凭证文件现在已在本地计算机上配置完毕，可以与 AWS 开发工具包一起使用。

使用 AWS 开发工具包发送经身份验证的请求

使用 AWS 开发工具包发送经过身份验证的请求。有关发送经过身份验证的请求的更多信息，请参阅 [AWS 安全证书](#) 或 [IAM Identity Center 身份验证](#)。

Java

有关使用适用于 Java 的 AWS SDK 对请求进行身份验证的信息，请参阅《AWS SDK 和工具参考指南》中的 [使用共享配置和凭证文件全局配置 AWS SDK 和工具](#) 以及 [使用 AWS SDK 和工具进行身份验证和访问](#)。

.NET

使用 AWS 账户 或 IAM 用户凭证发送经身份验证的请求：

- 创建 AmazonS3Client 类的实例。
- 运行 AmazonS3Client 方法之一，以向 Amazon S3 发送请求。客户端将通过您提供的凭证生成所需的签名并将其包含在它发送到 Amazon S3 的请求中。

有关更多信息，请参阅[使用 AWS 账户或 IAM 用户凭证发出请求](#)。

 Note

- 您可以在不提供安全凭证的情况下创建 AmazonS3Client 客户端。使用此客户端发送的请求是匿名的请求，它们不带签名。如果为不是公开可用的资源发送了匿名请求，Amazon S3 将返回错误。
- 您可以创建 AWS 账户 并创建所需的用户。您还可以管理这些用户的凭证。您需要这些凭证才能执行以下示例中的任务。有关更多信息，请参阅《适用于 .NET 的 SDK 开发人员指南》中的[配置 AWS 凭证](#)。

然后，您还可以将应用程序配置为主动检索配置文件和凭证，然后在创建 AWS 服务客户端时明确使用这些凭证。有关更多信息，请参阅《适用于 .NET 的 SDK 开发人员指南》中的[访问应用程序中的凭证和配置文件](#)。

以下 C# 示例说明如何执行上述任务。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的[适用于 .NET 的 AWS SDK 入门](#)。

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

此部分说明如何使用版本 3 的 适用于 PHP 的 AWS SDK 中的类来通过 AWS 账户 或 IAM 用户凭证发送经身份验证的请求。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到[适用于 Ruby 的 AWS 开发工具包 – 版本 2](#)。

以下 PHP 示例说明客户端如何使用您的安全凭证发出请求以列出您的账户的所有存储桶。

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

您可以在不提供安全凭证的情况下创建 S3Client 客户端。使用此客户端发送的请求是匿名的请求，它们不带签名。如果为不是公开可用的资源发送了匿名请求，Amazon S3 将返回错误。有关更多信息，请参阅 [适用于 PHP 的 AWS SDK 文档](#) 中的 [创建匿名客户端](#)。

Ruby

使用 适用于 Ruby 的 AWS SDK 的版本 3 调用 Amazon S3 之前，必须设置 AWS 访问凭证以便开发工具包用来验证您对存储桶和对象的访问权限。如果您在本地系统上的 AWS 凭证配置文件中设置了共享凭证，则适用于 Ruby 的开发工具包的版本 3 可以在无需您在代码中声明这些凭证的情况下使用它们。有关设置共享凭证的更多信息，请参阅[使用 AWS 账户或 IAM 用户凭证发出请求](#)。

以下 Ruby 代码段使用本地计算机上凭证文件中的共享 AWS 凭证，对要获取特定存储桶中的所有对象键名的请求进行身份验证。它将执行以下操作：

1. 创建 `Aws::S3::Client` 类的实例。
2. 使用 `list_objects_v2` 的 `Aws::S3::Client` 方法，通过枚举存储桶中的对象向 Amazon S3 发出请求。客户端将根据计算机上 AWS 凭证文件中的凭证生成所需的签名值，并将其包含在发送给 Amazon S3 的请求中。
3. 将对象键名阵列打印到终端。

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__
```

即使没有本地 AWS 凭证文件，您仍可以创建 `Aws::S3::Client` 资源并针对 Amazon S3 存储桶和对象运行代码。使用适用于 Ruby 的开发工具包的版本 3 发送的请求是匿名的，默认情况下没有签名。如果为不是公开可用的资源发送了匿名请求，Amazon S3 将返回错误。

可以针对适用于 Ruby 的开发工具包应用程序使用并扩展前一个代码段，如下面的更可靠的示例所示。


```
# 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

以下示例使用由 SDK for Go 从共享凭证文件中自动加载的 AWS 凭证。

```
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)
    }
}
}
```

相关资源

- [使用 AWS SDK 通过 Amazon S3 进行开发](#)
- [适用于 PHP 的 AWS SDK 面向 Amazon S3 Aws\S3\S3Client 类的](#)
- [适用于 PHP 的 AWS SDK 文档](#)

使用 IAM 用户临时凭证发出请求

AWS 账户 或 IAM 用户可以请求临时安全证书，然后使用它们来向 Amazon S3 发送经身份验证的请求。本节提供了一些示例，介绍了如何使用 适用于 Java 的 AWS SDK、.NET 和 PHP 获取临时安全证书，以及如何使用这些证书来对您发送至 Amazon S3 的请求进行身份验证。

Java

IAM 用户或 AWS 账户 可以使用采用 适用于 Java 的 AWS SDK 的临时安全凭证（请参阅 [提出请求](#)），然后使用这些凭证访问 Amazon S3。在指定会话持续时间结束后，这些凭证将过期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的[临时安全凭证](#)。有关发出请求的更多信息，请参阅[提出请求](#)。

获取临时安全凭证并访问 Amazon S3

1. 创建 `AWSecurityTokenService` 类的实例。
2. 通过调用安全令牌服务 (STS) 客户端的 `assumeRole()` 方法，检索所需角色的临时安全凭证。
3. 将临时安全凭证打包到 `BasicSessionCredentials` 对象中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。
4. 使用临时安全凭证创建 `AmazonS3Client` 类的实例。您可以使用此客户端向 Amazon S3 发送请求。如果使用过期凭证发送请求，Amazon S3 将返回错误。

Note

如果使用 AWS 账户 安全凭证获取临时安全凭证，则临时凭证的有效期只有一小时。只有当您使用 IAM 用户证书来请求会话时，您才可以指定会话持续时间。

以下示例列出了指定存储桶中的一组对象键。该示例将为会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。

如果要使用 IAM 用户凭证测试示例，则必须在 AWS 账户 下创建一个 IAM 用户。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的[创建您的第一个 IAM 用户和管理员组](#)。

有关创建和测试有效示例的说明，请参阅《适用于 Java 的 AWS SDK 开发人员指南》中的[入门](#)。

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

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

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

            // 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

IAM 用户或 AWS 账户 可以使用 适用于 .NET 的 AWS SDK 请求临时安全凭证，然后使用这些凭证来访问 Amazon S3。在会话持续时间结束后，这些凭证将过期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的[临时安全凭证](#)。有关发出请求的更多信息，请参阅[提出请求](#)。

获取临时安全凭证并访问 Amazon S3

1. 创建AWS Security Token Service客户端的实例 `AmazonSecurityTokenServiceClient`。
2. 通过调用您在上一步骤中创建的 STS 客户端的 `GetSessionToken` 方法，开始会话。您可以使用 `GetSessionTokenRequest` 对象向此方法提供会话信息。

此方法将返回您的临时安全凭证。

3. 将临时安全凭证打包在 `SessionAWSCredentials` 对象的实例中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。
4. 通过传入临时安全凭证创建 `AmazonS3Client` 类的实例。您可以使用此客户端向 Amazon S3 发送请求。如果您使用过期的凭证发送请求，Amazon S3 将返回错误。

Note

如果使用 AWS 账户 安全凭证获取临时安全凭证，则这些凭证仅在一小时内有效。仅当使用 IAM 用户凭证请求会话时，才能指定会话持续时间。

以下 C# 示例列出了指定存储桶中的对象键。为了展示这个过程，示例会为默认一小时的会话获取临时安全凭证，然后使用这些凭证来向 Amazon S3 发送经身份验证的请求。

如果要使用 IAM 用户凭证测试示例，则必须在 AWS 账户 下创建一个 IAM 用户。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的[创建您的第一个 IAM 用户和管理员组](#)。有关发出请求的更多信息，请参阅[提出请求](#)。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的[适用于 .NET 的 AWS SDK 入门](#)。

```
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

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到[适用于 Ruby 的 AWS 开发工具包 – 版本 2](#)。

IAM 用户或 AWS 账户 可使用 适用于 PHP 的 AWS SDK 的版本 3 请求临时安全凭证。之后，它可使用临时凭证访问 Amazon S3。这些凭证将在会话持续时间结束时到期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的[临时安全凭证](#)。有关发出请求的更多信息，请参阅[提出请求](#)。

 Note

如果您使用 AWS 账户安全凭证获取临时安全凭证，则临时安全凭证的有效期限仅为一个小时。只有当您使用 IAM 用户证书来请求会话时，您才可以指定会话持续时间。

Example

以下 PHP 示例使用临时安全凭证列出指定存储桶中的对象键。该示例将为默认一小时的会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到[适用于 Ruby 的 AWS 开发工具包 – 版本 2](#)。

如果要使用 IAM 用户凭证测试示例，则必须在 AWS 账户下创建一个 IAM 用户。有关如何创建 IAM 用户的信息，请参阅《IAM 用户指南》中的[创建您的第一个 IAM 用户和管理员组](#)。有关在使用 IAM 用户凭证请求会话时设置会话持续时间的示例，请参阅[使用 IAM 用户临时凭证发出请求](#)。

```
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

IAM 用户或 AWS 账户 可以使用 适用于 Ruby 的 AWS SDK 请求临时安全凭证，然后使用这些凭证来访问 Amazon S3。在会话持续时间结束后，这些凭证将过期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的[临时安全凭证](#)。有关发出请求的更多信息，请参阅[提出请求](#)。

Note

如果您使用 AWS 账户 安全凭证获取临时安全凭证，则临时安全凭证的有效期限仅为一个小时。只有当您使用 IAM 用户证书来请求会话时，您才可以指定会话持续时间。

以下 Ruby 示例将创建一个临时用户来列出指定存储桶中的项目 1 小时。要使用此示例，则必须具有 AWS 凭证，此类凭证具有创建新的 AWS Security Token Service (AWS STS) 客户端和列出 Amazon S3 存储桶所需的权限。

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

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

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts "Error while determining whether the user ' \
    '#{user_name}' exists: #{e.message}"
end

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

```

    puts "Error while creating the user '#{user_name}': #{e.message}"
  end

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

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

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')

```

```

#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

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

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

```

```
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

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

使用联合身份用户临时凭证提出请求

您可以请求临时安全凭证并将它们提供给需要访问您的 AWS 资源的联合身份用户或应用程序。本节提供了一些示例，介绍了如何使用 AWS 开发工具包为联合身份用户或应用程序获取临时安全证书以及如何使用这些证书向 Amazon S3 发送经身份验证的请求。有关可用 AWS 开发工具包的列表，请参阅 [示例代码和库](#)。

Note

AWS 账户 和 IAM 用户皆可请求适用于联合身份用户的临时安全证书。然而，为了提高安全性，只有具有所需许可的 IAM 用户才能请求这些临时证书，这样可确保联合用户能够从请求的 IAM 用户中获得最多的许可。在某些应用程序中，您可能会发现创建一个具有特定权限的 IAM 用户以专门用于向联合身份用户和应用程序授予临时安全凭证比较合适。

Java

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。请求这些临时凭证时，必须提供用户名和 IAM 策略（描述要授予的资源权限）。默认情况下，会话的持续时间为一个小时。您可以在为联合身份用户和应用程序请求临时安全凭证时，显式地设置其他持续时间值。

Note

为了提高请求适用于联合身份用户和应用程序的临时安全凭证时的安全性，建议使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关更多信息，请参阅 [AWS Identity and Access Management 常见问题](#)。

要提供安全凭证并发送经身份验证的请求以访问资源，请执行以下操作：

- 创建 `AWSecurityTokenServiceClient` 类的实例。
- 通过调用安全令牌服务 (STS) 客户端的 `getFederationToken()` 方法启动会话。提供会话信息，包括要附加到临时凭证的用户名和 IAM 策略。您可以提供可选的会话持续时间。此方法将返回您的临时安全凭证。
- 将临时安全凭证打包在 `BasicSessionCredentials` 对象的实例中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。

- 使用临时安全凭证创建 AmazonS3Client 类的实例。您可以使用此客户端向 Amazon S3 发送请求。如果您使用过期的凭证发送请求，Amazon S3 将返回错误。

Example

该示例列出了指定 S3 存储桶中的键。在该示例中，您将为联合身份用户的时长两个小时的会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。要运行该示例，您需要创建具有附加策略（允许用户请求临时安全凭证和列出 AWS 资源）的 IAM 用户。以下策略将实现此操作：

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

有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的[创建您的第一个 IAM 用户和管理员组](#)。

创建 IAM 用户并附加上述策略之后，您可以运行以下示例。有关创建和测试有效示例的说明，请参阅《适用于 Java 的 AWS SDK 开发人员指南》中的[入门](#)。

```
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

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。请求这些临时凭证时，必须提供用户名和 IAM 策略（描述要授予的资源权限）。默认情况下，会话的持续时间为一个小时。您可以在为联合身份用户和应用程序请求临时安全凭证时，显式地设置其他持续时间值。有关发送经身份验证的请求的信息，请参阅[提出请求](#)。

Note

当请求适用于联合身份用户和应用程序的临时安全凭证时，为了提高安全性，建议使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关更多信息，请参阅 [AWS Identity and Access Management 常见问题](#)。

您应当执行以下操作：

- 创建 AWS Security Token Service 客户端 `AmazonSecurityTokenServiceClient` 类的实例。
- 通过调用 STS 客户端的 `GetFederationToken` 方法开始会话。您需要提供会话信息，包括要附加到临时凭证的用户名和 IAM 策略。（可选）您可以提供会话持续时间。此方法将返回您的临时安全凭证。
- 将临时安全凭证打包在 `SessionAWSCredentials` 对象的实例中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。
- 通过传递临时安全凭证创建 `AmazonS3Client` 类的实例。您可使用此客户端向 Amazon S3 发送请求。如果您使用过期的凭证发送请求，Amazon S3 将返回错误。

Example

以下 C# 示例列出了指定存储桶中的密钥。在此示例中，将为联合身份用户 (User1) 的时长两个小时的会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。

- 在此练习中，您将创建具有最低权限的 IAM 用户。通过使用此 IAM 用户的凭证，可为其他用户请求临时凭证。此示例仅列出特定存储桶中的对象。创建附加了以下策略的 IAM 用户：

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

该策略允许 IAM 用户请求临时安全凭证和用于仅列出 AWS 资源的访问权限。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的[创建您的 IAM 用户和管理员组](#)。

- 使用 IAM 用户安全凭证测试以下示例。该示例将使用临时安全证书向 Amazon S3 发送经身份验证的请求。该示例在为联合身份用户 (User1) 请求临时安全凭证时指定了以下策略，该策略对列出特定存储桶 (YourBucketName) 中的对象设置了访问权限限制。您必须更新策略并提供您自己的现有存储桶名称。

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

- Example

更新以下示例并提供在前面的联合身份用户访问策略中指定的存储桶名称。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的[适用于 .NET 的 AWS SDK 入门](#)。

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

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

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

```

private static async Task ListObjectsAsync()
{
    try
    {
        Console.WriteLine("Listing objects stored in a bucket");
        // Credentials use the default AWS SDK for .NET credential search
chain.

        // On local development machines, this is your default profile.
        SessionAWSCredentials tempCredentials =
            await GetTemporaryFederatedCredentialsAsync();

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

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

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
    }
}

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

```

```

        AmazonSecurityTokenServiceClient stsClient =
            new AmazonSecurityTokenServiceClient(
                config);

        GetFederationTokenRequest federationTokenRequest =
            new GetFederationTokenRequest();
        federationTokenRequest.DurationSeconds = 7200;
        federationTokenRequest.Name = "User1";
        federationTokenRequest.Policy = @"{
            ""Statement"":
            [
                {
                    ""Sid"":""Stmt1311212314284"",
                    ""Action"":[""s3:ListBucket""],
                    ""Effect"":""Allow"",
                    ""Resource"":""arn:aws:s3::" + bucketName + @""
                }
            ]
        }";

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

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

        return sessionCredentials;
    }
}

```

PHP

本主题介绍如何使用 适用于 PHP 的 AWS SDK 的版本 3 中的类请求适用于联合身份用户和应用程序的临时安全凭证，并使用这些凭证访问 Amazon S3 中存储的资源。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到[适用于 Ruby 的 AWS 开发工具包 – 版本 2](#)。

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。请求这些临时凭证时，必须提供用户名和 IAM 策略（描述要授予的资源权限）。当会话持续时间过期时，这些凭证将过期。默认情况下，会话的持续时间为一个小时。可以在请求适用于联合身份用户和应用程序的临时安全凭证时为持续时间显式设置不同的值。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的[临时安全凭证](#)。有关向联合身份用户和应用程序提供临时安全凭证的信息，请参阅[提出请求](#)。

为了提高请求适用于联合身份用户和应用程序的临时安全凭证时的安全性，建议使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关联合身份的更多信息，请参阅 [AWS Identity and Access Management 常见问题](#)。

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到[适用于 Ruby 的 AWS 开发工具包 – 版本 2](#)。

Example

以下 PHP 示例列出了指定存储桶中的密钥。在此示例中，您将获取联合身份用户 (User1) 的时长一小时的会话的临时安全凭证。然后，您使用临时安全凭证向 Amazon S3 发送经身份验证的请求。

为了提高请求适用于其他人的临时凭证时的安全性，请使用具有请求临时安全凭证的权限的 IAM 用户的安全凭证。要确保 IAM 用户仅向联合身份用户授予特定于应用程序的最低权限，还可以限制此 IAM 用户的访问权限。此示例仅列出特定存储桶中的对象。创建附加了以下策略的 IAM 用户：

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

该策略允许 IAM 用户请求临时安全凭证和用于仅列出 AWS 资源的访问权限。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的[创建您的第一个 IAM 用户和管理员组](#)。

您现在可以使用 IAM 用户安全证书测试下面的示例。该示例将使用临时安全凭证向 Amazon S3 发送经身份验证的请求。为联合身份用户 (User1) 请求临时安全凭证时，此示例指定了以下策略，该策略将访问权限限制为列出特定存储桶中的对象。请使用存储桶名称更新该策略。


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

在以下示例中，当指定策略资源时，请将 `YourBucketName` 替换为您的存储桶的名称：

```
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

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。从 IAM 服务请求临时凭证时，您必须提供用户名和 IAM 策略（描述您需要授予的资源权限）。默认情况下，会话的持续时间为一个小时。但是，如果您使用 IAM 用户证书来请求临时证书，您可以在为联合用户和应用程序请求临时安全证书时，显式地设置其他持续时间。有关适用于联合身份用户和应用程序的临时安全凭证的信息，请参阅[提出请求](#)。

Note

为了提高请求适用于联合身份用户和应用程序的临时安全凭证时的安全性，您可能需要使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关更多信息，请参阅 [AWS Identity and Access Management 常见问题](#)。

Example

以下 Ruby 代码示例允许具有一组有限权限的联合身份用户列出指定存储桶中的键。

```
# 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
  end
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' => 'Stmnt1',
        '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__

```

使用 REST API 提出请求

本节介绍如何使用 REST API 向 Amazon S3 终端节点发出请求。有关 Amazon S3 端点的列表，请参阅《AWS 一般参考》中的[区域和端点](#)。

为 REST API 请求构建 S3 主机名

Amazon S3 终端节点遵循如下所示的结构：

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

Amazon S3 访问点终端节点和双堆栈终端节点也遵循标准结构：

- Amazon S3 访问点 -s3-accesspoint.*Region*.amazonaws.com
- 双堆栈 – s3.dualstack.*Region*.amazonaws.com

有关 Amazon S3 区域和端点的完整列表，请参阅《Amazon Web Services 一般参考》中的[Amazon S3 端点和限额](#)。

虚拟托管类型和路径类型请求

在使用 REST API 发出请求时，您可以使用 Amazon S3 终端节点的虚拟托管类型或路径类型 URI。有关更多信息，请参阅[路径类型请求](#)。

Example 虚拟托管类型请求

下面是要求从美国西部（俄勒冈）区域中名为 puppy.jpg 的存储桶删除 examplebucket 文件的虚拟托管类型请求的示例。有关虚拟托管类型请求的更多信息，请参阅[路径类型请求](#)。

```
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 路径类型请求

下面是同一请求的路径类型版本示例。

```
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
```

目前，Amazon S3 在所有 AWS 区域中同时支持虚拟托管类型和路径类型 URL 访问。但是，路径类型 URL 将来会停用。有关更多信息，请参阅下面的重要提示。

有关路径类型请求的更多信息，请参阅[路径类型请求](#)。

Important

更新 (2020 年 9 月 23 日) – 为确保客户有时间转换到虚拟托管类型 URL，我们决定推迟弃用路径类型 URL。有关更多信息，请参阅 AWS 新闻博客中的 [Amazon S3 路径弃用计划 – 案例的其余部分](#)。

通过使用 REST API 向双堆栈终端节点发出请求

在使用 REST API 时，您可以通过使用虚拟托管类型或路径类型终端节点名称 (URI) 直接访问双堆栈终端节点。所有 Amazon S3 双堆栈终端节点名称中都包含区域。与标准的仅支持 IPv4 的终端节点不同，虚拟托管类型和路径类型终端节点都使用特定于区域的终端节点名称。

Example 虚拟托管类型双堆栈终端节点请求

如下例所示，您可以在 REST 请求中使用虚拟托管类型终端节点，该示例从美国西部（俄勒冈）区域中名为 puppy.jpg 的存储桶检索 examplebucket 对象。

```
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 路径类型双堆栈终端节点请求

如下例所示，您也可以使用路径类型终端节点。

```
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
```

有关双堆栈终端节点的更多信息，请参阅[使用 Amazon S3 双堆栈端点](#)。

有关如何使用 REST API 发出请求的更多信息，请参阅以下主题。

主题

- [请求重定向和 REST API](#)
- [请求路由选择](#)

请求重定向和 REST API

主题

- [重定向和 HTTP 用户代理](#)
- [重定向和 100-Continue](#)
- [重定向示例](#)

此部分介绍如何使用 Amazon S3 REST API 处理 HTTP 重定向。有关 Amazon S3 重定向的一般信息，请参阅《Amazon Simple Storage Service API 参考》中的[提出请求](#)。

重定向和 HTTP 用户代理

使用 Amazon S3 REST API 的程序既可以处理应用程序层上的重定向，也可以处理 HTTP 层上的重定向。许多 HTTP 客户端库和用户代理经过配置可以自动正确处理重定向；然而，许多其他 HTTP 客户端库和用户代理的重定向实施会不正确或不完整。

在依赖库完成重定向要求之前，请测试以下案例：

- 验证所有 HTTP 请求标头都已正确地包含在重定向的请求中 (收到重定向后的第二个请求)，包括诸如授权和日期等 HTTP 标准。
- 验证诸如 PUT 和 DELETE 等非 GET 的重定向运行正常。
- 验证大型 PUT 请求正确地遵循了重定向。

- 如果需要很长时间才能收到 100-Continue 响应，请验证 PUT 请求正确地遵循了重定向。

当 HTTP 请求方法不是 GET 或 HEAD 时，严格遵循 RFC 2616 的 HTTP 用户代理可能会在遵循重定向之前，请求显式确认。自动遵循 Amazon S3 生成的重定向通常是安全的，因为系统只会发出指向 amazonaws.com 域中主机的重定向，并且重定向请求的效果与原始请求的效果相同。

重定向和 100-Continue

要简化重定向处理过程、提高工作效率以及避免重复支付与发送重定向请求正文相关的费用，请将您的应用程序配置为对 PUT 操作使用 100-Continue。当应用程序使用 100-Continue 时，它不会发送请求正文，直到收到认可。如果根据标头拒绝了消息，则不会发送消息的正文。有关 100-Continue 的更多信息，请参阅 [RFC 2616 Section 8.2.3](#)

Note

根据 RFC 2616，在未知的 HTTP 服务器中使用 Expect: Continue 时，您应该直接发送请求正文，不要等待某个无限期的时段。这是因为某些 HTTP 服务器不能识别 100-Continue。但是，Amazon S3 能够识别您的请求是否包含了 Expect: Continue，并且将使用临时的 100-Continue 状态或最终的状态代码进行响应。此外，收到临时的 100 继续操作指示后，不会发生重定向错误。这可以帮助您避免在编写请求正文时，收到重定向响应。

重定向示例

本节提供了使用 HTTP 重定向和 100-Continue 执行客户端和服务端交互的示例。

下面是用于 quotes.s3.amazonaws.com 存储桶的示例 PUT。

```
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 将返回以下内容：

```
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>
```

客户端将遵循重定向响应并向 quotes.s3-4c25d83b.amazonaws.com 临时终端节点发布新的请求。

```
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 将返回 100-Continue，以指示客户端应继续发送请求正文。

```
HTTP/1.1 100 Continue
```

客户端将发送请求正文。

```
ha ha\n
```

Amazon S3 将返回最终响应。

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

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

请求路由选择

向使用 [CreateBucket](#) API 创建的存储桶发送请求且包含 [CreateBucketConfiguration](#) 的程序必须支持重新导向。此外，不考虑 DNS TTL 的某些客户端可能会遇到问题。

本节描述了在设计要和 Amazon S3 结合使用的服务或应用程序时，应考虑的路由选择和 DNS 问题。

请求重定向和 REST API

Amazon S3 使用域名系统 (DNS) 将请求路由到可以处理它们的设备。此系统处于有效的工作状态，但可能会发生临时路由错误。如果请求进入错误的 Amazon S3 位置，Amazon S3 将使用临时重定向进行响应，以告知请求者将请求重新发送到新的终端节点。如果请求的格式不正确，Amazon S3 将使用永久重定向来提供有关如何正确执行请求的说明。

Important

要使用此功能，您必须拥有一个可以处理 Amazon S3 重定向响应的应用程序。唯一的例外是专用于处理存储桶（在不使用 `<CreateBucketConfiguration>` 的情况下创建）的应用程序。有关位置约束的更多信息，请参阅[访问和列出 Amazon S3 存储桶](#)。

对于在 2019 年 3 月 20 日发布的所有区域，如果请求到达错误的 Amazon S3 位置，则 Amazon S3 返回 HTTP 400 错误请求错误。

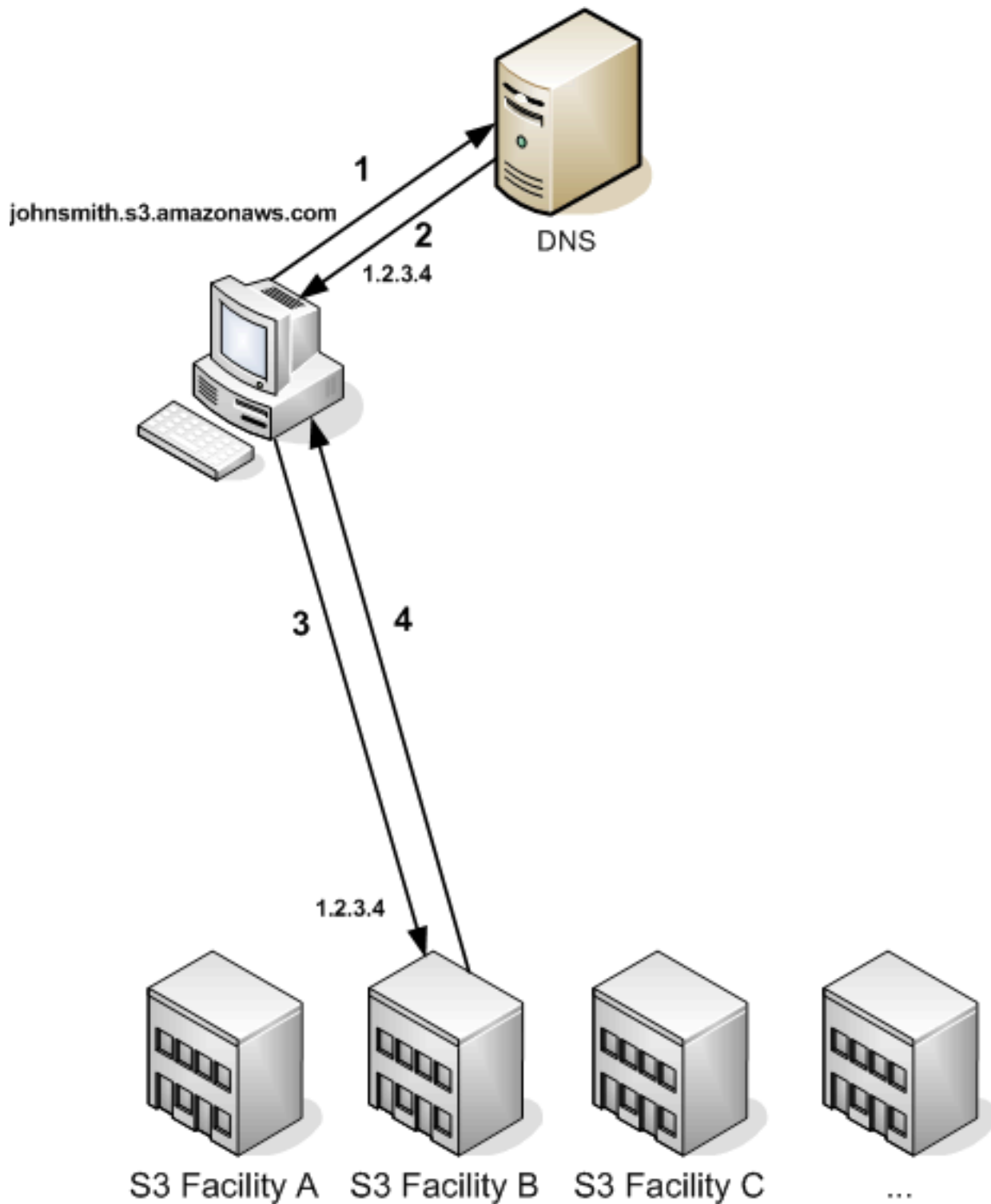
有关启用或禁用 AWS 区域的更多信息，请参阅《AWS 一般参考》中的[AWS 区域和端点](#)。

主题

- [DNS 路由选择](#)
- [临时请求重定向](#)
- [永久请求重定向](#)
- [请求重定向示例](#)

DNS 路由选择

DNS 路由选择会将请求路由到合适的 Amazon S3 设备。下面的图和过程显示 DNS 路由选择的示例。



DNS 路由请求步骤

1. 客户端创建 DNS 请求来获取存储在 Amazon S3 上的对象。
2. 客户端收到可以处理请求的设备的一个或多个 IP 地址。在本示例中，IP 地址用于设备 B。

3. 客户端创建面向 Amazon S3 设备 B 的请求。
4. 设备 B 将对象的副本返回给客户端。

临时请求重定向

临时重定向是一种错误响应类型，指示请求者应将请求重新发送到其他终端节点。由于 Amazon S3 的分布式特性，请求可能暂时会被路由到错误的设备。创建或删除存储桶后，很可能会立即发生此情况。

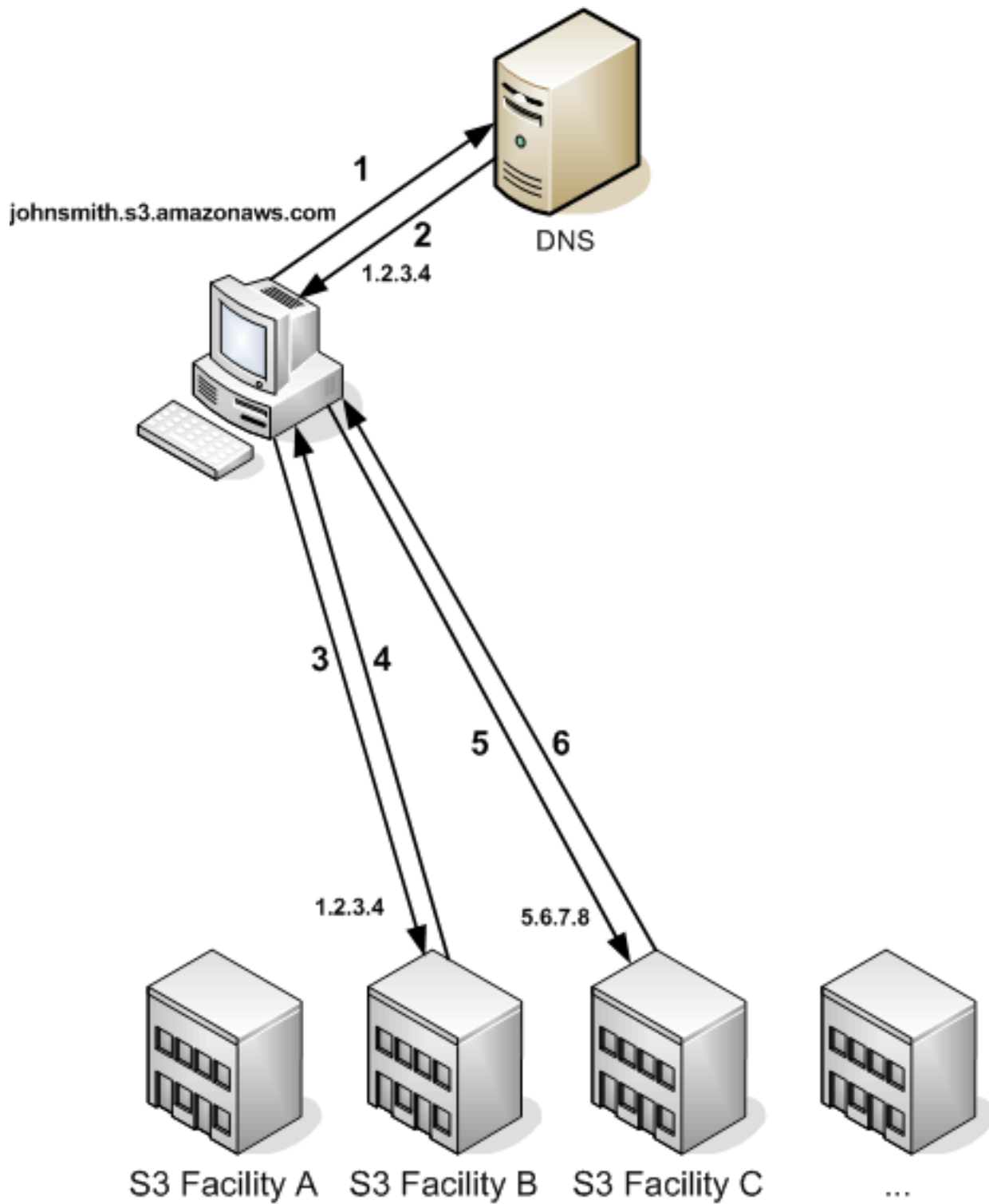
例如，如果您创建新的存储桶并立即向该存储桶发送请求，则根据存储桶的位置约束，您可能收到临时重定向。如果您在美国东部（弗吉尼亚北部）AWS 区域中创建了该存储桶，则看不到此重定向，因为这也是默认的 Amazon S3 终端节点。

但如果存储桶是在其他任何区域创建的，则对该存储桶的任何请求都会转到默认终端节点，同时会传播该存储桶的 DNS 条目。默认终端节点会将此请求重定向到具有 HTTP 302 响应的正确终端节点。临时重定向包含指向正确设备的 URI，您可以使用它来立即重新发送请求。

Important

不要重复使用之前重定向响应提供的终端节点。它似乎能够正常工作（甚至可以持续很长一段时间），但它可能会产生不可预测的结果，且最终会在不另行通知的情况下导致失败。

下面的图和过程显示临时重定向的示例。



临时请求重定向步骤

1. 客户端创建 DNS 请求来获取存储在 Amazon S3 上的对象。
2. 客户端收到可以处理请求的设备的一个或多个 IP 地址。

3. 客户端创建面向 Amazon S3 设备 B 的请求。
4. 设备 B 将返回一个重定向，指示可从位置 C 获取对象。
5. 客户端会将请求重新发送到设备 C。
6. 设备 C 将返回对象的副本。

永久请求重定向

永久重定向指示请求中资源的寻址不正确。例如，如果您使用路径类型请求来访问使用 `<CreateBucketConfiguration>` 创建的存储桶，将发生永久重定向。有关更多信息，请参阅[访问和列出 Amazon S3 存储桶](#)。

为帮助您在开发期间查找这些错误，此类型的重定向不包含位置 HTTP 标头 (该标头允许您自动遵循请求，以定向到正确的位置)。参考生成的 XML 错误文档，获取有关使用正确的 Amazon S3 终端节点的帮助。

请求重定向示例

以下是临时请求重定向响应的示例。

REST API 临时请求重定向

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

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

SOAP API 临时请求重定向

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS 开发工具包。

```
<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-gz4tb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

DNS 注意事项

Amazon S3 的设计要求之一是实现高可用性。为达到此要求，我们采取的一个方法是根据需要更新 DNS 中与 Amazon S3 终端节点相关联的 IP 地址。这些更改将自动在短期有效的客户端而不是某些长期有效的客户端中反映出来。长期有效的客户端将需要采取特殊的操作，定期重新解析 Amazon S3 终端节点，才能从这些更改中获益。有关虚拟机 (VM) 的更多信息，请参阅以下内容：

- 对于 Java，Sun 的 JVM 在默认情况下将永久缓存 DNS 查找结果；请参阅 [InetAddress 文档](#) 中的“[InetAddress 缓存](#)”一节，以了解有关如何更改此行为的信息。
- 对于 PHP，在最常见的部署配置中运行的永久性 PHP VM 将缓存 DNS 查找结果，直到 VM 重新启动。请参阅 [getHostByName PHP 文档](#)。

使用 AWS CLI 进行 Amazon S3 开发

Amazon S3 AWS CLI 命令在《AWS CLI Command Reference》中整理为不同的集合，每个集合都有其自己的可用命令。如果您在一个集合中未找到所查找的命令，请检查其它集合之一。不同的集合如下所示：

- `s3`：描述用于处理对象和存储桶的高级命令。其中包括复制、列出和移动操作。有关此集合中命令的完整列表，请参阅《AWS CLI Command Reference》中的 [s3](#)。
- `s3api`：描述用于管理 S3 资源（例如存储桶、对象、会话和分段上传）的低级命令。这些命令对应于 [Amazon S3 API](#) 操作。有关此集合中 CLI 命令的完整列表，请参阅《AWS CLI Command Reference》中的 [s3api](#)。
- `s3control`：描述用于管理所有其它 S3 资源的低级命令，例如访问权限管控、Storage Lens 组和 Amazon S3 on Outposts 存储桶。这些命令对应于 [Amazon S3 Control](#) API 操作。有关此集合中 CLI 命令的完整列表，请参阅《AWS CLI Command Reference》中的 [s3control](#)。

如果您在设置 AWS CLI 时遇到问题，请参阅 AWS re:Post 知识中心中的[如何解决 Amazon S3 中的“无法找到凭证”错误](#)。

Note

AWS 中的服务（如 Amazon S3）要求您在访问时提供凭证。然后该服务才能确定您是否有权访问该服务所拥有的资源。控制台要求您的密码。您可以为您的 AWS 账户创建访问密钥以访问 AWS CLI 或 API。但是，我们不建议使用您的 AWS 账户的凭证访问 AWS。相反，我们建议您使用 AWS Identity and Access Management (IAM)。创建 IAM 用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的 IAM 用户授予管理权限。然后，您就可以使用专门的 URL 和该 IAM 用户的凭证来访问 AWS。有关说明，请转到《IAM 用户指南》中的[创建您的第一个 IAM 用户和管理员组](#)。

了解有关 AWS CLI 的更多信息

要了解有关 AWS 的更多信息，请参阅以下资源：

- [AWS Command Line Interface](#)
- [AWS Command Line Interface 版本 2 的用户指南](#)

使用 AWS SDK 通过 Amazon S3 进行开发

AWS 软件开发工具包（SDK）适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地了解其首选语言构建应用程序。

Note

您可以使用 AWS Amplify 进行 Web 和移动应用程序的端到端全栈开发。Amplify Storage 将文件存储和管理功能无缝集成到基于 Amazon S3 构建的前端 Web 和移动应用程序中。有关更多信息，请参阅《Amplify 用户指南》中的 [Storage](#)。

SDK 文档	代码示例
适用于 C++ 的 AWS SDK	适用于 C++ 的 AWS SDK 代码示例
AWS CLI	AWS CLI 代码示例
适用于 Go 的 AWS SDK	适用于 Go 的 AWS SDK 代码示例
适用于 Java 的 AWS SDK	适用于 Java 的 AWS SDK 代码示例
适用于 JavaScript 的 AWS SDK	适用于 JavaScript 的 AWS SDK 代码示例
适用于 Kotlin 的 AWS SDK	适用于 Kotlin 的 AWS SDK 代码示例
适用于 .NET 的 AWS SDK	适用于 .NET 的 AWS SDK 代码示例
适用于 PHP 的 AWS SDK	适用于 PHP 的 AWS SDK 代码示例
AWS Tools for PowerShell	Tools for PowerShell 代码示例
适用于 Python (Boto3) 的 AWS SDK	适用于 Python (Boto3) 的 AWS SDK 代码示例
适用于 Ruby 的 AWS SDK	适用于 Ruby 的 AWS SDK 代码示例
适用于 Rust 的 AWS SDK	适用于 Rust 的 AWS SDK 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
适用于 Swift 的 AWS SDK	适用于 Swift 的 AWS SDK 代码示例

有关特定示例，请参阅[适用于使用 AWS 软件开发工具包的 Amazon S3 的代码示例](#)。

SDK 编程接口

每个 AWS SDK 都提供一个或多个编程接口来与 Amazon S3 结合使用。每个 SDK 都为 Amazon S3 提供一个低级别接口，其方法与 API 操作非常相似。一些 SDK 为 Amazon S3 提供高级别接口，这类抽象旨在简化常见的应用场景。

例如，当您使用低级别 API 操作来执行分段上传时，您使用一个操作来启动上传，使用另一个操作来上传各个分段，然后使用最后一个操作来完成上传。高级别分段上传 API 操作可让您在单个 API 调用中执行上传所需的所有操作。有关示例，请参阅《Amazon S3 用户指南》中的[使用分段上传操作上传对象](#)。

低级别 API 操作可以更好地控制上传。如果您需要暂停和恢复上传，在上传期间更改分段的大小，或者在事先不知道数据大小的情况下开始上传，建议您使用低级别 API。

在请求身份验证中指定签名版本

Amazon S3 在大多数 AWS 区域中只支持 AWS Signature Version 4。不过，在某些较旧的 AWS 区域中，Amazon S3 同时支持 Signature Version 4 和 Signature Version 2。但是，签名版本 2 将要被关闭（弃用）。有关结束对签名版本 2 支持的更多信息，请参阅[已为 Amazon S3 关闭（弃用）AWS 签名版本 2](#)。

有关所有 Amazon S3 区域的列表以及这些区域支持的 Signature Version，请参阅《AWS 一般参考》中的[区域和终端节点](#)。

对于所有 AWS 区域，默认情况下，AWS 开发工具包使用 Signature Version 4 对请求进行身份验证。如果使用 2016 年 5 月之前发布的 AWS 开发工具包，您可能需要请求 Signature Version 4，如下表所示。

开发工具包	请求使用签名版本 4 进行请求身份验证
AWS CLI	对于默认配置文件，运行以下命令： <pre>\$ aws configure set default.s3.signature_version s3v4</pre> 对于自定义配置文件，运行以下命令： <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>

开发工具包	请求使用签名版本 4 进行请求身份验证
Java 软件开发工具包	在代码中添加以下内容： <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> 或者在命令行中指定以下内容： <pre>-Dcom.amazonaws.services.s3.enableV4</pre>
JavaScript 开发工具包	在构建客户端时，将 <code>signatureVersion</code> 参数设置为 <code>v4</code>： <pre>var s3 = new AWS.S3({signatureVersion: 'v4'});</pre>
PHP 开发工具包	在为 PHP 开发工具包 v2 构建 Amazon S3 服务客户端时，将 <code>signature</code> 参数设置为 <code>v4</code>： <pre><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre> 使用 PHP 开发工具包 v3 时，在构建 Amazon S3 服务客户端期间将 <code>signature_version</code> 参数设置为 <code>v4</code>： <pre><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
Python-Boto 开发工具包	在默认配置文件 <code>boto</code> 中指定以下内容： <pre>[s3] use-sigv4 = True</pre>

开发工具包	请求使用签名版本 4 进行请求身份验证
Ruby 开发工具包	Ruby 开发工具包 – 版本 1：在构建客户端时，将 <code>:s3_signature_version</code> 参数设置为 <code>:v4</code>： <pre>s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> Ruby 开发工具包 – 版本 3：在构建客户端时，将 <code>signature_version</code> 参数设置为 <code>v4</code>： <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>
.NET 开发工具包	在创建 Amazon S3 客户端之前将以下内容添加到代码中： <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> 或将以下内容添加到配置文件中： <pre><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

已为 Amazon S3 关闭 (弃用) AWS 签名版本 2

但是，Amazon S3 即将关闭 (弃用) 签名版本 2。之后，Amazon S3 仅接受使用签名版本 4 签名的 API 请求。

本部分提供有关结束签名版本 2 支持的常见问题解答。

什么是签名版本 2/4，以及这对于签名请求意味着什么？

签名版本 2 或签名版本 4 签名过程用于对 Amazon S3 API 请求进行身份验证。签名请求使得 Amazon S3 可以确定谁在发送请求并保护您的请求免遭恶意人士侵害。

有关签署 AWS 请求的更多信息，请参阅《AWS 一般参考》中的[签署 AWS API 请求](#)。

进行了哪些更新？

我们当前支持使用签名版本 2 和签名版本 4 过程签名的 Amazon S3 API 请求。在此之后，Amazon S3 将仅接受使用签名版本 4 签名的请求。

有关签署 AWS 请求的更多信息，请参阅《AWS 一般参考》中的[签名版本 4 中的变化](#)。

为什么进行这些更新？

签名版本 4 使用签名密钥而非您的秘密访问密钥，提供了更好的安全性。当前所有 AWS 区域中都支持 Signature Version 4，而只有在 2014 年 1 月之前推出的区域中支持 Signature Version 2。此更新使得我们可以在所有区域中提供更为一致的体验。

如何确保我使用的是签名版本 4，需要进行哪些更新？

对请求进行签名所用的签名版本通常由客户端上的工具或开发工具包设置。默认情况下，AWS 开发工具包的最新版本使用 Signature Version 4。对于第三方软件，请联系您软件的相应支持团队以确认所需的版本。如果您向 Amazon S3 发送直接 REST 调用，则必须修改应用程序以使用签名版本 4 签名过程。

有关在转向 Signature Version 4 时所用 AWS 开发工具包版本的信息，请参阅[从签名版本 2 转向签名版本 4](#)。

有关将 Signature Version 4 与 Amazon S3 REST API 结合使用的信息，请参阅《Amazon Simple Storage Service API 参考》中的[对请求进行身份验证 \(AWS Signature Version 4\)](#)。

如果我不更新会怎样？

在此之后使用签名版本 2 签名的请求将无法通过 Amazon S3 的身份验证。请求者将看到错误，说明请求必须使用签名版本 4 签名。

如果我使用要求我签名 7 天以上的预签名 URL，是否应进行更改？

如果您使用了要求您签名 7 天以上的预签名 URL，目前无需任何操作。您可以继续使用 AWS Signature Version 2 对预签名 URL 进行签名并通过身份验证。我们将后续跟进并提供更多详情，说明如何为预签名 URL 场景迁移签名版本 4。

更多信息

- 有关使用 Signature Version 4 的更多信息，请参阅[签署 AWS API 请求](#)。
- 在[签名版本 4 中的变化](#)中可查看签名版本 2 与签名版本 4 之间的改动列表。

- 查看 AWS 论坛中的帖子：[AWS Signature Version 4 将取代 AWS Signature Version 2 用于签署 Amazon S3 API 请求](#)。
- 如果您有任何问题或疑虑，请联系 [支持](#)。

从签名版本 2 转向签名版本 4

如果您为 Amazon S3 API 请求身份验证使用签名版本 2，则应改为使用签名版本 4。对签名版本 2 的支持将结束，如[已为 Amazon S3 关闭（弃用）AWS 签名版本 2](#)中所述。

有关将 Signature Version 4 与 Amazon S3 REST API 结合使用的信息，请参阅《Amazon Simple Storage Service API 参考》中的[对请求进行身份验证 \(AWS Signature Version 4\)](#)。

下表列出了使用签名版本 4 (SigV4) 所需的开发工具包最低版本列表。如果您使用通过 AWS Java、JavaScript (Node.js) 或 Python (Boto/CLI) 开发工具包预签名 URL，则必须设置正确的 AWS 区域并在客户端配置中设置 Signature Version 4。有关在客户端配置中设置 SigV4 的信息，请参阅[在请求身份验证中指定签名版本](#)。

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4 ?	指向开发工具包文档的链接
适用于 Java 的 AWS SDK v1	升级至 Java 1.11.201+ 或 v2。	是	在请求身份验证中指定签名版本
适用于 Java 的 AWS SDK v2	无需开发工具包升级。	否	适用于 Java 的 AWS SDK
适用于 .NET 的 AWS SDK v1	升级至 3.1.10 或更高版本。	是	适用于 .NET 的 AWS SDK

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4 ?	指向开发工具包文档的链接
适用于 .NET 的 AWS SDK v2	升级至 3.1.10 或更高版本。	否	适用于 .NET 的 AWS SDK v2
适用于 .NET 的 AWS SDK v3	升级至 3.3.0.0 或更高版本。	是	适用于 .NET 的 AWS SDK v3
适用于 JavaScript 的 AWS SDK v1	升级至 2.68.0 或更高版本。	是	适用于 JavaScript 的 AWS SDK
适用于 JavaScript 的 AWS SDK v2	升级至 2.68.0 或更高版本。	是	适用于 JavaScript 的 AWS SDK
适用于 JavaScript 的 AWS SDK v3	当前无需任何操作。2019 年 3 季度升级至主要版本 V3。	否	适用于 JavaScript 的 AWS SDK
适用于 PHP 的 AWS SDK v1	建议升级至最新版本的 PHP，或者至少升级至已在 S3 客户端的配置中将签名参数设置为 v4 的 v2.7.4。	是	适用于 PHP 的 AWS SDK

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4 ?	指向开发工具包文档的链接
适用于 PHP 的 AWS SDK v2	建议升级至最新版本的 PHP，或者至少升级至已在 S3 客户端的配置中将签名参数设置为 v4 的 v2.7.4。	否	适用于 PHP 的 AWS SDK
适用于 PHP 的 AWS SDK v3	无需开发工具包升级。	否	适用于 PHP 的 AWS SDK
Boto2	升级至 Boto2 v2.49.0。	是	Boto 2 升级
Boto3	升级至 1.5.71 (Botocore)、1.4.6 (Boto3)。	是	Boto 3 – 适用于 Python 的 AWS 开发工具包
AWS CLI	升级至 1.11.108。	是	AWS Command Line Interface
AWS CLI v2 (预览)	无需开发工具包升级。	否	AWS Command Line Interface 版本 2
适用于 Ruby 的 AWS SDK v1	升级至 Ruby V3。	是	的 Ruby V3AWS

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4 ?	指向开发工具包文档的链接
适用于 Ruby 的 AWS SDK v2	升级至 Ruby V3。	是	的 Ruby V3AWS
适用于 Ruby 的 AWS SDK v3	无需开发工具包升级。	否	的 Ruby V3AWS
Go	无需开发工具包升级。	否	适用于 Go 的 AWS SDK
C++	无需开发工具包升级。	否	适用于 C++ 的 AWS SDK

AWS Tools for Windows PowerShell 或者 AWS Tools for PowerShell Core

如果您使用的模块版本早于 3.3.0.0，则必须升级到 3.3.0.0。

要获取版本信息，请使用 Get-Module cmdlet：

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

要更新 3.3.0.0 版本，请使用 Update-Module cmdlet：

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

您可以使用有效期超过 7 天的预签名 URL 以将签名版本 2 流量发送到其上。

获取 AWS 支持的 Amazon S3 请求 ID

每当您因为在 Amazon S3 中遇到错误或意外行为而联系 AWS 支持时，您必须提供与失败操作相关联的请求 ID。AWS 支持使用这些请求 ID 来帮助解决您遇到的问题。

请求 ID 成对出现，并在 Amazon S3 处理的每个响应（甚至是错误的响应）中返回并可通过详细日志访问。可通过许多常见的方法来获取您的请求 ID，包括 S3 服务器访问日志和 AWS CloudTrail 事件或数据事件。

在您恢复这些日志后，复制并保留这两个值，因为您在联系时需要它们。AWS 支持有关联系 AWS 支持的信息，请参阅[联系 AWS](#) 或 [AWS 支持文档](#)。

主题

- [使用 AWS CLI 获得请求 ID](#)
- [使用 Windows PowerShell 获取请求 ID](#)
- [使用 AWS CloudTrail 数据事件获取请求 ID](#)
- [使用 S3 服务器访问日志记录获取请求 ID](#)
- [使用 HTTP 获得请求 ID](#)
- [使用 Web 浏览器获得请求 ID](#)
- [使用 AWS SDK 获得请求 ID](#)

使用 AWS CLI 获得请求 ID

要在使用 AWS Command Line Interface (AWS CLI) 时获取您的请求 ID，请将 `--debug` 添加到您的命令中。例如，您应该在调试日志中看到 `x-amz-request-id` 和 `x-amz-id-2`，如下所示：

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

使用 Windows PowerShell 获取请求 ID

有关使用 Windows PowerShell 恢复日志的信息，请参阅 AWS 开发人员工具博客中的 [Response Logging in AWS Tools for Windows PowerShell](#)。

使用 AWS CloudTrail 数据事件获取请求 ID

使用 CloudTrail 数据事件配置为记录 S3 对象级别 API 操作的 Amazon S3 桶提供了有关用户、角色或 AWS 服务在 Amazon S3 桶中所采取操作的详细信息。您可以通过向 Athena 查询 CloudTrail 事件来识别 S3 请求 ID。

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [使用 CloudTrail 识别 Amazon S3 请求](#)。

使用 S3 服务器访问日志记录获取请求 ID

为 S3 服务器访问日志记录配置的 Amazon S3 桶为向桶发出的每个请求提供详细记录。您可以通过使用 Athena 查询服务器访问日志来识别 S3 请求 ID。

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [使用 Amazon Athena 查询访问日志中的请求](#)。

使用 HTTP 获得请求 ID

您可以在 HTTP 请求到达目标应用程序之前记录该请求的位数来获取您的请求 ID，即 x-amz-request-id 和 x-amz-id-2。可使用多种第三方工具来恢复 HTTP 请求的详细日志。选择您信任的某个工具，然后运行该工具，以便在发送出另一个 Amazon S3 HTTP 请求时侦听 Amazon S3 流量通过的端口。

对于 HTTP 请求，请求 ID 对将与以下内容类似：

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

Note

HTTPS 请求将加密并隐藏在大多数数据包捕获中。

使用 Web 浏览器获得请求 ID

大多数 Web 浏览器都具有您可用于查看请求标头的开发人员工具。

对于返回错误的基于 Web 浏览器的请求，请求 ID 对将与以下示例类似。

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

要从成功的请求中获取请求 ID 对，请使用浏览器的开发人员工具来查看 HTTP 响应标头。

使用 AWS SDK 获得请求 ID

以下各节包含有关使用不同的 AWS SDK 配置日志记录的信息。

Note

尽管您可以对每个请求和响应启用详细的日志记录，但我们不建议在生产系统中启用日志记录，因为大型请求或响应会显著降低应用程序的速度。

对于 AWS SDK 请求，请求 ID 对将如下所示：

```
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++

日志记录程序的类型和详细程度是在 SDK 初始化期间在 `SDKOptions` 参数中指定的。以下示例将详细程度级别指定为 `LogLevel::Debug`。

默认日志记录程序将写入文件系统，并使用以下约定 `aws_sdk_YYYY-MM-DD-HH.log` 对文件命名。日志记录程序会在整点创建一个新文件。

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
```

```
Aws::InitAPI(options);  
// ...  
Aws::ShutdownAPI(options);
```

有关更多信息，请参阅 GitHub 上适用于 C++ 的 AWS SDK wiki 中的 [How do I turn on logging?](#)

Go

可以使用适用于 Go 的 SDK 配置日志记录。有关更多信息，请参阅《适用于 Go 的 AWS SDK v2 开发人员指南》中的 [日志记录](#)。

Java

可以为特定的请求或响应启用日志记录，以仅捕获和返回相关的标头。为此，请导入 `com.amazonaws.services.s3.S3ResponseMetadata` 类。稍后，您可以先将请求存储在变量中，然后执行实际请求。要获取记录的请求或响应，请调用 `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`。

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

或者，您可以使用每个 Java 请求和响应的详细日志记录。有关更多信息，请参阅《适用于 Java 的 AWS SDK 开发人员指南》中的 [详细线路日志记录](#)。

JavaScript

适用于 JavaScript 的 AWS SDK 具有内置的日志记录程序，因此您可以记录使用它发出的 API 调用。要打开日志记录程序并在控制台中输出日志条目，请将以下语句添加到代码中：

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

有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》中的 [记录适用于 JavaScript 的 AWS SDK 调用](#)。

Kotlin

借助适用于 Kotlin 的 AWS SDK，您可以使用代码或环境设置为线路级消息指定日志模式。您可以为 HTTP 请求和 HTTP 响应设置日志模式。

要选择使用其它日志记录，请在构造服务客户端时设置 `logMode` 属性：

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

// ...

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

或者，您可以使用环境变量设置日志模式：

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

有关更多信息，请参阅《适用于 Kotlin 的 AWS SDK 开发人员指南》中的 [日志记录](#)。

.NET

可以使用内置的 适用于 .NET 的 SDK 日志记录工具通过 `System.Diagnostics` 配置日志记录。有关更多信息，请参阅 AWS 开发人员博客文章 [Logging with the 适用于 .NET 的 SDK](#)。

Note

默认情况下，返回的日志仅包含错误信息。要获取请求 ID，配置文件必须添加了 `AWSLogMetrics`（可以选择添加 `AWSResponseLogging`）。

PHP

通过在客户端构造函数中将调试选项设置为 `true`，可以获取调试信息，包括通过线路发送的数据。

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

有关更多信息，请参阅《适用于 PHP 的 AWS SDK 开发人员指南》中的 [如何查看在网上发送了什么数据？](#)

Python (Boto3)

使用适用于 Python (Boto3) 的 AWS SDK，您可以记录特定的响应。您可以使用此功能仅捕获相关的标题。以下代码显示如何将响应的各个部分记录到文件中：

```
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'])
```

还可以在引发异常时捕获异常并记录相关信息。有关更多信息，请参阅《适用于 Python 的 AWS SDK (Boto3) API 参考》中的[从错误响应中识别有用信息](#)。

此外，您可以使用以下代码将 Boto3 配置为输出详细调试日志：

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

有关更多信息，请参阅《适用于 Python 的 AWS SDK (Boto3) API 参考》中的[set_stream_logger](#)。

Ruby

您可以使用适用于 Ruby 的 SDK 版本 3 来获取请求 ID。您可以使用以下代码在客户端中启用 HTTP 线路日志记录：

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

Note

线路日志记录可能包含敏感信息，例如您的访问密钥 ID。在与 AWS Support 共享敏感信息之前，应对其进行清理。

您还可以在请求响应或错误中找到请求上下文对象的请求 ID：

```
# 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]
```

有关更多信息，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》中的[使用来自适用于 Ruby 的 AWS SDK 客户端的线路跟踪信息进行调试](#)。

Rust

要启用日志记录，请添加 tracing-subscriber crate 并在 Rust 应用程序中对其进行初始化。

将跟踪库添加到您的 Cargo.toml 文件中：

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

然后，在 Rust 代码中，在调用任何 SDK 操作之前，在 main 函数中初始化日志记录程序：

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

有关更多信息，请参阅《适用于 Rust 的 AWS SDK 开发人员指南》中的[在适用于 Rust 的 AWS SDK 中配置和使用日志记录](#)。

S3 表类数据存储服务支持的 Amazon S3 对象级 API 操作

下表包括 S3 表类数据存储服务支持的 S3 对象级 API 操作和相应标头。有关 Amazon S3 表类数据存储服务 API 的列表，请参阅 [Amazon S3 表类数据存储服务](#)。有关 Amazon S3 表类数据存储服务的更多信息，请参阅《Amazon S3 用户指南》中的[使用 Amazon S3 表类数据存储服务和表存储桶](#)。

对象级 API	支持的标头	备注	
AbortMultipartUpload	x-amz-expected-bucket-owner	无	
CompleteMultipartUpload	x-amz-checksum-crc32	无	
	x-amz-checksum-crc32c	无	
	x-amz-checksum-sha1	无	
	x-amz-checksum-sha256	无	
	x-amz-expected-bucket-owner	无	
	If-Match	无	
	If-None-Match	无	
CreateMultipartUpload	x-amz-acl: ACL	对于 S3 表类数据存储服务，默认值为 bucket-owner-full-control，且无法更改。	
	Cache-Control	无	
	Content-Disposition	无	
	Content-Encoding	无	
	Content-Language	无	

对象级 API	支持的标头	备注	
	Content-Type	无	
	Expires	无	
	x-amz-checksum-algorithm	无	
	x-amz-storage-class	对于 S3 表类数据存储服务，默认值为 STANDARD，且无法更改。	
	x-amz-server-side-encryption	对于 S3 表类数据存储服务，默认值为 (SSE-S3) (AES256)，且无法更改。	
GetObject	If-Match	无	
	If-Modified-Since	无	
	If-None-Match	无	
	If-Unmodified-Since	无	
	Range	无	
	x-amz-expected-bucket-owner	无	
	x-amz-checksum-mode	无	
HeadObject	If-Match	无	
	If-Modified-Since	无	

对象级 API	支持的标头	备注	
	If-None-Match	无	
	If-Unmodified-Since	无	
	Range	无	
	x-amz-checksum-mode	无	
	x-amz-expected-bucket-owner	无	
ListParts	x-amz-expected-bucket-owner	无	
PutObject	x-amz-acl: ACL	对于 S3 表类数据存储服务，默认值为 bucket-owner-full-control，且无法更改。	
	Cache-Control	无	
	Content-Disposition	无	
	Content-Encoding	无	
	Content-Language	无	
	Content-Length	无	
	Content-MD5	无	
	Content-Type	无	

对象级 API	支持的标头	备注	
	x-amz-sdk-checksum-algorithm	无	
	x-amz-checksum-crc32	无	
	x-amz-checksum-crc32c	无	
	x-amz-checksum-sha1	无	
	x-amz-checksum-sha256	无	
	Expires	无	
	If-None-Match	无	
	x-amz-expected-bucket-owner	无	
	x-amz-storage-class	对于 S3 表类数据存储服务，默认值为 STANDARD，且无法更改。	
	x-amz-server-side-encryption	对于 S3 表类数据存储服务，默认值为 (SSE-S3) (AES256)，且无法更改。	
UploadPart	Content-Length	无	
	Content-MD5	无	
	x-amz-checksum-crc32	无	

对象级 API	支持的标头	备注	
	x-amz-checksum-crc32c	无	
	x-amz-checksum-sha1	无	
	x-amz-checksum-sha256	无	
	x-amz-expected-bucket-owner	无	

适用于使用 AWS 软件开发工具包的 Amazon S3 的代码示例

以下代码示例演示了如何将 Amazon S3 与 AWS 软件开发工具包 (SDK) 一起使用。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- [使用 AWS SDK 的 Amazon S3 代码示例](#)
 - [使用 AWS SDK 的 Amazon S3 的基本功能示例](#)
 - [开始使用 Amazon S3](#)
 - [了解使用 AWS SDK 的 Amazon S3 的基本功能](#)
 - [使用 AWS SDK 对 Amazon S3 执行的操作](#)
 - [将 AbortMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CompleteMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CopyObject 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreateBucket 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreateMultiRegionAccessPoint 和 AWS SDK 搭配使用](#)
 - [将 CreateMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreatePresignedPost 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucket 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketAnalyticsConfiguration 与 CLI 配合使用](#)
 - [将 DeleteBucketCors 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketEncryption 与 CLI 配合使用](#)
 - [将 DeleteBucketInventoryConfiguration 与 CLI 配合使用](#)
 - [将 DeleteBucketLifecycle 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketMetricsConfiguration 与 CLI 配合使用](#)
 - [将 DeleteBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketReplication 与 CLI 配合使用](#)
 - [将 DeleteBucketTagging 与 CLI 配合使用](#)
 - [将 DeleteBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteObject 与 AWS SDK 或 CLI 配合使用](#)

- [将 DeleteObjectTagging 与 CLI 配合使用](#)
- [将 DeleteObjects 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeletePublicAccessBlock 与 CLI 配合使用](#)
- [将 GetBucketAccelerateConfiguration 与 CLI 配合使用](#)
- [将 GetBucketAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketAnalyticsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketEncryption 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketInventoryConfiguration 与 CLI 配合使用](#)
- [将 GetBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLocation 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLogging 与 CLI 配合使用](#)
- [将 GetBucketMetricsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketNotification 与 CLI 配合使用](#)
- [将 GetBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketPolicyStatus 与 CLI 配合使用](#)
- [将 GetBucketReplication 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketRequestPayment 与 CLI 配合使用](#)
- [将 GetBucketTagging 与 CLI 配合使用](#)
- [将 GetBucketVersioning 与 CLI 配合使用](#)
- [将 GetBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAttributes 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLegalHold 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectRetention 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectTagging 与 CLI 配合使用](#)
- [将 GetPublicAccessBlock 与 CLI 配合使用](#)
- [将 HeadBucket 与 AWS SDK 或 CLI 配合使用](#)

- [将 HeadObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListBucketAnalyticsConfigurations 与 CLI 配合使用](#)
- [将 ListBucketInventoryConfigurations 与 CLI 配合使用](#)
- [将 ListBuckets 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListMultipartUploads 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListObjectVersions 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListObjects 与 CLI 配合使用](#)
- [将 ListObjectsV2 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketEncryption 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketLogging 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketNotification 与 CLI 配合使用](#)
- [将 PutBucketNotificationConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketReplication 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketRequestPayment 与 CLI 配合使用](#)
- [将 PutBucketTagging 与 CLI 配合使用](#)
- [将 PutBucketVersioning 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectLegalHold 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用](#)
- [将 RestoreObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 SelectObjectContent 与 AWS SDK 或 CLI 配合使用](#)
- [将 UploadPart 与 AWS SDK 或 CLI 配合使用](#)

- [将 UploadPartCopy 与 AWS SDK 或 CLI 配合使用](#)
- [适用于使用 AWS 软件开发工具包的 Amazon S3 的场景](#)
 - [检查存储桶是否存在](#)
 - [检查对象是否存在](#)
 - [使用 AWS SDK 将文本转换为语音以及将语音转换回文本](#)
 - [使用 AWS SDK 创建 Amazon S3 的预签名 URL](#)
 - [创建照片资产管理应用程序，让用户能够使用标签管理照片](#)
 - [使用 AWS SDK 列出 Amazon S3 对象的网页](#)
 - [创建 Amazon Textract 浏览器应用程序](#)
 - [使用 AWS SDK 删除给定 Amazon S3 存储桶中的所有对象](#)
 - [使用 AWS SDK 删除未完成的 Amazon S3 分段上传](#)
 - [使用 AWS SDK 通过 Amazon Rekognition 检测图像中的 PPE](#)
 - [使用 AWS SDK 检测从图像中提取的文本中的实体](#)
 - [使用 AWS SDK 检测图像中的人脸](#)
 - [使用 AWS 软件开发工具包通过 Amazon Rekognition 检测图像中的对象](#)
 - [使用 AWS SDK 通过 Amazon Rekognition 检测视频中的人物和对象](#)
 - [从 Amazon Simple Storage Service \(Amazon S3\) 存储桶下载 S3“目录”](#)
 - [将 Amazon Simple Storage Service \(Amazon S3 \) 桶中的所有对象下载到本地目录](#)
 - [使用 AWS SDK 从 Amazon S3 对象下载未知大小的流](#)
 - [使用 AWS SDK 从多区域接入点中获取 Amazon S3 对象](#)
 - [使用 AWS SDK 并指定 If-Modified-Since 标头以从 Amazon S3 存储桶获取对象](#)
 - [使用 AWS SDK 加密 Amazon S3 对象入门](#)
 - [使用 AWS SDK 为 Amazon S3 存储桶和对象加标签入门](#)
 - [大数据处理集群入门](#)
 - [配置管理入门](#)
 - [文档文本提取入门](#)
 - [机器学习特征存放区入门](#)
 - [对象存储入门](#)
 - [查询分析入门](#)
- [通过 AWS SDK 使用 Amazon S3 对象锁定功能](#)

- [使用 AWS SDK 发起 Amazon S3 条件请求](#)
- [使用 AWS SDK 管理 Amazon S3 存储桶的访问控制列表 \(ACL \)](#)
- [使用 AWS SDK 的 Amazon S3 管理大型 Amazon SQS 消息](#)
- [通过 AWS SDK 使用 Lambda 函数批量管理版本控制的 Amazon S3 对象](#)
- [使用 AWS SDK 解析 Amazon S3 URI](#)
- [使用 AWS SDK 执行 Amazon S3 对象的分段复制](#)
- [使用 AWS SDK 接收和处理 Amazon S3 事件通知](#)
- [使用 AWS SDK 保存 EXIF 和其他图像信息](#)
- [使用 AWS SDK 向 Amazon EventBridge 发送 S3 事件通知](#)
- [跟踪使用 AWS SDK 执行的 Amazon S3 对象上传或下载操作](#)
- [使用 S3 对象 Lambda 转换应用程序的数据](#)
- [使用 AWS SDK 进行单元测试和集成测试的示例方法](#)
- [以递归方式将本地目录上传到 Amazon Simple Storage Service \(Amazon S3 \) 桶](#)
- [借助 AWS SDK 向 Amazon S3 上传大文件或从 Amazon S3 下载大文件](#)
- [使用 AWS SDK 将未知大小的流上传到 Amazon S3 对象](#)
- [使用 AWS 开发工具包，对 Amazon S3 对象使用校验和](#)
- [通过 AWS SDK 使用 Amazon S3 对象完整性功能](#)
- [使用 AWS SDK 使用 Amazon S3 版本控制对象](#)
- [Amazon S3 无服务器示例](#)
 - [通过 Amazon S3 触发器调用 Lambda 函数](#)
- [使用 AWS SDK 的 Amazon S3 Control 代码示例](#)
- [使用 AWS SDK 的 Amazon S3 Control 的基本功能示例](#)
 - [Hello Amazon S3 Control](#)
 - [了解使用 AWS SDK 的 Amazon S3 Control 的基本功能](#)
 - [使用 AWS SDK 对 Amazon S3 控件执行的操作](#)
 - [将 CreateJob 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteJobTagging 和 AWS SDK 搭配使用](#)
 - [将 DescribeJob 与 AWS SDK 或 CLI 配合使用](#)
 - [将 GetJobTagging 和 AWS SDK 搭配使用](#)
 - [将 PutJobTagging 和 AWS SDK 搭配使用](#)

- [将 UpdateJobPriority 与 AWS SDK 或 CLI 配合使用](#)
- [将 UpdateJobStatus 与 AWS SDK 或 CLI 配合使用](#)
- [使用 AWS SDK 的 S3 目录存储桶代码示例](#)
 - [使用 AWS SDK 的 S3 目录存储桶基本功能示例](#)
 - [Hello Amazon S3 目录存储桶](#)
 - [通过 AWS SDK 了解 S3 目录存储桶基础功能](#)
 - [适用于使用 AWS SDK 的 S3 目录存储桶的操作](#)
 - [将 AbortMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CompleteMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CopyObject 和 AWS SDK 搭配使用](#)
 - [将 CreateBucket 和 AWS SDK 搭配使用](#)
 - [将 CreateMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CreateSession 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucket 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucketEncryption 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucketPolicy 和 AWS SDK 搭配使用](#)
 - [将 DeleteObject 和 AWS SDK 搭配使用](#)
 - [将 DeleteObjects 和 AWS SDK 搭配使用](#)
 - [将 GetBucketEncryption 和 AWS SDK 搭配使用](#)
 - [将 GetBucketPolicy 和 AWS SDK 搭配使用](#)
 - [将 GetObject 和 AWS SDK 搭配使用](#)
 - [将 GetObjectAttributes 和 AWS SDK 搭配使用](#)
 - [将 HeadBucket 和 AWS SDK 搭配使用](#)
 - [将 HeadObject 和 AWS SDK 搭配使用](#)
 - [将 ListDirectoryBuckets 和 AWS SDK 搭配使用](#)
 - [将 ListMultipartUploads 和 AWS SDK 搭配使用](#)
 - [将 ListObjectsV2 和 AWS SDK 搭配使用](#)
 - [将 ListParts 和 AWS SDK 搭配使用](#)
 - [将 PutBucketEncryption 和 AWS SDK 搭配使用](#)
 - [将 PutBucketPolicy 和 AWS SDK 搭配使用](#)

- [将 PutObject 和 AWS SDK 搭配使用](#)
- [将 UploadPart 和 AWS SDK 搭配使用](#)
- [将 UploadPartCopy 和 AWS SDK 搭配使用](#)
- [使用 AWS SDK 的 S3 目录存储桶场景](#)
 - [使用 AWS SDK 为 Amazon S3 目录存储桶创建获取对象的预签名 URL](#)

使用 AWS SDK 的 Amazon S3 代码示例

以下代码示例演示了如何将 Amazon S3 与 AWS 软件开发工具包 (SDK) 一起使用。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务 服务结合来完成特定任务的代码示例。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon S3

- [使用 AWS SDK 的 Amazon S3 的基本功能示例](#)
 - [开始使用 Amazon S3](#)
 - [了解使用 AWS SDK 的 Amazon S3 的基本功能](#)
 - [使用 AWS SDK 对 Amazon S3 执行的操作](#)
 - [将 AbortMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CompleteMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CopyObject 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreateBucket 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreateMultiRegionAccessPoint 和 AWS SDK 搭配使用](#)
 - [将 CreateMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreatePresignedPost 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucket 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketAnalyticsConfiguration 与 CLI 配合使用](#)

- [将 DeleteBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteBucketEncryption 与 CLI 配合使用](#)
- [将 DeleteBucketInventoryConfiguration 与 CLI 配合使用](#)
- [将 DeleteBucketLifecycle 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteBucketMetricsConfiguration 与 CLI 配合使用](#)
- [将 DeleteBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteBucketReplication 与 CLI 配合使用](#)
- [将 DeleteBucketTagging 与 CLI 配合使用](#)
- [将 DeleteBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteObjectTagging 与 CLI 配合使用](#)
- [将 DeleteObjects 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeletePublicAccessBlock 与 CLI 配合使用](#)
- [将 GetBucketAccelerateConfiguration 与 CLI 配合使用](#)
- [将 GetBucketAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketAnalyticsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketEncryption 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketInventoryConfiguration 与 CLI 配合使用](#)
- [将 GetBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLocation 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLogging 与 CLI 配合使用](#)
- [将 GetBucketMetricsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketNotification 与 CLI 配合使用](#)
- [将 GetBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketPolicyStatus 与 CLI 配合使用](#)
- [将 GetBucketReplication 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketRequestPayment 与 CLI 配合使用](#)
- [将 GetBucketTagging 与 CLI 配合使用](#)
- [将 GetBucketVersioning 与 CLI 配合使用](#)

- [将 GetBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAttributes 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLegalHold 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectRetention 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectTagging 与 CLI 配合使用](#)
- [将 GetPublicAccessBlock 与 CLI 配合使用](#)
- [将 HeadBucket 与 AWS SDK 或 CLI 配合使用](#)
- [将 HeadObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListBucketAnalyticsConfigurations 与 CLI 配合使用](#)
- [将 ListBucketInventoryConfigurations 与 CLI 配合使用](#)
- [将 ListBuckets 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListMultipartUploads 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListObjectVersions 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListObjects 与 CLI 配合使用](#)
- [将 ListObjectsV2 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketEncryption 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketLogging 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketNotification 与 CLI 配合使用](#)
- [将 PutBucketNotificationConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketReplication 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketRequestPayment 与 CLI 配合使用](#)
- [将 PutBucketTagging 与 CLI 配合使用](#)

- [将 PutBucketVersioning 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectLegalHold 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用](#)
- [将 RestoreObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 SelectObjectContent 与 AWS SDK 或 CLI 配合使用](#)
- [将 UploadPart 与 AWS SDK 或 CLI 配合使用](#)
- [将 UploadPartCopy 与 AWS SDK 或 CLI 配合使用](#)
- [适用于使用 AWS 软件开发工具包的 Amazon S3 的场景](#)
 - [检查存储桶是否存在](#)
 - [检查对象是否存在](#)
 - [使用 AWS SDK 将文本转换为语音以及将语音转换回文本](#)
 - [使用 AWS SDK 创建 Amazon S3 的预签名 URL](#)
 - [创建照片资产管理应用程序，让用户能够使用标签管理照片](#)
 - [使用 AWS SDK 列出 Amazon S3 对象的网页](#)
 - [创建 Amazon Textract 浏览器应用程序](#)
 - [使用 AWS SDK 删除给定 Amazon S3 存储桶中的所有对象](#)
 - [使用 AWS SDK 删除未完成的 Amazon S3 分段上传](#)
 - [使用 AWS SDK 通过 Amazon Rekognition 检测图像中的 PPE](#)
 - [使用 AWS SDK 检测从图像中提取的文本中的实体](#)
 - [使用 AWS SDK 检测图像中的人脸](#)
 - [使用 AWS 软件开发工具包通过 Amazon Rekognition 检测图像中的对象](#)
 - [使用 AWS SDK 通过 Amazon Rekognition 检测视频中的人物和对象](#)
 - [从 Amazon Simple Storage Service \(Amazon S3\) 存储桶下载 S3“目录”](#)
 - [将 Amazon Simple Storage Service \(Amazon S3 \) 桶中的所有对象下载到本地目录](#)
 - [使用 AWS SDK 从 Amazon S3 对象下载未知大小的流](#)
- [使用 AWS SDK 从多区域接入点中获取 Amazon S3 对象](#)

- [使用 AWS SDK 并指定 If-Modified-Since 标头以从 Amazon S3 存储桶获取对象](#)
 - [使用 AWS SDK 加密 Amazon S3 对象入门](#)
 - [使用 AWS SDK 为 Amazon S3 存储桶和对象加标签入门](#)
 - [大数据处理集群入门](#)
 - [配置管理入门](#)
 - [文档文本提取入门](#)
 - [机器学习特征存放区入门](#)
 - [对象存储入门](#)
 - [查询分析入门](#)
 - [通过 AWS SDK 使用 Amazon S3 对象锁定功能](#)
 - [使用 AWS SDK 发起 Amazon S3 条件请求](#)
 - [使用 AWS SDK 管理 Amazon S3 存储桶的访问控制列表 \(ACL \)](#)
 - [使用 AWS SDK 的 Amazon S3 管理大型 Amazon SQS 消息](#)
 - [通过 AWS SDK 使用 Lambda 函数批量管理版本控制的 Amazon S3 对象](#)
 - [使用 AWS SDK 解析 Amazon S3 URI](#)
 - [使用 AWS SDK 执行 Amazon S3 对象的分段复制](#)
 - [使用 AWS SDK 接收和处理 Amazon S3 事件通知](#)
 - [使用 AWS SDK 保存 EXIF 和其他图像信息](#)
 - [使用 AWS SDK 向 Amazon EventBridge 发送 S3 事件通知](#)
 - [跟踪使用 AWS SDK 执行的 Amazon S3 对象上传或下载操作](#)
 - [使用 S3 对象 Lambda 转换应用程序的数据](#)
 - [使用 AWS SDK 进行单元测试和集成测试的示例方法](#)
 - [以递归方式将本地目录上传到 Amazon Simple Storage Service \(Amazon S3 \) 桶](#)
 - [借助 AWS SDK 向 Amazon S3 上传大文件或从 Amazon S3 下载大文件](#)
 - [使用 AWS SDK 将未知大小的流上传到 Amazon S3 对象](#)
 - [使用 AWS 开发工具包，对 Amazon S3 对象使用校验和](#)
 - [通过 AWS SDK 使用 Amazon S3 对象完整性功能](#)
 - [使用 AWS SDK 使用 Amazon S3 版本控制对象](#)
 - [Amazon S3 无服务器示例](#)
- [通过 Amazon S3 触发器调用 Lambda 函数](#)

使用 AWS SDK 的 Amazon S3 的基本功能示例

以下代码示例展示了如何将 Amazon Simple Storage Service 的基本功能与 AWS SDK 结合使用。

示例

- [开始使用 Amazon S3](#)
- [了解使用 AWS SDK 的 Amazon S3 的基本功能](#)
- [使用 AWS SDK 对 Amazon S3 执行的操作](#)
 - [将 AbortMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CompleteMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CopyObject 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreateBucket 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreateMultiRegionAccessPoint 和 AWS SDK 搭配使用](#)
 - [将 CreateMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
 - [将 CreatePresignedPost 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucket 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketAnalyticsConfiguration 与 CLI 配合使用](#)
 - [将 DeleteBucketCors 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketEncryption 与 CLI 配合使用](#)
 - [将 DeleteBucketInventoryConfiguration 与 CLI 配合使用](#)
 - [将 DeleteBucketLifecycle 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketMetricsConfiguration 与 CLI 配合使用](#)
 - [将 DeleteBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteBucketReplication 与 CLI 配合使用](#)
 - [将 DeleteBucketTagging 与 CLI 配合使用](#)
 - [将 DeleteBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteObject 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteObjectTagging 与 CLI 配合使用](#)
 - [将 DeleteObjects 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeletePublicAccessBlock 与 CLI 配合使用](#)
 - [将 GetBucketAccelerateConfiguration 与 CLI 配合使用](#)
 - [将 GetBucketAcl 与 AWS SDK 或 CLI 配合使用](#)

- [将 GetBucketAnalyticsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketEncryption 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketInventoryConfiguration 与 CLI 配合使用](#)
- [将 GetBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLocation 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLogging 与 CLI 配合使用](#)
- [将 GetBucketMetricsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketNotification 与 CLI 配合使用](#)
- [将 GetBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketPolicyStatus 与 CLI 配合使用](#)
- [将 GetBucketReplication 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketRequestPayment 与 CLI 配合使用](#)
- [将 GetBucketTagging 与 CLI 配合使用](#)
- [将 GetBucketVersioning 与 CLI 配合使用](#)
- [将 GetBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAttributes 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLegalHold 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectRetention 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectTagging 与 CLI 配合使用](#)
- [将 GetPublicAccessBlock 与 CLI 配合使用](#)
- [将 HeadBucket 与 AWS SDK 或 CLI 配合使用](#)
- [将 HeadObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListBucketAnalyticsConfigurations 与 CLI 配合使用](#)
- [将 ListBucketInventoryConfigurations 与 CLI 配合使用](#)
- [将 ListBuckets 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListMultipartUploads 与 AWS SDK 或 CLI 配合使用](#)

- [将 ListObjectVersions 与 AWS SDK 或 CLI 配合使用](#)
- [将 ListObjects 与 CLI 配合使用](#)
- [将 ListObjectsV2 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketEncryption 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketLogging 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketNotification 与 CLI 配合使用](#)
- [将 PutBucketNotificationConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketReplication 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketRequestPayment 与 CLI 配合使用](#)
- [将 PutBucketTagging 与 CLI 配合使用](#)
- [将 PutBucketVersioning 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectLegalHold 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用](#)
- [将 RestoreObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 SelectObjectContent 与 AWS SDK 或 CLI 配合使用](#)
- [将 UploadPart 与 AWS SDK 或 CLI 配合使用](#)
- [将 UploadPartCopy 与 AWS SDK 或 CLI 配合使用](#)

开始使用 Amazon S3

以下代码示例展示了如何开始使用 Amazon S3。

基本功能

.NET

适用于 .NET 的 SDK (v4)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [ListBuckets](#)。

C++

SDK for C++

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# 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})

```

hello_s3.cpp 源文件的代码。

```

#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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [ListBuckets](#)。

Go

适用于 Go 的 SDK V2

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```



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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListBuckets](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;  
    }  
}  
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [ListBuckets](#)。

PHP

适用于 PHP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 PHP 的 AWS SDK API Reference》中的 [ListBuckets](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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()
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [ListBuckets](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# frozen_string_literal: true

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

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

    response = @client.list_buckets

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

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



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

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [ListBuckets](#)。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/// 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 实用程序。

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

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [ListBuckets](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

了解使用 AWS SDK 的 Amazon S3 的基本功能

以下代码示例演示了如何：

- 创建存储桶并将文件上传到其中。
- 从桶中下载对象。
- 将对象复制到存储桶中的子文件夹。
- 列出存储桶中的对象。
- 删除桶对象和桶。

.NET

适用于 .NET 的 SDK (v4)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行一个交互式场景，演示 Amazon S3 的功能。

```
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.Write("Please enter the path and filename of the file
to upload: ");
        filePath = Console.ReadLine();

        // Confirm that the file exists on the local computer.
        if (!File.Exists(filePath))
        {
            Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
            filePath = string.Empty;
        }
    }
}
else
{
    // Use the public variable if set, otherwise create a temp file
    if (!string.IsNullOrEmpty(TempFilePath))
    {
        filePath = TempFilePath;
        Console.WriteLine($"Using provided test file: {filePath}");
    }
    else
    {
        // Create a temporary test file for non-interactive mode
        filePath = Path.GetTempFileName();
        var testContent = "This is a test file for S3 basics
scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
UTC");

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

```

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

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

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

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

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

        downloadPath = Console.ReadLine();

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

        if (File.Exists($"{downloadPath}\\{keyName}"))
        {
            Console.WriteLine($"Sorry, the file already exists in
that location.\n");

            downloadPath = string.Empty;
        }
    }
}
else
{
    downloadPath = Path.GetTempPath();
    var downloadFile = Path.Combine(downloadPath, keyName);
    if (File.Exists(downloadFile))
```



```
        {
            File.Delete(downloadFile);
        }

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

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

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

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

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

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

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

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

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

if (IsInteractive)
{
    // Deleting the bucket too quickly after separately deleting its
contents can
    // cause an error that the bucket isn't empty. To delete contents
and bucket in one
    // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
    Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
    _ = Console.ReadLine();
}
else
{
    // Add a small delay for non-interactive mode to ensure objects
are fully deleted.
    Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
    await Task.Delay(2000);
}

// Delete the bucket.
var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
if (deleteSuccess)
```

```
        {
            Console.WriteLine($"Successfully deleted {bucketName}.\n");
        }
        else
        {
            Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
        }

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

                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                    Console.WriteLine("Cleaned up downloaded test file.");
                }
            }
            catch (Exception ex)
            {
                scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

            // Clean up on error - delete bucket if it exists
            try
            {
                if (!string.IsNullOrEmpty(bucketName))
                {
                    await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                }
            }
            catch { }
        }
    }
}
```

```

        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;
}
}
}

```

Amazon S3 SDK 方法的包装器类。

```

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;
    }
}
}

```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的以下主题。

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

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# 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
}

```

```
    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
}

```

此场景中使用的 Amazon S3 函数。

```

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
#     And:

```

```

#      0 - If successful.
#      1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

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

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

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

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

```

```

    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

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

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

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

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

#####
# 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
}

```

```

    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
}

```

```
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

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

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#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();
}

```

- 有关 API 详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考中的以下主题。
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

定义一个结构来包装此场景使用的桶和对象操作。

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

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

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

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
```

```

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

  return exists, err
}

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

```

```

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

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

```

```
}
return err
}

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

    return err
}
```



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

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

```

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

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

```

```

    }
    }
    return err
}

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

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

```

```

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

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

```

```

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

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

```

```
}
```

运行一个交互式场景，演示如何使用 S3 存储桶和对象。

```
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() {
```



```
    panic(err)
} else {
    log.Println("Bucket created.")
}
}
log.Println(strings.Repeat("-", 88))

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

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

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

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
```



```
var objKeys []string
for _, object := range objects {
    objKeys = append(objKeys, *object.Key)
    log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

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

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

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)

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

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

场景示例。

```
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.");
            }
        }
    }
}

```

包含操作的包装器类。

```

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;
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

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

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

首先，导入所有必需的模块。

```
// 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";

```

前面的导入引用了一些帮助程序实用程序。这些实用程序是在本节开头链接的 GitHub 存储库的本地实用程序。为了便于参考，请参阅这些实用程序的以下实现。

```

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`;
};

```

对象存储在“存储桶”中。让我们定义一个用于创建新存储桶的函数。

```

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;
};

```

存储桶包含“对象”。此函数将目录的内容作为对象上传到您的存储桶。

```

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.`);
}
};
```

上传对象后，检查以确认它们已正确上传。可以使用 `ListObjects` 完成此操作。您将使用“Key”属性，但响应中还有其他有用的属性。

```
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`);
};
```

有时，您可能想要将对象从一个桶复制到另一个桶。为此，请使用 `CopyObject` 命令。

```
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();
};

```

没有用于从桶中获取多个对象的 SDK 方法。相反，您将创建一个要下载的对象列表并对其进行迭代。

```

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");
};
```

是时候清除资源了。桶必须为空才能删除它。这两个函数清空并删除桶。

```
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`);
};
```

“main”函数将所有内容整合在一起。如果您直接运行此文件，将调用 main 函数。

```
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);
}
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的以下主题。
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)

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

Kotlin

SDK for Kotlin

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

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

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

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

```
val toBucket = args[4]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

        val response = s3.listObjects(request)
```

```

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

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

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

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

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

```

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

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

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

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的以下主题。

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

PHP

适用于 PHP 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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";
```

- 有关 API 详细信息，请参阅 适用于 PHP 的 AWS SDK API 参考中的以下主题。

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

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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"))
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。

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

Ruby

适用于 Ruby 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
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

```



```
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

# 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__
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考中的以下主题。

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

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

用于运行场景的二进制 crate 的代码。

```
#![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(())
    }
}

```

场景使用的常用操作。

```

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))
            }
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的以下主题。

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

SAP ABAP

适用于 SAP ABAP 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.

```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的 AWS SDK 的 API 参考中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

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

Swift

适用于 Swift 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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)
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

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

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 对 Amazon S3 执行的操作

以下代码示例演示了如何使用 AWS SDK 来执行各个 Amazon S3 操作。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon S3 API，是必须在上下文中运行的较大型程序的代码节选。您可以在[适用于使用 AWS 软件开发工具包的 Amazon S3 的场景](#)中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 [Amazon Simple Storage Service API 参考](#)。

示例

- [将 AbortMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
- [将 CompleteMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
- [将 CopyObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 CreateBucket 与 AWS SDK 或 CLI 配合使用](#)
- [将 CreateMultiRegionAccessPoint 和 AWS SDK 搭配使用](#)
- [将 CreateMultipartUpload 与 AWS SDK 或 CLI 配合使用](#)
- [将 CreatePresignedPost 和 AWS SDK 搭配使用](#)
- [将 DeleteBucket 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteBucketAnalyticsConfiguration 与 CLI 配合使用](#)
- [将 DeleteBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteBucketEncryption 与 CLI 配合使用](#)
- [将 DeleteBucketInventoryConfiguration 与 CLI 配合使用](#)
- [将 DeleteBucketLifecycle 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteBucketMetricsConfiguration 与 CLI 配合使用](#)
- [将 DeleteBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteBucketReplication 与 CLI 配合使用](#)

- [将 DeleteBucketTagging 与 CLI 配合使用](#)
- [将 DeleteBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteObjectTagging 与 CLI 配合使用](#)
- [将 DeleteObjects 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeletePublicAccessBlock 与 CLI 配合使用](#)
- [将 GetBucketAccelerateConfiguration 与 CLI 配合使用](#)
- [将 GetBucketAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketAnalyticsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketCors 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketEncryption 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketInventoryConfiguration 与 CLI 配合使用](#)
- [将 GetBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLocation 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketLogging 与 CLI 配合使用](#)
- [将 GetBucketMetricsConfiguration 与 CLI 配合使用](#)
- [将 GetBucketNotification 与 CLI 配合使用](#)
- [将 GetBucketPolicy 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketPolicyStatus 与 CLI 配合使用](#)
- [将 GetBucketReplication 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetBucketRequestPayment 与 CLI 配合使用](#)
- [将 GetBucketTagging 与 CLI 配合使用](#)
- [将 GetBucketVersioning 与 CLI 配合使用](#)
- [将 GetBucketWebsite 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAcl 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectAttributes 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLegalHold 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用](#)

- [将 `GetObjectRetention` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `GetObjectTagging` 与 CLI 配合使用](#)
- [将 `GetPublicAccessBlock` 与 CLI 配合使用](#)
- [将 `HeadBucket` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `HeadObject` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `ListBucketAnalyticsConfigurations` 与 CLI 配合使用](#)
- [将 `ListBucketInventoryConfigurations` 与 CLI 配合使用](#)
- [将 `ListBuckets` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `ListMultipartUploads` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `ListObjectVersions` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `ListObjects` 与 CLI 配合使用](#)
- [将 `ListObjectsV2` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketAccelerateConfiguration` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketAcl` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketCors` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketEncryption` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketLifecycleConfiguration` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketLogging` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketNotification` 与 CLI 配合使用](#)
- [将 `PutBucketNotificationConfiguration` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketPolicy` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketReplication` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketRequestPayment` 与 CLI 配合使用](#)
- [将 `PutBucketTagging` 与 CLI 配合使用](#)
- [将 `PutBucketVersioning` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutBucketWebsite` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutObject` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutObjectAcl` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutObjectLegalHold` 与 AWS SDK 或 CLI 配合使用](#)
- [将 `PutObjectLockConfiguration` 与 AWS SDK 或 CLI 配合使用](#)

- [将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用](#)
- [将 RestoreObject 与 AWS SDK 或 CLI 配合使用](#)
- [将 SelectObjectContent 与 AWS SDK 或 CLI 配合使用](#)
- [将 UploadPart 与 AWS SDK 或 CLI 配合使用](#)
- [将 UploadPartCopy 与 AWS SDK 或 CLI 配合使用](#)

将 **AbortMultipartUpload** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 AbortMultipartUpload。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [删除未完成的分段上传](#)
- [使用 Amazon S3 对象完整性](#)

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
//! 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();
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [AbortMultipartUpload](#)。

CLI

AWS CLI

中止指定的分段上传

以下 `abort-multipart-upload` 命令中止存储桶 `amzn-s3-demo-bucket` 中键 `multipart/01` 的分段上传：

```

aws s3api abort-multipart-upload \
    --bucket amzn-s3-demo-bucket \
    --key multipart/01 \
    --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZlJF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P

```

此命令所需的上传 ID 由 `create-multipart-upload` 输出，也可以使用 `list-multipart-uploads` 进行检索。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [AbortMultipartUpload](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [AbortMultipartUpload](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令中止在 5 天前创建的分段上传。

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

示例 2：此命令中止在 2014 年 1 月 2 日之前创建的分段上传。

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

示例 3：此命令中止在 2014 年 1 月 2 日 10:45:37 之前创建的分段上传。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [AbortMultipartUpload](#)。

适用于 PowerShell V5 的工具

示例 1：此命令中止在 5 天前创建的分段上传。

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

示例 2：此命令中止在 2014 年 1 月 2 日之前创建的分段上传。

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

示例 3：此命令中止在 2014 年 1 月 2 日 10:45:37 之前创建的分段上传。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [AbortMultipartUpload](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CompleteMultipartUpload** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CompleteMultipartUpload。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [执行分段复制](#)
- [使用校验和](#)
- [使用 Amazon S3 对象完整性](#)

C++

SDK for C++

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#!/ 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;
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [CompleteMultipartUpload](#)。

CLI

AWS CLI

以下命令完成存储桶 `amzn-s3-demo-bucket` 中密钥 `multipart/01` 的分段上传：

```

aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WMCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P

```

此命令所需的上传 ID 由 `create-multipart-upload` 输出，也可以使用 `list-multipart-uploads` 进行检索。

上述命令中的分段上传选项采用 JSON 结构，用于描述分段上传中应重新组合成完整文件的各个部分。在此示例中，`file://` 前缀用于从名为 `mpustruct` 的本地文件夹中的文件加载 JSON 结构。

mpustruct :

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

每次使用 `upload-part` 命令上传分段时，都会输出每个分段的 ETag 值，也可以通过调用 `list-parts` 来检索，或者通过对每个分段进行 MD5 校验和执行计算。

输出：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CompleteMultipartUpload](#)。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 [CompleteMultipartUpload](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CopyObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CopyObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [加密入门](#)
- [对象存储入门](#)

- [提出条件请求](#)

.NET

适用于 .NET 的 SDK (v4)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/// <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;
    }
}

```

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [CopyObject](#)。

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用条件请求复制对象。

```

    /// <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;
    }
}

```

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [CopyObject](#)。

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CopyObject](#)。

C++

SDK for C++

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
    }
}

```

- 有关 API 详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考中的 [CopyObject](#)。

CLI

AWS CLI

以下命令将对象从 bucket-1 复制到 bucket-2：

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

输出：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CopyObject](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 AWS SDK API 参考中的 [CopyObject](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 [S3Client](#) 复制对象。

```

/**
 * 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);
}

```

使用 [S3TransferManager](#) 从一个桶将[对象复制](#)到另一个桶。查看[完整文件](#)并[进行测试](#)。

```

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();
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 [CopyObject](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

复制对象。

```
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;
    }
}
};

```

在对象的 ETag 与提供的 ETag 不匹配的条件下复制该对象。

```

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"));
  }
}

```

在对象的 ETag 与提供的 ETag 不匹配的条件下复制该对象。

```

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"));
    }
}
}

```

对象在给定时间范围内创建或修改的条件下复制该对象。

```

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"));
  }
}
```

对象未在给定时间范围内创建或修改的条件下复制该对象。

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

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [CopyObject](#)。

Kotlin

SDK for Kotlin

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

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

```
s3.copyObject(request)
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [CopyObject](#)。

PHP

适用于 PHP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

对象的简单副本。

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

- 有关 API 的详细信息，请参阅 适用于 PHP 的 AWS SDK API 参考中的 [CopyObject](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令将对象“sample.txt”从存储桶“test-files”复制到同一个存储桶，但新键为“sample-copy.txt”。

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

示例 2：此命令将对象“sample.txt”从存储桶“test-files”复制到存储桶“backup-files”，键为“sample-copy.txt”。

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

示例 3：此命令将对象“sample.txt”从存储桶“test-files”下载到名为“local-sample.txt”的本地文件。

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

示例 4：将单个对象下载到指定的文件。下载的文件可以在 c:\downloads\data\archive.zip 中找到

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

示例 5：将与指定的键前缀匹配的所有对象下载到本地文件夹。相对键层次结构将作为子文件夹保留在总体下载位置中。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [CopyObject](#)。

适用于 PowerShell V5 的工具

示例 1：此命令将对象“sample.txt”从存储桶“test-files”复制到同一个存储桶，但新键为“sample-copy.txt”。


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

示例 2：此命令将对象“sample.txt”从存储桶“test-files”复制到存储桶“backup-files”，键为“sample-copy.txt”。

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

示例 3：此命令将对象“sample.txt”从存储桶“test-files”下载到名为“local-sample.txt”的本地文件。

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

示例 4：将单个对象下载到指定的文件。下载的文件可以在 c:\downloads\data\archive.zip 中找到

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

示例 5：将与指定的键前缀匹配的所有对象下载到本地文件夹。相对键层次结构将作为子文件夹保留在总体下载位置中。

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

示例 6：使用分段并行下载将单个对象下载到本地文件，以提高处理大型文件的性能。

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

示例 7：使用 RANGE 模式以自定义 64 MB 分段大小下载大型对象。RANGE 模式使用 HTTP 字节范围请求并适用于所有对象，而无论它们是如何上传的。

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

示例 8：将分段下载与并发文件下载结合使用，以将所有与键前缀匹配的对象下载到本地文件夹。

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

示例 9：将分段下载与 20 个并发连接结合使用来下载单个对象，以在高带宽网络上实现最大吞吐量。

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

示例 10：使用分段下载将所有与键前缀匹配的对象下载到本地文件夹，继续处理过去的单个文件失败而不是中止。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [CopyObject](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

使用条件请求复制对象。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [CopyObject](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

复制对象。

```
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__

```

复制对象并向目标对象添加服务器端加密。

```
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__
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考中的 [CopyObject](#)。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [CopyObject](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [CopyObject](#)。

Swift

适用于 Swift 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》中的 [CopyObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateBucket** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateBucket。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [配置管理入门](#)
- [机器学习特征存放区入门](#)
- [对象存储入门](#)
- [使用 S3 管理大型消息](#)
- [处理版本控制对象](#)

.NET

适用于 .NET 的 SDK (v4)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [CreateBucket](#)。

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建一个启用对象锁定的存储桶。

```
/// <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;
    }
}

```

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [CreateBucket](#)。

Bash

AWS CLI 及 Bash 脚本

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CreateBucket](#)。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
```



```
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考中的 [CreateBucket](#)。

CLI

AWS CLI

示例 1：创建存储桶

以下 create-bucket 示例创建一个名为 amzn-s3-demo-bucket 的存储桶：

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

输出：

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

有关更多信息，请参阅《Amazon S3 用户指南》中的[创建存储桶](#)。

示例 2：创建带有强制拥有者的存储桶

以下 create-bucket 示例创建一个名为 amzn-s3-demo-bucket 的存储桶，该存储桶对于 S3 对象所有权使用强制存储桶所有者设置。

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

输出：

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

有关更多信息，请参阅《Amazon S3 用户指南》中的[控制对象所有权和禁用 ACL](#)。

示例 3：在“us-east-1”区域之外创建存储桶

以下 create-bucket 示例在 eu-west-1 区域中创建名为 amzn-s3-demo-bucket 的存储桶。us-east-1 之外的区域需要指定相应的 LocationConstraint 才能在所需区域创建存储桶。

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

输出：

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

有关更多信息，请参阅《Amazon S3 用户指南》中的[创建存储桶](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CreateBucket](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用默认配置创建存储桶。

```
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
}
```

创建带有对象锁定功能的存储桶，并等待该存储桶创建成功。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [CreateBucket](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建存储桶。

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

创建一个启用对象锁定的存储桶。

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

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 [CreateBucket](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建存储桶。

```
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;  
    }  
}  
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-creating-buckets.html#s3-example-creating-buckets-new-bucket-2>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [CreateBucket](#)。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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")  
    }  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [CreateBucket](#)。

PHP

适用于 PHP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建存储桶。

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

- 有关 API 详细信息，请参阅 适用于 PHP 的 AWS SDK API 参考中的 [CreateBucket](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用默认设置创建存储桶。

```

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

```

使用生命周期配置创建版本控制的桶。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [CreateBucket](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__

```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考中的 [CreateBucket](#)。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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))
    }
}))
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [CreateBucket](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.
    " determine our region from our session
```



```

        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstnt( 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.

```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [CreateBucket](#)。

Swift

适用于 Swift 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 [CreateBucket](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateMultiRegionAccessPoint** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 CreateMultiRegionAccessPoint。

Kotlin

SDK for Kotlin

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

将 S3 控制客户端配置为向 us-west-2 区域发送请求。

```
suspend fun createS3ControlClient(): S3ControlClient {
    // Configure your S3ControlClient to send requests to US West
    (Oregon).
    val s3Control = S3ControlClient.fromEnvironment {
        region = "us-west-2"
    }
    return s3Control
}
```

创建多区域接入点。

```
suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrapName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrapName
                regions = listOf(
                    Region {
                        bucket = bucketName1
                    },
                ),
            }
        }
}
```

```

                Region {
                    bucket = bucketName2
                },
            )
        }
    }

    val requestToken: String? = createMrapResponse.requestTokenArn

    // Use the request token to check for the status of the
    CreateMultiRegionAccessPoint operation.
    if (requestToken != null) {
        waitForSucceededStatus(s3Control, requestToken, accountIdParam)
        println("MRAP created")
    }

    val getMrapResponse =
        s3Control.getMultiRegionAccessPoint(
            input = GetMultiRegionAccessPointRequest {
                accountId = accountIdParam
                name = mrapName
            },
        )
    val mrapAlias = getMrapResponse.accessPoint?.alias
    return "arn:aws:s3:::$accountIdParam:accesspoint/$mrapAlias"
}

```

等待多区域接入点变为可用状态。

```

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)
        }
    }
```

- 有关更多信息，请参阅 [AWS SDK for Kotlin 开发人员指南](#)。
- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 [CreateMultiRegionAccessPoint](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateMultipartUpload** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateMultipartUpload。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [执行分段复制](#)
- [使用校验和](#)
- [使用 Amazon S3 对象完整性](#)

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
//! 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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [CreateMultipartUpload](#)。

CLI

AWS CLI

以下命令在存储桶 `amzn-s3-demo-bucket` 中创建带有密钥 `multipart/01` 的分段上传：

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

输出：

```
{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
  "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3
  "Key": "multipart/01"
}
```

完成的文件在存储桶 `amzn-s3-demo-bucket` 中名为 `multipart` 文件夹中将命名为 `01`。保存上传 ID、密钥和存储桶名称以供 `upload-part` 命令使用。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CreateMultipartUpload](#)。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// 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?;
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 [CreateMultipartUpload](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreatePresignedPost** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 CreatePresignedPost。

.NET

适用于 .NET 的 SDK (v4)

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建预签名 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);  
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [CreatePresignedPost](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucket** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucket。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [配置管理入门](#)
- [机器学习特征存放区入门](#)
- [对象存储入门](#)

.NET

适用于 .NET 的 SDK (v4)

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/// <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;
        }
    }
}

```

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [DeleteBucket](#)。

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucket](#)。

C++

SDK for C++

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考中的 [DeleteBucket](#)。

CLI

AWS CLI

以下命令删除名为 `amzn-s3-demo-bucket` 的存储桶：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucket](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 AWS SDK API 参考中的 [DeleteBucket](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 [DeleteBucket](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除存储桶。

```
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;
    }
  }
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-creating-buckets.html#s3-example-deleting-buckets>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [DeleteBucket](#)。

PHP

适用于 PHP 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除空桶。

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

- 有关 API 的详细信息，请参阅 适用于 PHP 的 AWS SDK API 参考中的 [DeleteBucket](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令从存储桶“test-files”中移除所有对象和对象版本，然后删除该存储桶。在继续操作之前，该命令将提示进行确认。添加 -Force 开关可禁止确认。请注意，不能删除不为空的存储桶。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucket](#)。

适用于 PowerShell V5 的工具

示例 1：此命令从存储桶“test-files”中移除所有对象和对象版本，然后删除该存储桶。在继续操作之前，该命令将提示进行确认。添加 -Force 开关可禁止确认。请注意，不能删除不为空的存储桶。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucket](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [DeleteBucket](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考中的 [DeleteBucket](#)。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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))
            }
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [DeleteBucket](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DeleteBucket](#)。

Swift

适用于 Swift 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》中的 [DeleteBucket](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketAnalyticsConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketAnalyticsConfiguration。

CLI

AWS CLI

删除存储桶的分析配置

以下 delete-bucket-analytics-configuration 示例移除指定存储桶和 ID 的分析配置。

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

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketAnalyticsConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testfilter”的分析筛选条件。


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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketAnalyticsConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testfilter”的分析筛选条件。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketAnalyticsConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketCors** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketCors。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [DeleteBucketCors](#)。

CLI

AWS CLI

以下命令从名为 `amzn-s3-demo-bucket` 的存储桶中删除跨源资源共享配置：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketCors](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [DeleteBucketCors](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [DeleteBucketCors](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DeleteBucketCors](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketEncryption** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketEncryption。

CLI

AWS CLI

删除存储桶的服务器端加密配置

以下 delete-bucket-encryption 示例删除指定存储桶的服务器端加密配置。

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

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketEncryption](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：这将禁用为所提供的 S3 存储桶启用的加密。

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

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketEncryption](#)。

适用于 PowerShell V5 的工具

示例 1：这将禁用为所提供的 S3 存储桶启用的加密。

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

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketEncryption](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketInventoryConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketInventoryConfiguration。

CLI

AWS CLI

删除存储桶的清单配置

以下 delete-bucket-inventory-configuration 示例删除指定存储桶的 ID 为 1 的清单配置。

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

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketInventoryConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令移除与给定 S3 存储桶相对应的名为“testInventoryName”的清单。

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

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketInventoryConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：此命令移除与给定 S3 存储桶相对应的名为“testInventoryName”的清单。

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

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketInventoryConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketLifecycle** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketLifecycle。

.NET

适用于 .NET 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [DeleteBucketLifecycle](#)。

CLI

AWS CLI

以下命令从名为 `amzn-s3-demo-bucket` 的存储桶中删除生命周期配置：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketLifecycle](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [DeleteBucketLifecycle](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DeleteBucketLifecycle](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketMetricsConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketMetricsConfiguration。

CLI

AWS CLI

删除存储桶的指标配置

以下 delete-bucket-metrics-configuration 示例移除指定存储桶和 ID 的指标配置。

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

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketMetricsConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testmetrics”的指标筛选条件。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketMetricsConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testmetrics”的指标筛选条件。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketMetricsConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketPolicy** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketPolicy。

C++

SDK for C++

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();  
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [DeleteBucketPolicy](#)。

CLI

AWS CLI

以下命令从名为 `amzn-s3-demo-bucket` 的存储桶中删除存储桶策略：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketPolicy](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteBucketPolicy](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除存储桶策略。

```
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;
  }
}
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-bucket-policies.html#s3-example-bucket-policies-delete-policy>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [DeleteBucketPolicy](#)。

Kotlin

SDK for Kotlin

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [DeleteBucketPolicy](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令移除与给定 S3 存储桶关联的存储桶策略。

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketPolicy](#)。

适用于 PowerShell V5 的工具

示例 1：该命令移除与给定 S3 存储桶关联的存储桶策略。

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketPolicy](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [DeleteBucketPolicy](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [DeleteBucketPolicy](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DeleteBucketPolicy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketReplication** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketReplication。

CLI

AWS CLI

以下命令从名为 amzn-s3-demo-bucket 的存储桶中删除复制配置：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketReplication](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：删除与名为“amzn-s3-demo-bucket”的存储桶关联的复制配置。请注意，此操作需要 s3:DeleteReplicationConfiguration 操作的权限。在操作继续之前，系统将提示您进行确认 - 要取消确认，请使用 -Force 开关。

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketReplication](#)。

适用于 PowerShell V5 的工具

示例 1：删除与名为“amzn-s3-demo-bucket”的存储桶关联的复制配置。请注意，此操作需要 s3:DeleteReplicationConfiguration 操作的权限。在操作继续之前，系统将提示您进行确认 - 要取消确认，请使用 -Force 开关。

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketReplication](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketTagging** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketTagging。

CLI

AWS CLI

以下命令从名为 amzn-s3-demo-bucket 的存储桶中删除标记配置：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketTagging](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令移除与给定 S3 存储桶关联的所有标签。

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketTagging](#)。

适用于 PowerShell V5 的工具

示例 1：此命令移除与给定 S3 存储桶关联的所有标签。

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `DeleteBucketWebsite` 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteBucketWebsite`。

C++

SDK for C++

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [DeleteBucketWebsite](#)。

CLI

AWS CLI

以下命令从名为 `amzn-s3-demo-bucket` 的存储桶中删除网站配置：

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

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteBucketWebsite](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteBucketWebsite](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

从存储桶中删除网站配置。

```
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;
    }
}
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-static-web-host.html#s3-example-static-web-host-delete-website>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [DeleteBucketWebsite](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令禁用给定 S3 存储桶的静态网站托管属性。

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteBucketWebsite](#)。

适用于 PowerShell V5 的工具

示例 1：此命令禁用给定 S3 存储桶的静态网站托管属性。

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteBucketWebsite](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [使用 Amazon S3 对象完整性](#)
- [处理版本控制对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除不受版本控制的 S3 存储桶中的对象。

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

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string keyName = "testfile.txt";

        // If the Amazon S3 bucket is located in an AWS Region other than the
        // Region of the default account, define the AWS Region for the
        // Amazon S3 bucket in your call to the AmazonS3Client constructor.
        // For example RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();
        await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
    }

    /// <summary>
```

```

    /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
    /// desired object from the named bucket.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to delete
    /// an object from an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the bucket from which the
    /// object will be deleted.</param>
    /// <param name="keyName">The name of the object to delete.</param>
    public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            var deleteObjectRequest = new DeleteObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
            };

            Console.WriteLine($"Deleting object: {keyName}");
            await client.DeleteObjectAsync(deleteObjectRequest);
            Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' when deleting an object.");
        }
    }
}

```

删除受版本控制的 S3 存储桶中的对象。

```

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

/// <summary>
/// This example creates an object in an Amazon Simple Storage Service

```

```

    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
keyName);

                // Delete the object by specifying an object key and a version
ID.

                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,

```

```

        };

        Console.WriteLine("Deleting an object");
        await client.DeleteObjectAsync(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

/// <summary>
/// This method is used to create the temporary Amazon S3 object.
/// </summary>
/// <param name="client">The initialized Amazon S3 object which will be
used
/// to create the temporary Amazon S3 object.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
object
/// will be created.</param>
/// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
/// <returns>The Version ID of the created object.</returns>
public static async Task<string> PutAnObject(IAmazonS3 client, string
bucketName, string objectKey)
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await client.PutObjectAsync(request);
    return response.VersionId;
}
}

```

- 有关更多信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [DeleteObject](#)。

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteObject](#)。

C++

SDK for C++

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();  
}
```

- 有关更多信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [DeleteObject](#)。

CLI

AWS CLI

以下命令从名为 `amzn-s3-demo-bucket` 的存储桶中删除名为 `test.txt` 的对象：

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

如果启用了存储桶版本控制，则输出将包含删除标记的版本 ID：

```
{  
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1WlGq",  
  "DeleteMarker": true  
}
```

有关删除对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“删除对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteObject](#)。

Go

适用于 Go 的 SDK V2

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关更多信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [DeleteObject](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关更多信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteObject](#)。

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除对象。

```
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;
    }
  }
}
```

```
    }  
  }  
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [DeleteObject](#)。

PHP

适用于 PHP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;  
    }  
}
```


- 有关更多信息，请参阅《适用于 PHP 的 AWS SDK API Reference》中的 [DeleteObject](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除对象。

```
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

```

通过删除对象的较高版本，将对象回滚到以前的版本。

```

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}."
    )

```

通过移除对象的活动删除标记来恢复已删除对象。

```

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)

```

创建一个 Lambda 处理程序，从 S3 对象中移除删除标记。此处理程序可用于有效地清理版本控制的桶中无关的删除标记。

```

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,  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [DeleteObject](#)。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [DeleteObject](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DeleteObject](#)。

Swift

适用于 Swift 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。


```
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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》中的 [DeleteObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObjectTagging** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteObjectTagging。

CLI

AWS CLI

删除对象的标签集

以下 delete-object-tagging 示例从对象 doc1.rtf 中删除带有指定键的标签。

```
aws s3api delete-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteObjectTagging](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令移除给定 S3 存储桶中与键为“testfile.txt”的对象关联的所有标签。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteObjectTagging](#)。

适用于 PowerShell V5 的工具

示例 1：此命令移除给定 S3 存储桶中与键为“testfile.txt”的对象关联的所有标签。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

输出：

```
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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteObjectTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObjects** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteObjects。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [删除存储桶中的所有对象](#)
- [对象存储入门](#)

.NET

适用于 .NET 的 SDK (v4)

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [DeleteObjects](#)。

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除不受版本控制的 S3 存储桶中的多个对象。

```
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
    /// 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;
    }
}

```

删除受版本控制的 S3 存储桶中的多个对象。

```

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
deleted
    /// them again. The method returns a list of the Amazon S3 objects
deleted.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created and then deleted.</param>
    /// <returns>A list of DeletedObjects.</returns>
    public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

        // Delete objects using only keys. Amazon S3 creates a delete marker
and
        // returns its version ID in the response.
        List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
        return deletedObjects;
    }

    /// <summary>
    /// This method creates several temporary objects and then deletes them.
    /// </summary>
    /// <param name="client">The S3 client.</param>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
    {
        // Upload the sample objects.

```

```

        var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

        // Delete the specific object versions.
        await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
    }

    /// <summary>
    /// Displays the list of information about deleted files to the console.
    /// </summary>
    /// <param name="e">Error information from the delete process.</param>
    private static void DisplayDeletionErrors>DeleteObjectsException e)
    {
        var errorResponse = e.Response;
        Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
        Console.WriteLine("Printing error data...");
        foreach (var deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// Delete multiple objects from a version-enabled bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    private static async Task VersionedDeleteAsync(IAmazonS3 client, string
bucketName, List<KeyVersion> keys)
    {
        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keys, // This includes the object keys and specific
version IDs.

```

```

        };

        try
        {
            Console.WriteLine("Executing VersionedDelete...");
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted all the
{response.DeletedObjects.Count} items");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    /// <returns>A list of the deleted objects.</returns>
    private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
keys)
    {
        // Create a request that includes only the object key names.
        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
        multiObjectDeleteRequest.BucketName = bucketName;

        foreach (var key in keys)
        {
            multiObjectDeleteRequest.AddKey(key.Key);
        }

        // Execute DeleteObjectsAsync.
        // The DeleteObjectsAsync method adds a delete marker for each

```

```

        // object deleted. You can verify that the objects were removed
        // using the Amazon S3 console.
        DeleteObjectsResponse response;
        try
        {
            Console.WriteLine("Executing NonVersionedDelete...");
            response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
            throw; // Some deletions failed. Investigate before continuing.
        }

        // This response contains the DeletedObjects list which we use to
delete the delete markers.
        return response.DeletedObjects;
    }

    /// <summary>
    /// Deletes the markers left after deleting the temporary objects.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
    private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
    {
        var keyVersionList = new List<KeyVersion>();

        foreach (var deletedObject in deletedObjects)
        {
            KeyVersion keyVersion = new KeyVersion
            {
                Key = deletedObject.Key,
                VersionId = deletedObject.DeleteMarkerVersionId,

```

```

        };
        keyVersionList.Add(keyVersion);
    }

    // Create another request to delete the delete markers.
    var multiObjectDeleteRequest = new DeleteObjectsRequest
    {
        BucketName = bucketName,
        Objects = keyVersionList,
    };

    // Now, delete the delete marker to bring your objects back to the
    bucket.
    try
    {
        Console.WriteLine("Removing the delete markers .....");
        var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
    }
    catch (DeleteObjectsException ex)
    {
        DisplayDeletionErrors(ex);
    }
}

/// <summary>
/// Create temporary Amazon S3 objects to show how object deletion works
in an
/// Amazon S3 bucket with versioning enabled.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// PutObjectAsync to create temporary objects for the example.</param>
/// <param name="bucketName">A string representing the name of the S3
/// bucket where we will create the temporary objects.</param>
/// <param name="number">The number of temporary objects to create.</
param>
/// <returns>A list of the KeyVersion objects.</returns>
private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
{
    var keys = new List<KeyVersion>();

```

```

    for (var i = 0; i < number; i++)
    {
        string key = "ObjectToDelete-" + new System.Random().Next();
        PutObjectRequest request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = key,
            ContentBody = "This is the content body!",
        };

        var response = await client.PutObjectAsync(request);
        KeyVersion keyVersion = new KeyVersion
        {
            Key = key,
            VersionId = response.VersionId,
        };

        keys.Add(keyVersion);
    }

    return keys;
}
}

```

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [DeleteObjects](#)。

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# 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
}

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteObjects](#)。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考中的 [DeleteObjects](#)。

CLI

AWS CLI

以下命令从名为 `amzn-s3-demo-bucket` 的存储桶中删除对象：

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` 是当前目录中指定要删除的对象的 JSON 文档：

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

输出：

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

```
]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteObjects](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 AWS SDK API 参考中的 [DeleteObjects](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

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

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

        Usage:    <bucketName>
```

```
Where:
    bucketName - the Amazon S3 bucket name.
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteBucketObjects(s3, bucketName);
s3.close();
}

/**
 * Deletes multiple objects from an Amazon S3 bucket.
 *
 * @param s3 An Amazon S3 client object.
 * @param bucketName The name of the Amazon S3 bucket to delete objects from.
 */
public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
    }
}
```

```
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
        DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(del)
            .build();

        s3.deleteObjects(multiObjectDeleteRequest);
        System.out.println("Multiple objects are deleted!");

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

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 [DeleteObjects](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除多个对象。

```
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;
    }
}
};
```

- 有关 API 详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考中的 [DeleteObjects](#)。

Kotlin

SDK for Kotlin

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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")
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [DeleteObjects](#)。

PHP

适用于 PHP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

从键列表中删除一组对象。

```

$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.");
}
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 AWS SDK API 参考中的 [DeleteObjects](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令从存储桶“test-files”中移除对象“sample.txt”。在命令执行之前，系统会提示您进行确认；要取消提示，请使用 -Force 开关。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

示例 2：假设存储桶已配置为启用对象版本，则此命令会从存储桶“test-files”中移除对象“sample.txt”的指定版本。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

示例 3：此命令通过单个批量操作，从存储桶“test-files”中移除对象“sample1.txt”、“sample2.txt”和“sample3.txt”。无论删除的成功或错误状态如何，服务响应都将列出所有已处理的键。要仅获取服务无法处理的键的错误，请添加 -ReportErrorsOnly 参数（也可以使用别名 -Quiet 来指定此参数）。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

示例 4：此示例使用带有 -KeyCollection 参数的内联表达式来获取要删除的对象的键。Get-S3Object 返回 Amazon.S3.Model.S3Object 实例的集合，每个实例都有一个标识对象的字符串类型的 Key（键）成员。

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

示例 5：此示例获取存储桶中所有具有键前缀“prefix/subprefix”的对象并将其删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传递给 cmdlet 的 -InputObject (别名 -S3ObjectCollection) 参数，这样，只需调用一次服务即可批量删除。

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

示例 6：此示例将代表删除标记的 Amazon.S3.Model.S3ObjectVersion 实例的集合传送到 cmdlet 进行删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传递给 cmdlet 的 -InputObject (别名 -S3ObjectCollection) 参数，这样，只需调用一次服务即可批量删除。

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

示例 7：此脚本显示如何通过构造与 -KeyAndVersionCollection 参数一起使用的对象数组，来批量删除一组对象 (在本例中为删除标记)。

```
$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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeleteObjects](#)。

适用于 PowerShell V5 的工具

示例 1：此命令从存储桶“test-files”中移除对象“sample.txt”。在命令执行之前，系统会提示您进行确认；要取消提示，请使用 -Force 开关。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

示例 2：假设存储桶已配置为启用对象版本，则此命令会从存储桶“test-files”中移除对象“sample.txt”的指定版本。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

示例 3：此命令通过单个批量操作，从存储桶“test-files”中移除对象“sample1.txt”、“sample2.txt”和“sample3.txt”。无论删除的成功或错误状态如何，服务响应都将列出所有已处理的键。要仅获取服务无法处理的键的错误，请添加 -ReportErrorsOnly 参数（也可以使用别名 -Quiet 来指定此参数）。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

示例 4：此示例使用带有 -KeyCollection 参数的内联表达式来获取要删除的对象的键。Get-S3Object 返回 Amazon.S3.Model.S3Object 实例的集合，每个实例都有一个标识对象的字符串类型的 Key（键）成员。

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

示例 5：此示例获取存储桶中所有具有键前缀“prefix/subprefix”的对象并将其删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传递给 cmdlet 的 -InputObject（别名 -S3ObjectCollection）参数，这样，只需调用一次服务即可批量删除。

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

示例 6：此示例将代表删除标记的 Amazon.S3.Model.S3ObjectVersion 实例的集合传送到 cmdlet 进行删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传递给 cmdlet 的 -InputObject（别名 -S3ObjectCollection）参数，这样，只需调用一次服务即可批量删除。

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

示例 7：此脚本显示如何通过构造与 -KeyAndVersionCollection 参数一起使用的对象数组，来批量删除一组对象（在本例中为删除标记）。

```
$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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeleteObjects](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用对象键列表删除一组对象。

```
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

```

删除存储桶中的所有对象。

```

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

```

通过删除版本控制对象的所有版本永久删除该对象。

```

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

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [DeleteObjects](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考中的 [DeleteObjects](#)。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [DeleteObjects](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DeleteObjects](#)。

Swift

适用于 Swift 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》中的 [DeleteObjects](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeletePublicAccessBlock** 与 CLI 配合使用

以下代码示例演示如何使用 DeletePublicAccessBlock。

CLI

AWS CLI

删除存储桶的屏蔽公共访问权限配置

以下 `delete-public-access-block` 示例移除指定存储桶上的屏蔽公共访问权限配置。

```
aws s3api delete-public-access-block \  
  --bucket amzn-s3-demo-bucket
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeletePublicAccessBlock](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令关闭给定存储桶的屏蔽公共访问权限配置。

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

输出：

```
amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DeletePublicAccessBlock](#)。

适用于 PowerShell V5 的工具

示例 1：此命令关闭给定存储桶的屏蔽公共访问权限配置。

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

输出：

```
amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [DeletePublicAccessBlock](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketAccelerateConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketAccelerateConfiguration。

CLI

AWS CLI

检索存储桶的加速配置

以下 get-bucket-accelerate-configuration 示例检索指定存储桶的加速配置。

```
aws s3api get-bucket-accelerate-configuration \
  --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "Status": "Enabled"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketAccelerateConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：如果为指定的存储桶启用了传输加速设置，则此命令将返回值 Enabled。

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Value
-----
Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketAccelerateConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：如果为指定的存储桶启用了传输加速设置，则此命令将返回值 Enabled。

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Value
-----
Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketAccelerateConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketAcl。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [检查存储桶是否存在](#)
- [管理访问控制列表 \(ACL \)](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetBucketAcl](#)。

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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";
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [GetBucketAcl](#)。

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶的访问控制列表：

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketAcl](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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 "";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetBucketAcl](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取 ACL 权限。

```
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;
    }
}
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-access-permissions.html#s3-example-access-permissions-get-acl>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [GetBucketAcl](#)。

PowerShell

适用于 PowerShell V5 的工具

示例 1：该命令获取 S3 对象的对象所有者的详细信息。

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

输出：

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketAcl](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```


- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetBucketAcl](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetBucketAcl](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketAnalyticsConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketAnalyticsConfiguration。

CLI

AWS CLI

检索具有特定 ID 的存储桶的分析配置

以下 get-bucket-analytics-configuration 示例显示了指定存储桶和 ID 的分析配置。

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

输出：

```
{  
  "AnalyticsConfiguration": {  
    "StorageClassAnalysis": {},  
    "Id": "1"  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketAnalyticsConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶中名为“testfilter”的分析筛选条件的详细信息。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketAnalyticsConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回给定 S3 存储桶中名为“testfilter”的分析筛选条件的详细信息。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketAnalyticsConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketCors** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetBucketCors`。

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetBucketCors](#)。

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶的跨源资源共享配置：

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

输出：

```
{
  "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": [
        "*"
      ]
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketCors](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取存储桶的 CORS 策略。

```
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}`,
      );
    });
  } catch (err) {
    if (err instanceof S3ServiceException) {
      console.log(`S3ServiceException: ${err.message}`);
    } else {
      console.log(`Error: ${err.message}`);
    }
  }
};
```

```
        `\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;
    }
}
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-configuring-buckets.html#s3-example-configuring-buckets-get-cors>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API 参考》中的 [GetBucketCors](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetBucketCors](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  # configuration for the bucket.
  def cors
    @bucket_cors.data
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
      why: #{e.message}"
      nil
    end
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [GetBucketCors](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetBucketCors](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketEncryption** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketEncryption。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

    /// <summary>
    /// Get and print the encryption settings of a bucket.
    /// </summary>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task GetEncryptionSettings(string bucketName)
    {
        // Check and print the bucket encryption settings.
        Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

        try
        {
            var settings =
                await _s3Client.GetBucketEncryptionAsync(
                    new GetBucketEncryptionRequest() { BucketName =
bucketName });

            foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
            {
                Console.WriteLine(
                    $"Algorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
                Console.WriteLine(
                    $"Key:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementServi
}
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                    ? $"Bucket {bucketName} was not found."
                    : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
            }
        }
    }

```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetBucketEncryption](#)。

CLI

AWS CLI

检索存储桶的服务器端加密配置

以下 `get-bucket-encryption` 示例检索存储桶 `amzn-s3-demo-bucket` 的服务器端加密配置。

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

输出：

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketEncryption](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回与给定存储桶关联的所有服务器端加密规则。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketEncryption](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回与给定存储桶关联的所有服务器端加密规则。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketEncryption](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketInventoryConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketInventoryConfiguration。

CLI

AWS CLI

检索存储桶的清单配置

以下 get-bucket-inventory-configuration 示例检索 ID 为 1 的指定存储桶的清单配置。

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

输出：

```
{  
  "InventoryConfiguration": {  
    "IsEnabled": true,  
    "Destination": {  
      "S3BucketDestination": {  
        "Format": "ORC",  
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",  
        "AccountId": "123456789012"  
      }  
    },  
    "IncludedObjectVersions": "Current",  
    "Id": "1",  
    "Schedule": {
```

```
        "Frequency": "Weekly"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketInventoryConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的名为“testinventory”的清单的详细信息。

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketInventoryConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回给定 S3 存储桶的名为“testinventory”的清单的详细信息。

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketInventoryConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketLifecycleConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetBucketLifecycleConfiguration`。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetBucketLifecycleConfiguration](#)。

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶的生命周期配置：

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "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"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketLifecycleConfiguration](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetBucketLifecycleConfiguration](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetBucketLifecycleConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketLocation** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketLocation。

CLI

AWS CLI

如果存在约束条件，则以下命令会检索名为 amzn-s3-demo-bucket 的存储桶的位置约束：

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

输出：

```
{  
  "LocationConstraint": "us-west-2"  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketLocation](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：如果存在约束，则此命令返回存储桶“amzn-s3-demo-bucket”的位置约束。

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Value  
-----  
ap-south-1
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketLocation](#)。

适用于 PowerShell V5 的工具

示例 1：如果存在约束，则此命令返回存储桶“amzn-s3-demo-bucket”的位置约束。

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Value  
-----  
ap-south-1
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketLocation](#)。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [GetBucketLocation](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketLogging** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketLogging。

CLI

AWS CLI

检索存储桶的日志记录状态

以下 get-bucket-logging 示例检索指定存储桶的日志记录状态。

```
aws s3api get-bucket-logging \
    --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "amzn-s3-demo-bucket-logs"
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketLogging](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回指定存储桶的日志记录状态。

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

输出：

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketLogging](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回指定存储桶的日志记录状态。

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

输出：

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketLogging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketMetricsConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketMetricsConfiguration。

CLI

AWS CLI

检索具有特定 ID 的存储桶的指标配置

以下 `get-bucket-metrics-configuration` 示例显示了指定存储桶和 ID 的指标配置。

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

输出：

```
{  
  "MetricsConfiguration": {  
    "Filter": {  
      "Prefix": "logs"  
    },  
    "Id": "123"  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketMetricsConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回有关给定 S3 存储桶的名为“testfilter”的指标筛选条件的详细信息。

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketMetricsConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回有关给定 S3 存储桶的名为“testfilter”的指标筛选条件的详细信息。

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketMetricsConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketNotification** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketNotification。

CLI

AWS CLI

以下命令检索名为 amzn-s3-demo-bucket 的存储桶的通知配置：

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

输出：

```
{  
  "TopicConfiguration": {  
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",  
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",  
    "Event": "s3:ObjectCreated:*",  
    "Events": [  
      "s3:ObjectCreated:*"  
    ]  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketNotification](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索给定存储桶的通知配置

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty  
TopicConfigurations
```

输出：

```
Id    Topic  
--    -  
mimo  arn:aws:sns:eu-west-1:123456789012:topic-1
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketNotification](#)。

适用于 PowerShell V5 的工具

示例 1：此示例检索给定存储桶的通知配置

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty  
TopicConfigurations
```

输出：

```
Id    Topic  
--    -  
mimo  arn:aws:sns:eu-west-1:123456789012:topic-1
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketNotification](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketPolicy** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 **GetBucketPolicy**。

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关更多信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [GetBucketPolicy](#)。

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶的存储桶策略：

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "Policy": "{\"Version\":\"2008-10-17\",\"Statement\": [{\"Sid\":\"\",\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::amzn-s3-demo-bucket/*\"}, {\"Sid\":\"\",\"Effect\":\"Deny\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::amzn-s3-demo-bucket/secret/*\"}]}"
}
```

获取并放置存储桶策略 以下示例演示了如何下载 Amazon S3 存储桶策略，修改文件，然后使用 `put-bucket-policy` 来应用修改后的存储桶策略。要将存储桶策略下载到文件中，您可以运行：

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

然后，您可以根据需要修改 `policy.json` 文件。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

`policy.json` 文件（根据需要）。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

文件（根据需要）。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketPolicy](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetBucketPolicy](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取存储桶策略。

```
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;  
    }  
    }  
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-bucket-policies.html#s3-example-bucket-policies-get-policy>。
- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [GetBucketPolicy](#)。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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  
    }  
}
```

```
}  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [GetBucketPolicy](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令输出与给定 S3 存储桶关联的存储桶策略。

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketPolicy](#)。

适用于 PowerShell V5 的工具

示例 1：此命令输出与给定 S3 存储桶关联的存储桶策略。

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketPolicy](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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)
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetBucketPolicy](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。


```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [GetBucketPolicy](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_s3->getbucketpolicy(
testing purposes. " oo_result is returned for
```

```
        iv_bucket = iv_bucket_name ).
    DATA(lv_policy) = oo_result->get_policy( ).
    MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetBucketPolicy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketPolicyStatus** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketPolicyStatus。

CLI

AWS CLI

检索存储桶的策略状态，此状态指示存储桶是否为公有存储桶

以下 get-bucket-policy-status 示例检索存储桶 amzn-s3-demo-bucket 的策略状态。

```
aws s3api get-bucket-policy-status \
    --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketPolicyStatus](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的策略状态，此状态指示存储桶是否为公有存储桶。

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketPolicyStatus](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回给定 S3 存储桶的策略状态，此状态指示存储桶是否为公有存储桶。

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketPolicyStatus](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketReplication** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketReplication。

CLI

AWS CLI

以下命令检索名为 amzn-s3-demo-bucket 的存储桶的复制配置：

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "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"
    }
}

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketReplication](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetBucketReplication](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回在名为“amzn-s3-demo-bucket”的存储桶上设置的复制配置信息。

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketReplication](#)。

适用于 PowerShell V5 的工具

示例 1：返回在名为“amzn-s3-demo-bucket”的存储桶上设置的复制配置信息。

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketReplication](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketRequestPayment** 与 CLI 配合使用

以下代码示例演示如何使用 `GetBucketRequestPayment`。

CLI

AWS CLI

检索存储桶的请求付款配置

以下 `get-bucket-request-payment` 示例检索指定存储桶的申请方付款配置。

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "Payer": "BucketOwner"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketRequestPayment](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回名为“amzn-s3-demo-bucket”的存储桶的请求付款配置。默认情况下，存储桶所有者支付从存储桶进行下载的费用。

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketRequestPayment](#)。

适用于 PowerShell V5 的工具

示例 1：返回名为“amzn-s3-demo-bucket”的存储桶的请求付款配置。默认情况下，存储桶所有者支付从存储桶进行下载的费用。

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketRequestPayment](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketTagging** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketTagging。

CLI

AWS CLI

以下命令检索名为 amzn-s3-demo-bucket 的存储桶的标记配置：

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketTagging](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回与给定存储桶关联的所有标签。

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketTagging](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回与给定存储桶关联的所有标签。

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketVersioning** 与 CLI 配合使用

以下代码示例演示如何使用 GetBucketVersioning。

CLI

AWS CLI

以下命令检索名为 amzn-s3-demo-bucket 的存储桶的版本控制配置：

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "Status": "Enabled"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketVersioning](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回与给定存储桶相关的版本控制状态。


```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketVersioning](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回与给定存储桶相关的版本控制状态。

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketVersioning](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketWebsite** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketWebsite。

.NET

适用于 .NET 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关更多信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetBucketWebsite](#)。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关更多信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [GetBucketWebsite](#)。

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶的静态网站配置：

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetBucketWebsite](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取网站配置。

```
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;  
  }  
}  
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [GetBucketWebsite](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的静态网站配置的详细信息。

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetBucketWebsite](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回给定 S3 存储桶的静态网站配置的详细信息。

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetBucketWebsite](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [如果对象已修改，则从桶中获取该对象](#)
- [从多区域接入点中获取对象](#)
- [加密入门](#)
- [对象存储入门](#)
- [提出条件请求](#)
- [跟踪上传和下载操作](#)

.NET

适用于 .NET 的 SDK (v4)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetObject](#)。
适用于 .NET 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用条件请求获取对象。

```
/// <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;
    }
}

```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetObject](#)。

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# 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
}

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObject](#)。

C++

SDK for C++

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [GetObject](#)。

CLI

AWS CLI

以下示例使用 `get-object` 命令从 Amazon S3 下载对象：

```
aws s3api get-object --bucket text-content --key dir/  
my_images.tar.bz2 my_images.tar.bz2
```

请注意，指定 `outfile` 参数时没有诸如“`--outfile`”之类的选项名称。输出文件的名称必须是命令中的最后一个参数。

以下示例演示了如何使用 `--range` 从对象下载特定字节范围。请注意，字节范围需要以“`bytes=`”为前缀：

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

有关检索对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“获取对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObject](#)。

Go

适用于 Go 的 SDK V2

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [GetObject](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 [S3Client](#) 以字节数组读取数据。

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

使用 [S3TransferManager](#) 将 S3 存储桶中的[对象下载](#)到本地文件中。查看[完整文件](#)并[进行测试](#)。

```
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();
    }

```

使用 [S3Client](#) 读取属于对象的标签。

```

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);
    }
}
}
```

使用 [S3Client](#) 获取对象的 URL。

```
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);
        }
    }
}

```

使用 [S3Client](#) 通过 S3Presigner 客户端对象获取对象。

```

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();
    }
}
}

```

使用 ResponseTransformer 对象和 [S3Client](#) 获取对象。

```
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);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetObject](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

下载对象。

```
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 'transformToArray' 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;
    }
}
};

```

在对象的 ETag 与提供的 ETag 匹配的条件下下载该对象。

```

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"));
  }
}

```

在对象的 ETag 与提供的 ETag 不匹配的条件下下载该对象。

```

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"));
  }
}

```

使用在给定时间范围内创建或修改对象的条件来下载该对象。

```

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"));
  }
}

```

使用在给定时间范围内尚未创建或修改对象的条件来下载该对象。

```

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

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
  }

```



```

    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();

```

```
if (!errors) {  
    main(results.values);  
} else {  
    console.error(errors.join("\n"));  
}  
}
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-creating-buckets.html#s3-example-creating-buckets-get-object>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [GetObject](#)。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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")  
        }  
    }
```

```
    }  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [GetObject](#)。

PHP

适用于 PHP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取对象。

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

- 有关 API 的详细信息，请参阅 适用于 PHP 的 AWS SDK API 参考中的 [GetObject](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令从存储桶“amzn-s3-demo-bucket”中检索项目“sample.txt”，并将其保存到当前位置中名为“local-sample.txt”的文件。在调用此命令之前，文件“local-sample.txt”不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

示例 2：此命令从存储桶“amzn-s3-demo-bucket”中检索虚拟目录“DIR”，并将其保存到当前位置中名为“Local-DIR”的文件夹。在调用此命令之前，文件夹“Local-DIR”不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

示例 3：将键以“.json”结尾的所有对象从存储桶名称中带有“config”的存储桶下载到指定文件夹中的文件。对象键用于设置文件名。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetObject](#)。

适用于 PowerShell V5 的工具

示例 1：此命令从存储桶“amzn-s3-demo-bucket”中检索项目“sample.txt”，并将其保存到当前位置中名为“local-sample.txt”的文件。在调用此命令之前，文件“local-sample.txt”不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

示例 2：此命令从存储桶“amzn-s3-demo-bucket”中检索虚拟目录“DIR”，并将其保存到当前位置中名为“Local-DIR”的文件夹。在调用此命令之前，文件夹“Local-DIR”不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

示例 3：将键以“.json”结尾的所有对象从存储桶名称中带有“config”的存储桶下载到指定文件夹中的文件。对象键用于设置文件名。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

示例 4：使用分段并行下载来下载单个对象，以提高处理大型文件的性能。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

示例 5：使用 RANGE 模式以自定义 64 MB 分段大小下载大型对象。RANGE 模式使用 HTTP 字节范围请求并适用于所有对象，而无论它们是如何上传的。

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

示例 6：将分段下载与并发文件下载结合使用，以将所有带有键前缀“data/”的对象下载到本地文件夹。

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

示例 7：将分段下载与 20 个并发连接结合使用来下载单个对象，以在高带宽网络上实现最大吞吐量。

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

示例 8：使用分段下载将所有带有键前缀“data/”的对象下载到本地文件夹，继续处理过去的单个文件失败而不是中止。

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

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetObject](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

使用条件请求获取对象。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》<https://docs.aws.amazon.com/goto/boto3/s3-2006-03-01/GetObject>中的 GetObject。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取对象。

```
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__

```

获取对象并报告其服务器端加密状态。

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

```

```

# @param object [Aws::S3::Object] An existing Amazon S3 object.
def initialize(object)
  @object = object
end

# Gets the object into memory.
#
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
successful; otherwise nil.
def object
  @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [GetObject](#)。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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)
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [GetObject](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_s3->getobject(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    DATA(lv_object_data) = oo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetObject](#)。

Swift

适用于 Swift 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》中的 [GetObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetObjectAcl`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [管理访问控制列表 \(ACL\)](#)

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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";
        }
    }
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [GetObjectAcl](#)。

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶中对象的访问控制列表：

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

输出：

```
{
  "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"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObjectAcl](#)。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [GetObjectAcl](#)。

PowerShell

适用于 PowerShell V5 的工具

示例 1：该命令获取 S3 对象的对象所有者的详细信息。

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

输出：

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetObjectAcl](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetObjectAcl](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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.

```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetObjectAcl](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectAttributes** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetObjectAttributes`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [使用 Amazon S3 对象完整性](#)

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// ! 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;
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [GetObjectAttributes](#)。

CLI

AWS CLI

从对象检索元数据而不返回对象本身

以下 get-object-attributes 示例从对象 doc1.rtf 检索元数据。

```

aws s3api get-object-attributes \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

输出：

```

{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
  "StorageClass": "STANDARD",
  "ObjectSize": 405
}

```

```
}
```

有关更多信息，请参阅《Amazon S3 API Reference》中的 [GetObjectAttributes](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObjectAttributes](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectLegalHold** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectLegalHold。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [锁定 Amazon S3 对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetObjectLegalHold](#)。

CLI

AWS CLI

检索对象的法定保留状态

以下 get-object-legal-hold 示例检索指定对象的法定保留状态。

```
aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

输出：

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObjectLegalHold](#)。

Go

适用于 Go 的 SDK V2

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [GetObjectLegalHold](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// 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;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetObjectLegalHold](#)。

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [GetObjectLegalHold](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

放置对象法定保留。

```
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,
        )
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetObjectLegalHold](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetObjectLegalHold](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectLockConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectLockConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [锁定 Amazon S3 对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetObjectLockConfiguration](#)。

CLI

AWS CLI

检索存储桶的对象锁定配置

以下 `get-object-lock-configuration` 示例检索指定存储桶的对象锁定配置。

```
aws s3api get-object-lock-configuration \  
    --bucket amzn-s3-demo-bucket-with-object-lock
```

输出：

```
{  
  "ObjectLockConfiguration": {  
    "ObjectLockEnabled": "Enabled",  
    "Rule": {  
      "DefaultRetention": {  
        "Mode": "COMPLIANCE",  
        "Days": 50  
      }  
    }  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObjectLockConfiguration](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [GetObjectLockConfiguration](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetObjectLockConfiguration](#)。

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [GetObjectLockConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：如果为给定的 S3 存储桶启用了对象锁定配置，则此命令将返回值“Enabled”。

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

输出：

```
Value
-----
Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetObjectLockConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：如果为给定的 S3 存储桶启用了对象锁定配置，则此命令将返回值“Enabled”。

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

输出：

```
Value
-----
Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetObjectLockConfiguration](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取对象锁定配置。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetObjectLockConfiguration](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetObjectLockConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectRetention** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetObjectRetention`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [锁定 Amazon S3 对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/// <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($"{objectKey} in
{bucketName}: " +
            $"{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}");
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to fetch object lock retention:
'{ex.Message}'");
        return new ObjectLockRetention();
    }
}
```

```
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [GetObjectRetention](#)。

CLI

AWS CLI

检索对象的对象保留配置

以下 `get-object-retention` 示例检索指定对象的对象保留配置。

```
aws s3api get-object-retention \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf
```

输出：

```
{  
  "Retention": {  
    "Mode": "GOVERNANCE",  
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObjectRetention](#)。

Go

适用于 Go 的 SDK V2

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [GetObjectRetention](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// 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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetObjectRetention](#)。

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [GetObjectRetention](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令返回保留对象之前的模式和日期。

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetObjectRetention](#)。

适用于 PowerShell V5 的工具

示例 1：该命令返回保留对象之前的模式和日期。

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetObjectRetention](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `GetObjectTagging` 与 CLI 配合使用

以下代码示例演示如何使用 `GetObjectTagging`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [标签入门](#)

CLI

AWS CLI

检索附加到对象的标签

以下 `get-object-tagging` 示例从指定的对象中检索指定键的值。

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf
```

输出：

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    }
  ]
}
```

以下 `get-object-tagging` 示例尝试检索没有标签的对象 `doc2.rtf` 的标签集。

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc2.rtf
```

输出：

```
{
```

```
"TagSet": []
}
```

以下 `get-object-tagging` 示例检索具有多个标签的对象 `doc3.rtf` 的标签集。

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc3.rtf
```

输出：

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    },
    {
      "Value": "finance",
      "Key": "department"
    },
    {
      "Value": "payroll",
      "Key": "team"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetObjectTagging](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该示例返回与给定 S3 存储桶上存在的对象关联的标签。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

输出：

Key	Value

```
--- -----  
test value
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetObjectTagging](#)。

适用于 PowerShell V5 的工具

示例 1：该示例返回与给定 S3 存储桶上存在的对象关联的标签。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Key  Value  
---  -----  
test value
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetObjectTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetPublicAccessBlock** 与 CLI 配合使用

以下代码示例演示如何使用 `GetPublicAccessBlock`。

CLI

AWS CLI

设置或修改存储桶的屏蔽公共访问权限配置

以下 `get-public-access-block` 示例显示了指定存储桶的屏蔽公共访问权限配置。

```
aws s3api get-public-access-block \  
  --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [GetPublicAccessBlock](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令返回给定 S3 存储桶的公共访问权限屏蔽设置。

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetPublicAccessBlock](#)。

适用于 PowerShell V5 的工具

示例 1：该命令返回给定 S3 存储桶的公共访问权限屏蔽设置。

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [GetPublicAccessBlock](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **HeadBucket** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 HeadBucket。

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [HeadBucket](#)。

CLI

AWS CLI

以下命令验证对名为 `amzn-s3-demo-bucket` 的存储桶的访问权限：

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

如果存储桶存在并且您可以访问它，则不返回任何输出。否则，会显示错误消息。例如：

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [HeadBucket](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [HeadBucket](#)。

PowerShell

适用于 PowerShell V5 的工具

示例 1：当用户有权访问现有存储桶时，此命令对于该存储桶返回带有 HTTP 状态代码 200 OK 的输出。只有 S3 目录存储桶支持 BucketArn 参数。

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

输出：

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength        : 0
HttpStatusCode       : OK
```

示例 2：对于不存在的存储桶，此命令会引发错误，并显示 HTTP 状态代码 NotFound。

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

输出：

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status Code NotFound. No further error information was returned by the service.
```

示例 3：对于用户无权访问的现有存储桶，此命令会引发错误，并显示 HTTP 状态代码 Forbidden。

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

输出：

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status Code Forbidden. No further error information was returned by the service.
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [HeadBucket](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [HeadBucket](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [HeadBucket](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **HeadObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 HeadObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [检查对象是否存在](#)
- [对象存储入门](#)

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶中对象的元数据：

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

输出：

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [HeadObject](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

确定对象的内容类型。

```
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);
        }
    }
}

```

获取对象的还原状态。

```

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);
    }
}
}
```

- 有关 API 详细，请参阅《AWS SDK for Java 2.x API Reference》中的 [HeadObject](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关 API 详细，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [HeadObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListBucketAnalyticsConfigurations** 与 CLI 配合使用

以下代码示例演示如何使用 ListBucketAnalyticsConfigurations。

CLI

AWS CLI

检索存储桶的分析配置列表

下面的 list-bucket-analytics-configurations 检索指定存储桶的分析配置列表。

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ]
}
```

```
    ],  
    "IsTruncated": false  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListBucketAnalyticsConfigurations](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个分析配置。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [ListBucketAnalyticsConfigurations](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个分析配置。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [ListBucketAnalyticsConfigurations](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListBucketInventoryConfigurations** 与 CLI 配合使用

以下代码示例演示如何使用 ListBucketInventoryConfigurations。

CLI

AWS CLI

检索存储桶的清单配置列表

以下 list-bucket-inventory-configurations 示例列出了指定存储桶的清单配置。

```
aws s3api list-bucket-inventory-configurations \
  --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListBucketInventoryConfigurations](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个清单配置。

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [ListBucketInventoryConfigurations](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个清单配置。

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [ListBucketInventoryConfigurations](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListBuckets** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListBuckets。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [ListBuckets](#)。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [ListBuckets](#)。

CLI

AWS CLI

以下命令使用 `list-buckets` 命令显示所有 Amazon S3 存储桶的名称（跨所有区域）：

```
aws s3api list-buckets --query "Buckets[].Name"
```

查询选项会筛选 `list-buckets` 的输出，使其范围缩小到仅限存储桶名称。

有关存储桶的更多信息，请参阅《Amazon S3 开发人员指南》中的“使用 Amazon S3 存储桶”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListBuckets](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [ListBuckets](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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());
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListBuckets](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出存储桶。

```
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}`)}`);
} catch (caught) {
  if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while listing buckets.  ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-creating-buckets.html#s3-example-creating-buckets-list-buckets>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [ListBuckets](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回所有 S3 存储桶。

```
Get-S3Bucket
```

示例 2：此命令返回名为“test-files”的存储桶

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [ListBuckets](#)。

适用于 PowerShell V5 的工具

示例 1：此命令返回所有 S3 存储桶。

```
Get-S3Bucket
```

示例 2：此命令返回名为“test-files”的存储桶

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [ListBuckets](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [ListBuckets](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.

```

```
def initialize(s3_resource)
  @s3_resource = s3_resource
end

# Lists buckets for the current account.
#
# @param count [Integer] The maximum number of buckets to list.
def list_buckets(count)
  puts 'Found these buckets:'
  @s3_resource.buckets.each do |bucket|
    puts "\t#{bucket.name}"
    count -= 1
    break if count.zero?
  end
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list buckets. Here's why: #{e.message}"
  false
end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [ListBuckets](#)。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```


- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [ListBuckets](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [ListBuckets](#)。

Swift

适用于 Swift 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 [ListBuckets](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListMultipartUploads** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListMultipartUploads。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [删除未完成的分段上传](#)

CLI

AWS CLI

以下命令列出名为 amzn-s3-demo-bucket 的存储桶的所有活动分段上传：

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

输出：

```
{
  "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": []
}
```

正在进行的分段上传会产生 Amazon S3 存储费用。完成或中止活动分段上传，可将其从您的账户中移除。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListMultipartUploads](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListMultipartUploads](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListObjectVersions** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListObjectVersions。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [对象存储入门](#)
- [处理版本控制对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        //     var client = new AmazonS3Client(RegionEndpoint.USSouth2);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
    /// This method lists all versions of the objects within an Amazon S3
    /// version enabled bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// ListVersionsAsync.</param>
    /// <param name="bucketName">The name of the version enabled Amazon S3
bucket
    /// for which you want to list the versions of the contained objects.</
param>
    public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
client, string bucketName)
    {
        try
        {
            // When you instantiate the ListVersionRequest, you can
            // optionally specify a key name prefix in the request
            // if you want a list of object versions of a specific object.

```

```
// For this example we set a small limit in MaxKeys to return
// a small list of versions.
ListVersionsRequest request = new ListVersionsRequest()
{
    BucketName = bucketName,
    MaxKeys = 2,
};

do
{
    ListVersionsResponse response = await
client.ListVersionsAsync(request);

    // Process response.
    foreach (S3ObjectVersion entry in response.Versions)
    {
        Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
    }

    // If response is truncated, set the marker to get the next
    // set of keys.
    if (response.IsTruncated)
    {
        request.KeyMarker = response.NextKeyMarker;
        request.VersionIdMarker = response.NextVersionIdMarker;
    }
    else
    {
        request = null;
    }
}
while (request != null);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: '{ex.Message}'");
}
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [ListObjectVersions](#)。

CLI

AWS CLI

以下命令检索名为 `amzn-s3-demo-bucket` 的存储桶中对象的版本信息：

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

输出：

```
{
  "DeleteMarkers": [
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": true,
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",
      "Key": "index.html",
      "LastModified": "2015-11-10T00:57:03.000Z"
    },
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
      "Key": "index.html",
      "LastModified": "2015-11-09T23:32:20.000Z"
    }
  ],
  "Versions": [
    {
      "LastModified": "2015-11-10T00:20:11.000Z",
      "VersionId": "Rb_l2T8UHDkFEwCgJjhl9P0ZC0qJ.vpD",
```



```

      "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 38
    },
    {
      "LastModified": "2015-11-09T23:26:41.000Z",
      "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
      "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 38
    },
    {
      "LastModified": "2015-11-09T22:50:50.000Z",
      "VersionId": "null",
      "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 533823
    }
  ]
}

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListObjectVersions](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [ListObjectVersions](#)。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

```
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [ListObjectVersions](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [ListObjectVersions](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListObjects** 与 CLI 配合使用

以下代码示例演示如何使用 ListObjects。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [创建列出 Amazon S3 对象的网页](#)

CLI

AWS CLI

以下示例使用 `list-objects` 命令显示指定存储桶中所有对象的名称：

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

该示例使用 `--query` 参数筛选 `list-objects` 的输出，使其范围缩小到每个对象的键值和大小。

有关对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“使用 Amazon S3 对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListObjects](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令检索有关存储桶“test-files”中所有项目的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

示例 2：此命令从存储桶“test-files”中检索有关项目“sample.txt”的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

示例 3：此命令从存储桶“test-files”中检索有关前缀为“sample”的所有项目的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [ListObjects](#)。

适用于 PowerShell V5 的工具

示例 1：此命令检索有关存储桶“test-files”中所有项目的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

示例 2：此命令从存储桶“test-files”中检索有关项目“sample.txt”的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

示例 3：此命令从存储桶“test-files”中检索有关前缀为“sample”的所有项目的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [ListObjects](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListObjectsV2** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListObjectsV2。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [删除存储桶中的所有对象](#)
- [对象存储入门](#)

.NET

适用于 .NET 的 SDK (v4)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [ListObjectsV2](#)。

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用分页工具列出对象。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [ListObjectsV2](#)。

Bash

AWS CLI 及 Bash 脚本

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListObjectsV2](#)。

C++

SDK for C++

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [ListObjectsV2](#)。

CLI

AWS CLI

获取存储桶中对象的列表

以下 list-objects-v2 示例列出了指定存储桶中的对象。

```
aws s3api list-objects-v2 \
    --bucket amzn-s3-demo-bucket
```

输出：

```
{
```

```
"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  
  }  
]  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ListObjectsV2](#)。

Go

适用于 Go 的 SDK V2

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [ListObjectsV2](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

```

使用分页列出对象。

```

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);
    }
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListObjectsV2](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出存储桶中的所有对象。如果有多个对象，则将使用 `IsTruncated` 和 `NextContinuationToken` 来迭代完整列表。

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
  list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
      objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
      console.log(
        `Page ${pageNum + 1}\n-----\n${objectList.map((o) => `•
        ${o}`)}.join("\n")\n`,
      );
    });
  } catch (err) {
    console.error(err);
  }
}
```

```
    );
  });
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while listing objects for "${bucketName}". The bucket
      doesn't exist.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while listing objects for "${bucketName}".
      ${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [ListObjectsV2](#)。

Kotlin

SDK for Kotlin

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

```

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    val response = s3.listObjects(request)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The object is ${myObject.size?.let { calKb(it) }} KBs")
        println("The owner is ${myObject.owner}")
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024

```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [ListObjectsV2](#)。

PHP

适用于 PHP 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出存储桶中的对象。

```

$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.");
}

```

```
}
```

- 有关 API 详细信息，请参阅《适用于 PHP 的 AWS SDK API Reference》中的 [ListObjectsV2](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [ListObjectsV2](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [ListObjectsV2](#)。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [ListObjectsV2](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [ListObjectsV2](#)。

Swift

适用于 Swift 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 [ListObjectsV2](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketAccelerateConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketAccelerateConfiguration。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketAccelerateConfiguration](#)。

CLI

AWS CLI

设置存储桶的加速配置

以下 `put-bucket-accelerate-configuration` 示例启用指定存储桶的加速配置。

```
aws s3api put-bucket-accelerate-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --accelerate-configuration Status=Enabled
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketAccelerateConfiguration](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令为给定 S3 存储桶启用传输加速。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketAccelerateConfiguration](#)。

适用于 PowerShell V5 的工具

示例 1：此命令为给定 S3 存储桶启用传输加速。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketAccelerateConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketAcl。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [管理访问控制列表 \(ACL\)](#)

.NET

适用于 .NET 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketAcl](#)。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [PutBucketAcl](#)。

CLI

AWS CLI

此示例向两个 AWS 用户 (user1@example.com 和 user2@example.com) 授予 full control，并向所有人授予 read 权限：

```
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
```

有关自定义 ACL (s3api ACL 命令 (如 put-bucket-acl) 使用相同的速记参数表示法) 的详细信息，请参阅 <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html>。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketAcl](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketAcl](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

放置存储桶 ACL。

```
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;
  }
}
};

```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-access-permissions.html#s3-example-access-permissions-put-acl>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [PutBucketAcl](#)。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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")
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [PutBucketAcl](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutBucketAcl](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutBucketAcl](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketCors** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketCors。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketCors](#)。

CLI

AWS CLI

以下示例启用来自 `www.example.com` 的 PUT、POST 和 DELETE 请求，并启用来自任何域的 GET 请求：

```
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
  }
]
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketCors](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketCors](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

添加 CORS 规则。

```
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;  
    }  
}  
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-configuring-buckets.html#s3-example-configuring-buckets-put-cors>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [PutBucketCors](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutBucketCors](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [PutBucketCors](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 [PutBucketCors](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketEncryption** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketEncryption。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [对象存储入门](#)

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketEncryption](#)。

CLI

AWS CLI

配置存储桶的服务器端加密

以下 put-bucket-encryption 示例将 AES256 加密设置为指定存储桶的默认值。

```
aws s3api put-bucket-encryption \
    --bucket amzn-s3-demo-bucket \
    --server-side-encryption-configuration '{"Rules":
[{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketEncryption](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令在指定的存储桶上启用具有 Amazon S3 托管式密钥的默认 AES256 服务器端加密 (SSE-S3)。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
    ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketEncryption](#)。

适用于 PowerShell V5 的工具

示例 1：此命令在指定的存储桶上启用具有 Amazon S3 托管式密钥的默认 AES256 服务器端加密 (SSE-S3)。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
    ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketEncryption](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketLifecycleConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketLifecycleConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [删除未完成的分段上传](#)
- [使用 S3 管理大型消息](#)

- [处理版本控制对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketLifecycleConfiguration](#)。

CLI

AWS CLI

以下命令将生命周期配置应用于名为 `amzn-s3-demo-bucket` 的存储桶：

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file:///lifecycle.json
```

文件 `lifecycle.json` 是当前文件夹中指定两个规则的 JSON 文档：

```
{  
  "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"  
    }  
  ]  
}
```

第一条规则在指定日期将带有前缀 `rotated` 的文件移动到 Glacier。第二条规则在旧对象版本不再是最新版本时将其移至 Glacier。有关可接受时间戳格式的信息，请参阅《AWS CLI 用户指南》中的“指定参数值”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketLifecycleConfiguration](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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 {

```

```
to the // Create a rule to archive objects with the "glacierobjects/" prefix

// 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);
    }
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketLifecycleConfiguration](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 [PutBucketLifecycleConfiguration](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutBucketLifecycleConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 PutBucketLogging 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketLogging。

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
    }
}
```

- 有关更多信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketLogging](#)。

CLI

AWS CLI

示例 1：设置存储桶策略日志记录

以下 put-bucket-logging 示例为 amzn-s3-demo-bucket 设置日志记录策略。首先，使用 put-bucket-policy 命令在存储桶策略中向日志记录服务主体授予权限。

```
aws s3api put-bucket-policy \
  --bucket amzn-s3-demo-bucket \
  --policy file://policy.json
```

的内容 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"}
      }
    }
  ]
}
```

要应用日志记录策略，请使用 `put-bucket-logging`。

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file://logging.json
```

的内容 `logging.json`：

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "amzn-s3-demo-bucket",  
    "TargetPrefix": "Logs/"  
  }  
}
```

向日志记录服务主体授予 `s3:PutObject` 权限需要使用 `put-bucket-policy` 命令。

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [Amazon S3 服务器访问日志记录](#)。

示例 2：设置仅向单个用户授予日志访问权限的存储桶策略

以下 `put-bucket-logging` 示例为 `amzn-s3-demo-bucket` 设置日志记录策略。AWS 用户 `bob@example.com` 将对日志文件具有完全控制权限，其他人没有任何访问权限。首先，使用 `put-bucket-acl` 授予 S3 权限。

```
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
```

然后，使用 `put-bucket-logging` 应用日志记录策略。

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file://logging.json
```

的内容 `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"
      }
    ]
  }
}

```

必须使用 `put-bucket-acl` 命令向 S3 的日志传输系统授予必要的权限 (`write-acp` 和 `read-acp` 权限)。

有关更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的 [Amazon S3 服务器访问日志记录](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketLogging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketNotification** 与 CLI 配合使用

以下代码示例演示如何使用 `PutBucketNotification`。

CLI

AWS CLI

将通知配置应用于名为 `amzn-s3-demo-bucket` 的存储桶：

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

文件 `notification.json` 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要监控的事件类型：

```
{
```

```
"TopicConfiguration": {
  "Event": "s3:ObjectCreated:*",
  "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
}
```

SNS 主题必须附加一个 IAM 策略，以允许 Amazon S3 向其发布：

```
{
  "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"
        }
      }
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketNotification](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例为 S3 事件 ObjectRemovedDelete 配置 SNS 主题配置，并为给定的 s3 存储桶启用通知

```
$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

```

示例 2：此示例为给定存储桶启用 ObjectCreatedAll 的通知，将其发送到 Lambda 函数。

```

$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

```

示例 3：此示例基于不同的键后缀创建 2 个不同的 Lambda 配置，并在单个命令中配置了这两个配置。

```

#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

```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketNotification](#)。

适用于 PowerShell V5 的工具

示例 1：此示例为 S3 事件 ObjectRemovedDelete 配置 SNS 主题配置，并为给定的 s3 存储桶启用通知

```

$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

```

示例 2：此示例为给定存储桶启用 ObjectCreatedAll 的通知，将其发送到 Lambda 函数。

```

$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

```

示例 3：此示例基于不同的键后缀创建 2 个不同的 Lambda 配置，并在单个命令中配置了这两个配置。

```

#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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketNotification](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketNotificationConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketNotificationConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [处理 S3 事件通知](#)
- [向 EventBridge 发送事件通知](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketNotificationConfiguration](#)。

CLI

AWS CLI

启用存储桶的指定通知

以下 `put-bucket-notification-configuration` 示例将通知配置应用于名为 `amzn-s3-demo-bucket` 的存储桶。文件 `notification.json` 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要监控的事件类型。

```
aws s3api put-bucket-notification-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --notification-configuration file://notification.json
```

的内容 `notification.json`：

```
{  
  "TopicConfigurations": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-  
topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

SNS 主题必须附加一个 IAM 策略，以允许 Amazon S3 向其发布。

```
{  
  "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"  
        }  
      }  
    }  
  ]  
}
```

```

    }
  }
}
]
}

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketNotificationConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketPolicy** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketPolicy。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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"
        "}";
}

```

```
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [PutBucketPolicy](#)。

CLI

AWS CLI

此示例允许所有用户检索 amzn-s3-demo-bucket 中的任何对象，但 MySecretFolder 中的对象除外。它还向 AWS 账户 1234-5678-9012 的根用户授予 put 和 delete 权限：

```
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/*"
    }
  ]
}
```

```
]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketPolicy](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketPolicy](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

添加策略。

```
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;
  }
}
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-bucket-policies.html#s3-example-bucket-policies-set-policy>。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [PutBucketPolicy](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutBucketPolicy](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [PutBucketPolicy](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutBucketPolicy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketReplication** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketReplication。

CLI

AWS CLI

为 S3 存储桶配置复制

以下 put-bucket-replication 示例将复制配置应用于指定的 S3 存储桶。

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

的内容 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"  
      }  
    }  
  ]  
}
```

目标存储桶必须已启用版本控制。指定的角色必须具有写入目标存储桶的权限，并且必须建立允许 Amazon S3 代入角色的信任关系。

示例角色权限策略：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetReplicationConfiguration",  
        "s3:ListBucket"  
      ],  
      "Resource": "*"   
    }  
  ]  
}
```

```

        "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/*"
    }
]
}

```

示例信任关系策略：

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Simple Storage Service 控制台用户指南》中的[这是主题标题](#)：

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketReplication](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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



```
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketReplication](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例设置具有单个规则的复制配置，支持将存储桶“amzn-s3-demo-bucket”中使用键名称前缀“TaxDocs”创建的任何新对象复制到“amzn-s3-demo-bucket”存储桶。

```
$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
```

示例 2：此示例设置具有多个规则的复制配置，支持将使用键名称前缀“TaxDocs”或“OtherDocs”创建的任何新对象复制到“amzn-s3-demo-bucket”存储桶。键前缀不得重叠。

```
$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

```

示例 3：此示例更新指定存储桶上的复制配置，以禁用用于控制将键名称前缀为“TaxDocs”的对象复制到存储桶“amzn-s3-demo-bucket”的规则。

```

$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

```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketReplication](#)。

适用于 PowerShell V5 的工具

示例 1：此示例设置具有单个规则的复制配置，支持将存储桶“amzn-s3-demo-bucket”中使用键名称前缀“TaxDocs”创建的任何新对象复制到“amzn-s3-demo-bucket”存储桶。

```

$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

```

示例 2：此示例设置具有多个规则的复制配置，支持将使用键名称前缀“TaxDocs”或“OtherDocs”创建的任何新对象复制到“amzn-s3-demo-bucket”存储桶。键前缀不得重叠。

```

$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

```

示例 3：此示例更新指定存储桶上的复制配置，以禁用用于控制将键名称前缀为“TaxDocs”的对象复制到存储桶“amzn-s3-demo-bucket”的规则。

```
$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
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketReplication](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketRequestPayment** 与 CLI 配合使用

以下代码示例演示如何使用 PutBucketRequestPayment。

CLI

AWS CLI

示例 1：为存储桶启用“申请方付款”配置

以下 put-bucket-request-payment 示例为指定的存储桶启用 requester pays。

```
aws s3api put-bucket-request-payment \
    --bucket amzn-s3-demo-bucket \
    --request-payment-configuration '{"Payer": "Requester"}
```

此命令不生成任何输出。

示例 2：为存储桶禁用“申请方付款”配置

以下 `put-bucket-request-payment` 示例为指定的存储桶禁用 `requester pays`。

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer":"BucketOwner"}'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketRequestPayment](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：更新名为“amzn-s3-demo-bucket”的存储桶的请求付款配置，以便向从该存储桶请求下载的人员收取下载费用。默认情况下，存储桶拥有者支付下载的费用。要将请求付款设置回默认值，请将“BucketOwner”用于 `RequestPaymentConfiguration_Payer` 参数。

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketRequestPayment](#)。

适用于 PowerShell V5 的工具

示例 1：更新名为“amzn-s3-demo-bucket”的存储桶的请求付款配置，以便向从该存储桶请求下载的人员收取下载费用。默认情况下，存储桶拥有者支付下载的费用。要将请求付款设置回默认值，请将“BucketOwner”用于 `RequestPaymentConfiguration_Payer` 参数。

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketRequestPayment](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketTagging** 与 CLI 配合使用

以下代码示例演示如何使用 PutBucketTagging。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [对象存储入门](#)

CLI

AWS CLI

以下命令将标记配置应用于名为 amzn-s3-demo-bucket 的存储桶：

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://tagging.json
```

文件 tagging.json 是当前文件夹中指定标签的 JSON 文档：

```
{
  "TagSet": [
    {
      "Key": "organization",
      "Value": "marketing"
    }
  ]
}
```

或者，直接从命令行将标记配置应用于 amzn-s3-demo-bucket：

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging 'TagSet=[{Key=organization,Value=marketing}]'
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketTagging](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令将两个标签应用于名为 **cloudtrail-test-2018** 的存储桶：一个标签的键为 Stage，值为 Test；另一个标签的键为 Environment，值为 Alpha。要验证标签已添加到存储桶，请运行 **Get-S3BucketTagging -BucketName bucket_name**。结果应显示您在第一个命令中应用于存储桶的标签。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。要添加或删除各标签，请运行资源组和标记 API cmdlet **Add-RGTResourceTag** 和 **Remove-RGTResourceTag**。或者，使用 AWS 管理控制台中的标签编辑器来管理 S3 存储桶标签。

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

示例 2：此命令将名为 **cloudtrail-test-2018** 的存储桶传送到 **Write-S3BucketTagging** cmdlet。它将标签 Stage:Production 和 Department:Finance 应用于存储桶。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketTagging](#)。

适用于 PowerShell V5 的工具

示例 1：此命令将两个标签应用于名为 **cloudtrail-test-2018** 的存储桶：一个标签的键为 Stage，值为 Test；另一个标签的键为 Environment，值为 Alpha。要验证标签已添加到存储桶，请运行 **Get-S3BucketTagging -BucketName bucket_name**。结果应显示您在第一个命令中应用于存储桶的标签。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。要添加或删除各标签，请运行资源组和标记 API cmdlet **Add-RGTResourceTag** 和 **Remove-RGTResourceTag**。或者，使用 AWS 管理控制台中的标签编辑器来管理 S3 存储桶标签。

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

示例 2：此命令将名为 **cloudtrail-test-2018** 的存储桶传送到 **Write-S3BucketTagging** cmdlet。它将标签 Stage:Production 和 Department:Finance 应用于存储桶。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketVersioning** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketVersioning。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [对象存储入门](#)

CLI

AWS CLI

以下命令对于名为 amzn-s3-demo-bucket 的存储桶启用版本控制。

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled
```

以下命令启用版本控制，并使用 mfa 代码

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled --mfa "SERIAL 123456"
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketVersioning](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketVersioning](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令为给定 S3 存储桶启用版本控制。

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketVersioning](#)。

适用于 PowerShell V5 的工具

示例 1：该命令为给定 S3 存储桶启用版本控制。

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketVersioning](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutBucketVersioning](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketWebsite** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketWebsite。

.NET

适用于 .NET 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// Put the website configuration.  
PutBucketWebsiteRequest putRequest = new  
PutBucketWebsiteRequest()  
{  
    BucketName = bucketName,  
    WebsiteConfiguration = new WebsiteConfiguration()  
    {  
        IndexDocumentSuffix = indexDocumentSuffix,  
        ErrorDocument = errorDocument,  
    },  
};  
PutBucketWebsiteResponse response = await  
client.PutBucketWebsiteAsync(putRequest);
```

- 有关更多信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutBucketWebsite](#)。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();  
}
```

- 有关更多信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [PutBucketWebsite](#)。

CLI

AWS CLI

将静态网站配置应用于名为 `amzn-s3-demo-bucket` 的存储桶：

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-  
configuration file://website.json
```

文件 `website.json` 是当前文件夹中的一个 JSON 文档，用于指定网站的索引和错误页面：

```
{  
  "IndexDocument": {  
    "Suffix": "index.html"  
  },  
  "ErrorDocument": {  
    "Key": "error.html"  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutBucketWebsite](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
    }
}
```

```
}
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketWebsite](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

设置网站配置。

```
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;
    }
}
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-static-web-host.html#s3-example-static-web-host-set-website>。
- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [PutBucketWebsite](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令为给定存储桶启用网站托管，索引文档为“index.html”，错误文档为“error.html”。

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutBucketWebsite](#)。

适用于 PowerShell V5 的工具

示例 1：该命令为给定存储桶启用网站托管，索引文档为“index.html”，错误文档为“error.html”。

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutBucketWebsite](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website
```

```

# @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
configured with an existing bucket.
def initialize(bucket_website)
  @bucket_website = bucket_website
end

# Sets a bucket as a static website.
#
# @param index_document [String] The name of the index document for the
website.
# @param error_document [String] The name of the error document to show for 4XX
errors.
# @return [Boolean] True when the bucket is configured as a website; otherwise,
false.
def set_website(index_document, error_document)
  @bucket_website.put(
    website_configuration: {
      index_document: { suffix: index_document },
      error_document: { key: error_document }
    }
  )
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
why: #{e.message}"
  false
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK API 参考》中的 [PutBucketWebsite](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [对象存储入门](#)
- [提出条件请求](#)
- [跟踪上传和下载操作](#)
- [使用 Amazon S3 对象完整性](#)

.NET

适用于 .NET 的 SDK (v4)

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [PutObject](#)。

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用服务器端加密上传对象。

```
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");
    }
}
}

```

使用条件请求放置对象。

```

/// <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;
}
}
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的 [PutObject](#)。

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。


```
#####
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutObject](#)。

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();  
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考中的 [PutObject](#)。

CLI

AWS CLI

示例 1：将对象上传到 Amazon S3

以下 put-object 命令示例将对象上传到 Amazon S3。

```
aws s3api put-object \  
    --bucket amzn-s3-demo-bucket \  
    --key my-dir/MySampleImage.png \  
    --body MySampleImage.png
```

有关上传对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“上传对象”<<http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>>。

示例 2：将视频文件上传到 Amazon S3

以下 put-object 命令示例上传视频文件。

```
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
```

有关上传对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“上传对象”<<http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>>。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutObject](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用低级别 API 将对象放入存储桶。

```
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
}

```

使用传输管理器将对象上传到存储桶。

```

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  
}
```

- 有关 API 详细信息，请参阅 [适用于 Go 的 AWS SDK API 参考](#) 中的 PutObject。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 [S3Client](#) 将文件上传到存储桶。

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

使用 [S3TransferManager](#) 将文件上传到存储桶。查看[完整文件](#)并[进行测试](#)。

```
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();
}
```

使用 [S3Client](#) 将对象上传到存储桶并设置标签。


```
/**
 * 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;
    }
}

```

使用 [S3Client](#) 将对象上传到存储桶并设置元数据。

```

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);
    }
}
}

```

使用 [S3Client](#) 将对象上传到存储桶并设置对象保留值。

```

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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutObject](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

上传对象。

```
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;
    }
  }
};
```

在对象的 ETag 与提供的 ETag 匹配的条件下上传该对象。

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

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-creating-buckets.html#s3-example-creating-buckets-new-bucket-2>。
- 有关 API 详细信息，请参阅 [适用于 JavaScript 的 AWS SDK API 参考](#) 中的 PutObject。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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}")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [PutObject](#)。

PHP

适用于 PHP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

将对象上传到存储桶。

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

- 有关 API 的详细信息，请参阅 适用于 PHP 的 AWS SDK API 参考中的 [PutObject](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令将单个文件“local-sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

示例 2：此命令将单个文件“sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。如果未提供 -Key 参数，则文件名将用作 S3 对象键。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

示例 3：此命令将单个文件“local-sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“prefix/to/sample.txt”的对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

示例 4：此命令将子目录“Scripts”中的所有文件上传到存储桶“test-files”，并将公用键前缀“SampleScripts”应用于每个对象。每个上传的文件都将有一个键“SampleScripts/filename”，其中“filename”不同。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

示例 5：此命令将本地目录“Scripts”中的所有 *.ps1 文件上传到存储桶“test-files”，并将公用键前缀“SampleScripts”应用于每个对象。每个上传的文件都将有一个键“SampleScripts/filename.ps1”，其中“filename”不同。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

示例 6：此命令创建一个新的 S3 对象，其中包含键为“sample.txt”的指定内容字符串。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

示例 7：此命令上传指定的文件（文件名用作键），并将指定的标签应用于新对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

示例 8：此命令以递归方式上传指定的文件夹，并将指定的标签应用于所有新对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutObject](#)。

适用于 PowerShell V5 的工具

示例 1：此命令将单个文件“local-sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-sample.txt
```

示例 2：此命令将单个文件“sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。如果未提供 -Key 参数，则文件名将用作 S3 对象键。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

示例 3：此命令将单个文件“local-sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“prefix/to/sample.txt”的对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .\local-sample.txt
```

示例 4：此命令将子目录“Scripts”中的所有文件上传到存储桶“test-files”，并将公用键前缀“SampleScripts”应用于每个对象。每个上传的文件都将有一个键“SampleScripts/filename”，其中“filename”不同。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix SampleScripts\
```

示例 5：此命令将本地目录“Scripts”中的所有 *.ps1 文件上传到存储桶“test-files”，并将公用键前缀“SampleScripts”应用于每个对象。每个上传的文件都将有一个键“SampleScripts/filename.ps1”，其中“filename”不同。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix SampleScripts\ -SearchPattern *.ps1
```

示例 6：此命令创建一个新的 S3 对象，其中包含键为“sample.txt”的指定内容字符串。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object contents"
```

示例 7：此命令上传指定的文件（文件名用作键），并将指定的标签应用于新对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

示例 8：此命令以递归方式上传指定的文件夹，并将指定的标签应用于所有新对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutObject](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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()

```

使用条件请求上传对象。

```

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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutObject](#)。

Ruby

适用于 Ruby 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用托管上传工具 (Object.upload_file) 上传文件。

```
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__
```

使用 `Object.put` 上传文件。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
    #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
```

```
return unless success

puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 `Object.put` 上传文件并添加服务器端加密。

```
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
```

```
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅 [适用于 Ruby 的 AWS SDK API 参考](#) 中的 PutObject。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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)
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [PutObject](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
"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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutObject](#)。

Swift

适用于 Swift 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
import AWSS3
import Smithy

    public fun 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 fun 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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》中的 [PutObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectAcl。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [管理访问控制列表 \(ACL\)](#)

C++

SDK for C++

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
    }
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [PutObjectAcl](#)。

CLI

AWS CLI

以下命令向两个 AWS 用户 (user1@example.com 和 user2@example.com) 授予 full control，并向所有人授予 read 权限：

```
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
```

有关自定义 ACL (s3api ACL 命令 (如 put-object-acl) 使用相同的速记参数表示法) 的详细信息，请参阅 <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html>。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutObjectAcl](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutObjectAcl](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutObjectAcl](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectLegalHold** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectLegalHold。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [锁定 Amazon S3 对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/// <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;
        }
    }
}

```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutObjectLegalHold](#)。

CLI

AWS CLI

对对象应用法定保留

以下 put-object-legal-hold 示例在 doc1.rtf 对象上设置了法定保留。

```
aws s3api put-object-legal-hold \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --legal-hold Status=ON
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutObjectLegalHold](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [PutObjectLegalHold](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// 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 +".");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutObjectLegalHold](#)。

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub , 了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例, 了解如何进行设置和运行。

```
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"));
    }
  }
}

```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [PutObjectLegalHold](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

放置对象法定保留。

```

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
    )
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutObjectLegalHold](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutObjectLegalHold](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectLockConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectLockConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [锁定 Amazon S3 对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

设置存储桶的对象锁定配置。

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

设置存储桶的默认保留期。

```
/// <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($"{bucketName}");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"{ex.Message}");
        return false;
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutObjectLockConfiguration](#)。

CLI

AWS CLI

在存储桶上设置对象锁定配置

以下 put-object-lock-configuration 示例在指定存储桶上设置了 50 天的对象锁定。

```
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 } } }'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutObjectLockConfiguration](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

设置存储桶的对象锁定配置。

```
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
}

```

设置存储桶的默认保留期。

```

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
}

```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [PutObjectLockConfiguration](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

设置存储桶的对象锁定配置。

```
// 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()
+ "'");
    }
}
```

```
    }  
}
```

设置存储桶的默认保留期。

```
// 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
+ ".");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutObjectLockConfiguration](#)。

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

设置存储桶的对象锁定配置。

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

设置存储桶的默认保留期。

```
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.
          },
        },
      })
    );
  } catch (err) {
    if (err instanceof S3ServiceException) {
      console.error(err.message);
    } else {
      console.error(err);
    }
  }
}
```

```

        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"));
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [PutObjectLockConfiguration](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

放置对象锁定配置。

```
s3_client.put_object_lock_configuration(  
    Bucket=bucket,  
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":  
{}},  
)
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutObjectLockConfiguration](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutObjectLockConfiguration](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectRetention** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectRetention。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [锁定 Amazon S3 对象](#)

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [PutObjectRetention](#)。

CLI

AWS CLI

为对象设置对象保留配置

以下 put-object-retention 示例为指定对象设置直到 2025 年 1 月 1 日的对象保留配置。

```
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" }'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [PutObjectRetention](#)。

Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的 [PutObjectRetention](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// 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 +".");
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutObjectRetention](#)。

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [PutObjectRetention](#)。

PowerShell

适用于 PowerShell V4 的工具

示例 1：该命令为给定 S3 存储桶中的“testfile.txt”对象启用监管保留模式，直到日期“2019 年 12 月 31 日 00:00:00”。

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [PutObjectRetention](#)。

适用于 PowerShell V5 的工具

示例 1：该命令为给定 S3 存储桶中的“testfile.txt”对象启用监管保留模式，直到日期“2019 年 12 月 31 日 00:00:00”。

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V5)》中的 [PutObjectRetention](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

放置对象保留。

```
s3_client.put_object_retention(
```

```

        Bucket=bucket,
        Key=key,
        VersionId=version_id,
        Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
        BypassGovernanceRetention=True,
    )

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutObjectRetention](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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.

```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutObjectRetention](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **RestoreObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `RestoreObject`。

.NET

适用于 .NET 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }
}
```

```

    }

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// RestoreObjectAsync.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object was located before it was archived.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object to restore.</param>
    public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
    {
        try
        {
            var restoreRequest = new RestoreObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Days = 2,
            };
            RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

            // Check the status of the restoration.
            await CheckRestorationStatusAsync(client, bucketName, objectKey);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Error: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// This method retrieves the status of the object's restoration.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectMetadataAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
    /// S3 bucket which contains the archived object.</param>
    /// <param name="objectKey">A string representing the name of the

```



```
/// archived object you want to restore.</param>
public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
{
    GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
    };

    GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

    var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
    Console.WriteLine($"Restoration status: {restStatus}");
}
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的 [RestoreObject](#)。

CLI

AWS CLI

为对象创建还原请求

以下 `restore-object` 示例将存储桶 `my-glacier-bucket` 的指定 Amazon S3 Glacier 对象还原为 10 天。

```
aws s3api restore-object \
  --bucket my-glacier-bucket \
  --key doc1.rtf \
  --restore-request Days=10
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [RestoreObject](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [RestoreObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **SelectObjectContent** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 SelectObjectContent。

CLI

AWS CLI

基于 SQL 语句筛选 Amazon S3 对象的内容

以下 select-object-content 示例使用指定的 SQL 语句筛选 my-data-file.csv 对象并将输出发送到文件。

```
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"
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [SelectObjectContent](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

以下示例演示了一个使用 JSON 对象的查询。该[完整示例](#)还演示了 CSV 对象的用法。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
    software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
```

```
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
        SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where
        c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
            .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
            .compressionType(CompressionType.NONE)
            .build();
    }
}
```

```

        OutputSerialization outputSerialization = OutputSerialization.builder()
            .json(JSONOutput.builder().recordDelimiter(null).build())
            .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
            .bucket(BUCKET_NAME)
            .key(FILE_JSON)
            .expression(queryExpression)
            .expressionType(ExpressionType.SQL)
            .inputSerialization(inputSerialization)
            .outputSerialization(outputSerialization)
            .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
        handler.
        // Supply an EventStreamInfo object to the response handler to gather
        records and information from the response.
        s3AsyncClient.selectObjectContent(select,
            buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
            record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
    buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
        information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
        SelectObjectContentResponseHandler.Visitor.builder()

```

```

        .onRecords(r -> {
            logger.info("Record event received.");
            eventStreamInfo.addRecord(r.payload().asUtf8String());
            eventStreamInfo.incrementOnRecordsCalled();
        })
        .onCont(ce -> {
            logger.info("Continuation event received.");
            eventStreamInfo.incrementContinuationEvents();
        })
        .onProgress(pe -> {
            Progress progress = pe.details();
            logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
                progress.bytesScanned(),
                progress.bytesProcessed(),
                progress.bytesReturned());
        })
        .onEnd(ee -> logger.info("End event received.))
        .onStats(se -> {
            logger.info("Stats event received.");
            eventStreamInfo.addStats(se.details());
        })
        .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
    // processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
// processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }
}

```



```
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 [SelectObjectContent](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UploadPart** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 UploadPart。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [使用校验和](#)
- [使用 Amazon S3 对象完整性](#)

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

//! 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);
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的 [UploadPart](#)。

CLI

AWS CLI

以下命令上传使用 create-multipart-upload 命令启动的分段上传中的第一个分段：

```

aws s3api upload-part --bucket amzn-s3-demo-bucket --key
'multipart/01' --part-number 1 --body part01 --upload-id
'dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P'

```

body 选项采用本地文件的名称或路径进行上传（不要使用 file:// 前缀）。最小分段大小为 5 MB。上传 ID 由 create-multipart-upload 返回，也可以使用 list-multipart-uploads 进行检索。存储桶和键是在您创建分段上传时指定的。

输出：

```
{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

保存每个分段的 ETag 值以备后用。需要这些值才能完成分段上传。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [UploadPart](#)。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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?;
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 [UploadPart](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UploadPartCopy** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 UploadPartCopy。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [执行分段复制](#)

CLI

AWS CLI

从现有对象复制数据作为数据来源，上传部分对象

以下 upload-part-copy 示例会通过从现有对象复制数据作为数据来源，上传部分对象。

```
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"
```

输出：

```
{
  "CopyPartResult": {
    "LastModified": "2019-12-13T23:16:03.000Z",
```

```
    "ETag": "\"711470fc377698c393d94aed6305e245\""
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [UploadPartCopy](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [UploadPartCopy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

适用于使用 AWS 软件开发工具包的 Amazon S3 的场景

以下代码示例显示如何通过 AWS SDK 实施 Amazon S3 中的常见场景。这些场景向您展示了如何通过调用 Amazon S3 中的多个函数或与其它 AWS 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- [检查存储桶是否存在](#)
- [检查对象是否存在](#)
- [使用 AWS SDK 将文本转换为语音以及将语音转换回文本](#)
- [使用 AWS SDK 创建 Amazon S3 的预签名 URL](#)
- [创建照片资产管理应用程序，让用户能够使用标签管理照片](#)
- [使用 AWS SDK 列出 Amazon S3 对象的网页](#)
- [创建 Amazon Textract 浏览器应用程序](#)
- [使用 AWS SDK 删除给定 Amazon S3 存储桶中的所有对象](#)

- [使用 AWS SDK 删除未完成的 Amazon S3 分段上传](#)
- [使用 AWS SDK 通过 Amazon Rekognition 检测图像中的 PPE](#)
- [使用 AWS SDK 检测从图像中提取的文本中的实体](#)
- [使用 AWS SDK 检测图像中的人脸](#)
- [使用 AWS 软件开发工具包通过 Amazon Rekognition 检测图像中的对象](#)
- [使用 AWS SDK 通过 Amazon Rekognition 检测视频中的人物和对象](#)
- [从 Amazon Simple Storage Service \(Amazon S3\) 存储桶下载 S3“目录”](#)
- [将 Amazon Simple Storage Service \(Amazon S3 \) 桶中的所有对象下载到本地目录](#)
- [使用 AWS SDK 从 Amazon S3 对象下载未知大小的流](#)
- [使用 AWS SDK 从多区域接入点中获取 Amazon S3 对象](#)
- [使用 AWS SDK 并指定 If-Modified-Since 标头以从 Amazon S3 存储桶获取对象](#)
- [使用 AWS SDK 加密 Amazon S3 对象入门](#)
- [使用 AWS SDK 为 Amazon S3 存储桶和对象加标签入门](#)
- [大数据处理集群入门](#)
- [配置管理入门](#)
- [文档文本提取入门](#)
- [机器学习特征存放区入门](#)
- [对象存储入门](#)
- [查询分析入门](#)
- [通过 AWS SDK 使用 Amazon S3 对象锁定功能](#)
- [使用 AWS SDK 发起 Amazon S3 条件请求](#)
- [使用 AWS SDK 管理 Amazon S3 存储桶的访问控制列表 \(ACL \)](#)
- [使用 AWS SDK 的 Amazon S3 管理大型 Amazon SQS 消息](#)
- [通过 AWS SDK 使用 Lambda 函数批量管理版本控制的 Amazon S3 对象](#)
- [使用 AWS SDK 解析 Amazon S3 URI](#)
- [使用 AWS SDK 执行 Amazon S3 对象的分段复制](#)
- [使用 AWS SDK 接收和处理 Amazon S3 事件通知](#)
- [使用 AWS SDK 保存 EXIF 和其他图像信息](#)
- [使用 AWS SDK 向 Amazon EventBridge 发送 S3 事件通知](#)
- [跟踪使用 AWS SDK 执行的 Amazon S3 对象上传或下载操作](#)

- [使用 S3 对象 Lambda 转换应用程序的数据](#)
- [使用 AWS SDK 进行单元测试和集成测试的示例方法](#)
- [以递归方式将本地目录上传到 Amazon Simple Storage Service \(Amazon S3 \) 桶](#)
- [借助 AWS SDK 向 Amazon S3 上传大文件或从 Amazon S3 下载大文件](#)
- [使用 AWS SDK 将未知大小的流上传到 Amazon S3 对象](#)
- [使用 AWS 开发工具包，对 Amazon S3 对象使用校验和](#)
- [通过 AWS SDK 使用 Amazon S3 对象完整性功能](#)
- [使用 AWS SDK 使用 Amazon S3 版本控制对象](#)

检查存储桶是否存在

以下代码示例演示如何检查存储桶是否存在。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

可以使用以下 `doesBucketExists` 方法来替代适用于 Java 的 SDK V1 [AmazonS3Client#doesBucketExistV2\(String\)](#) 方法。

```
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>
     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.getStatusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
            "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
            return true;
        }
        if (ase.getStatusCode() == HttpStatusCode.NOT_FOUND) {
            return false;
        }
        throw ase;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetBucketAcl](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

检查对象是否存在

以下代码示例演示如何检查对象是否存在。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

可以使用以下 `doesObjectExist` 方法来替代适用于 Java 的 SDK V1 [AmazonS3Client#doesObjectExist\(String, String\)](#) 方法。

```
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;
        }
    }
}
```

- 有关 API 详细，请参阅《AWS SDK for Java 2.x API 参考》中的 [HeadObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 将文本转换为语音以及将语音转换回文本

以下代码示例展示了如何：

- 使用 Amazon Polly 将纯文本 (UTF-8) 输入文件合成为音频文件。
- 将音频文件上传到 Amazon S3 存储桶。
- 使用 Amazon Transcribe 将音频文件转换为文本。
- 显示文本。

Rust

适用于 Rust 的 SDK

使用 Amazon Polly 将纯文本 (UTF-8) 输入文件合成为音频文件，将音频文件上传到 Amazon S3 存储桶，使用 Amazon Transcribe 将该音频文件转换为文本，然后显示文本。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Polly
- Amazon S3
- Amazon Transcribe

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 创建 Amazon S3 的预签名 URL

以下代码示例显示如何为 Amazon S3 创建预签名 URL 以及如何上传对象。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

生成可在有限时间内执行 Amazon S3 操作的预签名 URL。

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

public class GenPresignedUrl
{
    public static void Main()
```

```

{
    const string bucketName = "amzn-s3-demo-bucket";
    const string objectKey = "sample.txt";

    // Specify how long the presigned URL lasts, in hours
    const double timeoutDuration = 12;

    // Specify the AWS Region of your Amazon S3 bucket. If it is
    // different from the Region defined for the default user,
    // pass the Region to the constructor for the client. For
    // example: new AmazonS3Client(RegionEndpoint.USEast1);

    // If using the Region us-east-1, and server-side encryption with AWS
    KMS, you must specify Signature Version 4.
    // Region us-east-1 defaults to Signature Version 2 unless explicitly
    set to Version 4 as shown below.
    // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
    // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
    AWSConfigsS3.UseSignatureVersion4 = true;
    IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

    string urlString = GeneratePresignedURL(s3Client, bucketName,
objectKey, timeoutDuration);
    Console.WriteLine($"The generated URL is: {urlString}.");
}

/// <summary>
/// Generate a presigned URL that can be used to access the file named
/// in the objectKey parameter for the amount of time specified in the
/// duration parameter.
/// </summary>
/// <param name="client">An initialized S3 client object used to call
/// the GetPresignedUrl method.</param>
/// <param name="bucketName">The name of the S3 bucket containing the
/// object for which to create the presigned URL.</param>
/// <param name="objectKey">The name of the object to access with the
/// presigned URL.</param>
/// <param name="duration">The length of time for which the presigned
/// URL will be valid.</param>
/// <returns>A string representing the generated presigned URL.</returns>
public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)

```



```
{
    string urlString = string.Empty;
    try
    {
        var request = new GetPreSignedUrlRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            Expires = DateTime.UtcNow.AddHours(duration),
        };
        urlString = client.GetPreSignedURL(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: '{ex.Message}'");
    }

    return urlString;
}
}
```

生成预签名 URL 并使用该 URL 执行上传。

```
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;
    }
}
```

适用于 .NET 的 SDK (v4)

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建并使用预签名 POST URL 来进行直接浏览器上传。

```
/// <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 for C++

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

生成预签名 URL 来下载对象。

```
//! 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);
}

```

使用 libcurl 下载。

```

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::getObjectWithPresignedGetObjectUrl(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;
}
```

生成预签名 URL 来上传对象。

```

//! 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);
}

```

使用 libcurl 上传。

```

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

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建包装 S3 预签名操作的函数。

```
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
}

```

运行一个交互式示例，以生成并使用预签名 URL 上传、下载和删除 S3 对象。

```

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))
}
```

定义示例用于发出 HTTP 请求的 HTTP 请求包装器。

```
// 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

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

下面演示如何创建预签名 URL 并将 URL 与 HTTP 客户端库结合使用的三个示例：

- 一个 HTTP GET 请求，它将此 URL 与三个 HTTP 客户端库结合使用
- 一个在标头中包含元数据的 HTTP PUT 请求，它将此 URL 与三个 HTTP 客户端库结合使用
- 一个带有查询参数的 HTTP PUT 请求，它将此 URL 与一个 HTTP 客户端库结合使用

为对象生成预签名 URL，然后下载该 URL (GET 请求)。

导入。

```
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;
```

生成 URL。

```
/* Create a pre-signed URL to download an object in a subsequent GET request.
*/
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
            GetObjectPresignRequest.builder()
```

```

        .signatureDuration(Duration.ofMinutes(10)) // The URL will
        expire in 10 minutes.
        .getObjectRequest(objectRequest)
        .build();

        PresignedGetObjectRequest presignedRequest =
        presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]",
        presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
        presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

使用以下三种方法中的任何一种下载对象。

使用 JDK `URLConnection` (自 v1.1 起) 类进行下载。

```

/* 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();
}

```

使用 JDK HttpClient (自 v11 起) 类进行下载。

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

使用 AWS SDK for Java SdkHttpClient 类进行下载。

```
/* 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();
    }
}

```

为进行上传生成在标头中带有元数据的预签名 URL，然后上传文件（PUT 请求）。

导入。

```

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;
```

生成 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();
    }
}

```

使用以下三种方法中的任何一种上传文件对象。

使用 JDK `URLConnection` (自 v1.1 起) 类进行上传。

```

/* 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);
    }
}

```

使用 JDK `HttpClient` (自 v11 起) 类进行上传。

```

/* 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);
    }
}

```

使用 AWS for Java V2 `SdkHttpClient` 类进行上传。


```

/* 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);
    }
}

```

为进行上传生成带有查询参数的预签名 URL，然后上传文件（PUT 请求）。

导入。

```

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;

```

生成 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();
    }
}

```

使用 AWS for Java V2 SdkHttpClient 类进行上传。

```

/**
 * 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 for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建预签名 URL 以将对象上传到存储桶。

```
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;
    }
  }
};

```

创建预签名 URL 以从存储桶下载对象。

```

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;
    }
}
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/s3-example-creating-buckets.html#s3-create-presigendurl>。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建 GetObject 预签名请求并使用 URL 下载对象。

```
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
}
```

使用高级选项创建 `GetObject` 预签名请求。

```
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
}
```

创建 `PutObject` 预签名请求并使用它上传对象。

```
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)
}
```

- 有关更多信息，请参阅 [AWS SDK for Kotlin 开发人员指南](#)。

PHP

适用于 PHP 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

生成可在有限时间内执行 S3 操作的预签名 URL。使用请求软件包通过 URL 发出请求。

```
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()

```

生成预签名 POST 请求以上传文件。

```

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

适用于 Ruby 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建预签名请求来 GET S3 对象。

```
/// 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(())
    }

```

创建预签名请求来 PUT S3 对象。

```

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

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建预签名请求来 GET S3 对象。

```
" 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( ).
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建照片资产管理应用程序，让用户能够使用标签管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

.NET

适用于 .NET 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

要深入了解这个例子的起源，请参阅 [AWS 社区](#) 上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK for C++

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

要深入了解这个例子的起源，请参阅 [AWS 社区](#) 上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

适用于 Java 的 SDK 2.x

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

要深入了解这个例子的起源，请参阅 [AWS 社区](#) 上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

适用于 JavaScript 的 SDK (v3)

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

要深入了解这个例子的起源，请参阅 [AWS 社区](#) 上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK for Kotlin

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

要深入了解这个例子的起源，请参阅 [AWS 社区](#) 上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

适用于 PHP 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

要深入了解这个例子的起源，请参阅 [AWS 社区](#) 上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

适用于 Rust 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

要深入了解这个例子的起源，请参阅 [AWS 社区](#) 上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 列出 Amazon S3 对象的网页

以下代码示例显示如何在网页中列出 Amazon S3 对象。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

以下代码是对 AWS SDK 进行调用的相关 React 组件。可以在前面的 GitHub 链接中找到包含此组件的应用程序的可运行版本。

```
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;
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的 [ListObjects](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Amazon Textract 浏览器应用程序

以下代码示例展示如何通过交互式应用程序探索 Amazon Textract 输出。

JavaScript

SDK for JavaScript (v3)

演示了如何使用 适用于 JavaScript 的 AWS SDK 构建 React 应用程序，该应用程序使用 Amazon Textract 从文档图像中提取数据并在交互式网页中显示该数据。此示例在 Web 浏览器中运行，需要经过身份验证的 Amazon Cognito 身份才能获得凭证。它使用 Amazon Simple Storage Service (Amazon S3) 进行存储；对于通知，它将轮询订阅 Amazon Simple Notification Service (Amazon SNS) 主题的 Amazon Simple Queue Service (Amazon SQS) 队列。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

展示如何将 适用于 Python (Boto3) 的 AWS SDK 和 Amazon Textract 一起使用来检测文档图像中的文本、表单和表格元素。输入图像和 Amazon Textract 输出在 Tkinter 应用程序中显示，该应用程序可让您探索检测到的元素。

- 将文档图像提交到 Amazon Textract 并探索检测到的元素的输出。
- 将图像直接提交到 Amazon Textract，或通过 Amazon Simple Storage Service (Amazon S3) 桶提交图像。
- 使用异步 API 启动任务，在任务完成后将通知发布到 Amazon Simple Notification Service (Amazon SNS) 主题。
- 轮询 Amazon Simple Queue Service (Amazon SQS) 队列，以获取任务完成消息并显示结果。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 删除给定 Amazon S3 存储桶中的所有对象

以下代码示例显示如何删除 Amazon S3 存储桶中的所有对象。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除给定 Amazon S3 存储桶的所有对象。

```
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);
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的以下主题。
- [DeleteObjects](#)
- [ListObjectsV2](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 删除未完成的 Amazon S3 分段上传

以下代码示例显示如何删除或停止未完成的 Amazon S3 分段上传。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

要停止正在进行或由于任何原因而未完成的分段上传，您可以获取上传列表，然后删除这些上传，如以下示例所示。

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

```

要删除在某个日期之前或之后启动的未完成分段上传，您可以根据某个时间点有选择地删除分段上传，如以下示例所示。

```

/**
 * 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);
                }
            }
        }
    }
}

```

如果您在开始分段上传后可以访问上传 ID，则可以使用该 ID 删除正在进行的上传。

```

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

要一致地删除超过一定天数的未完成分段上传，请为存储桶设置存储桶生命周期配置。以下示例显示如何创建一条规则来删除超过 7 天的未完成上传。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

- [AbortMultipartUpload](#)
- [ListMultipartUploads](#)
- [PutBucketLifecycleConfiguration](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 通过 Amazon Rekognition 检测图像中的 PPE

以下代码示例展示如何构建采用 Amazon Rekognition 来检测图像中的个人防护设备（PPE）的应用程序。

Java

适用于 Java 的 SDK 2.x

展示如何创建 AWS Lambda 函数来检测包含个人防护设备的图像。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 检测从图像中提取的文本中的实体

以下代码示例显示了如何使用 Amazon Comprehend 检测 Amazon Textract 从存储在 Amazon S3 内的图像中提取的文本中的实体。

Python

SDK for Python (Boto3)

演示如何使用 适用于 Python (Boto3) 的 AWS SDK 在 Jupyter 笔记本中检测从图像中提取的文本中的实体。此示例使用 Amazon Textract 从存储在 Amazon Simple Storage Service (Amazon S3) 内的图像中提取文本，并使用 Amazon Comprehend 检测提取文本中的实体。

此示例是 Jupyter 笔记本，必须在可以托管笔记本电脑的环境中运行。有关如何使用 Amazon SageMaker AI 运行示例的说明，请参阅 [TextractAndComprehendNotebook.ipynb](#) 中的说明。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Comprehend
- Amazon S3
- Amazon Textract

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 检测图像中的人脸

以下代码示例展示了如何：

- 将图像保存到 Amazon S3 存储桶中。
- 使用 Amazon Rekognition 检测面部细节，例如年龄范围、性别和情绪（如微笑）。
- 显示这些细节。

Rust

适用于 Rust 的 SDK

将图像保存到具有 uploads 前缀的 Amazon S3 存储桶中，使用 Amazon Rekognition 检测面部细节，例如年龄范围、性别和情绪（微笑等），并显示这些细节。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition

- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS 软件开发工具包通过 Amazon Rekognition 检测图像中的对象

以下代码示例展示如何构建采用 Amazon Rekognition 来按类别检测图像中对象的应用程序。

.NET

适用于 .NET 的 SDK

展示如何使用 Amazon Rekognition .NET API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3

- Amazon SES

JavaScript

SDK for JavaScript (v3)

展示如何使用 Amazon Rekognition 和 适用于 JavaScript 的 AWS SDK 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

了解如何：

- 使用 Amazon Cognito 创建未经身份验证的用户。
- 使用 Amazon Rekognition 分析包含对象的图像。
- 为 Amazon SES 验证电子邮件地址。
- 使用 Amazon SES 发送电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK for Kotlin

展示如何使用 Amazon Rekognition Kotlin API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition

- Amazon S3
- Amazon SES

Python

SDK for Python (Boto3)

介绍如何使用 适用于 Python (Boto3) 的 AWS SDK 创建一个 Web 应用程序，让您可以执行以下操作：

- 将照片上载到 Amazon Simple Storage Service (Amazon S3) 存储桶。
- 使用 Amazon Rekognition 来分析和标注照片。
- 使用 Amazon Simple Email Service (Amazon SES) 发送图像分析的电子邮件报告。

此示例包含两个主要组件：用 JavaScript 编写的使用 React 构建的网页，以及用 Python 编写的使用 Flask-RESTful 构建的 REST 服务。

可以使用 React 网页执行以下操作：

- 显示存储在 S3 存储桶中的图像列表。
- 将计算机中的图像上载到 S3 存储桶。
- 显示图像和用于识别图像中检测到的物品的标注。
- 获取 S3 存储桶中所有图像的报告并发送报告电子邮件。

该网页调用 REST 服务。该服务将请求发送到 AWS 以执行以下操作：

- 获取并筛选 S3 存储桶中的图像列表。
- 将照片上载到 S3 存储桶。
- 使用 Amazon Rekognition 分析各张照片并获取标注列表，这些标注用于识别在照片中检测到的物品。
- 分析 S3 存储桶中的所有照片，然后使用 Amazon SES 通过电子邮件发送报告。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 通过 Amazon Rekognition 检测视频中的人物和对象

以下代码示例展示如何使用 Amazon Rekognition 检测视频中的人物和对象。

Java

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

适用于 Python 的 SDK (Boto3)

通过启动异步检测任务，使用 Amazon Rekognition 来检测视频中的人脸、对象和人物。此示例还将 Amazon Rekognition 配置为在任务完成时通知 Amazon Simple Notification Service (Amazon SNS) 主题，并订阅该主题的 Amazon Simple Queue Service (Amazon SQS) 队列。当队列收到有关任务的消息时，将检索该任务并输出结果。

此示例最好在 GitHub 上查看。有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3

- Amazon SES
- Amazon SNS
- Amazon SQS

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

从 Amazon Simple Storage Service (Amazon S3) 存储桶下载 S3“目录”

以下代码示例演示如何下载和筛选 Amazon S3 存储桶“目录”的内容。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例演示如何使用 AWS SDK for Java 2.x 中的 [S3TransferManager](#) 从 Amazon S3 存储桶下载“目录”。它还演示如何在请求中使用 [DownloadFilters](#)。

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

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 Amazon Simple Storage Service (Amazon S3) 桶中的所有对象下载到本地目录

以下代码示例演示了如何将 Amazon Simple Storage Service (Amazon S3) 桶中的所有对象下载到本地目录。

Java

适用于 Java 的 SDK 2.x

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 [S3TransferManager](#) 下载同一个 S3 存储桶中的所有 S3 对象。查看[完整文件](#)并[进行测试](#)。

```
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();
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 [DownloadDirectory](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 从 Amazon S3 对象下载未知大小的流

下面的代码示例演示如何从 Amazon S3 对象下载未知大小的流。

Swift

适用于 Swift 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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).")
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 从多区域接入点中获取 Amazon S3 对象

以下代码示例显示如何从多区域接入点中获取对象。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

将 S3 客户端配置为使用非对称 Sigv4 (Sigv4a) 签名算法。

```
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  
}
```

使用多区域接入点 ARN 而不是桶名称来检索对象。

```
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
}
```

- 有关更多信息，请参阅 [AWS SDK for Kotlin 开发人员指南](#)。
- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API Reference》中的 [GetObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 并指定 If-Modified-Since 标头以从 Amazon S3 存储桶获取对象

下面的代码示例显示如何从 S3 存储桶中的对象读取数据，但前提是该桶自上次检索以来未被修改。

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》中的 [GetObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 加密 Amazon S3 对象入门

下面的代码示例显示如何开始 Amazon S3 对象的加密。

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
    the
        S3
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon
        /// 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);
}
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考中的以下主题。
 - [CopyObject](#)
 - [GetObject](#)
 - [GetObjectMetadata](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 为 Amazon S3 存储桶和对象加标签入门

下面的代码示例显示如何开始为 AmazonS3 对象加标签。

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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}'");
    }
}
}

```

- 有关更多信息，请参阅《适用于 .NET 的 AWS SDK API 参考》中的 [GetObjectTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

大数据处理集群入门

以下代码示例展示了如何：

- 创建 EC2 密钥对
- 设置存储并准备应用程序
- 清理资源

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/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."
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
 - [AddSteps](#)
 - [Cp](#)
 - [CreateCluster](#) :
 - [CreateDefaultRoles](#)
 - [:CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [Ssh](#)
 - [TerminateClusters](#)
 - [Wait](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

配置管理入门

以下代码示例展示了如何：

- 创建 Amazon S3 存储桶
- 创建 Amazon SNS 主题
- 为 Config 创建 IAM 角色
- 设置 Config 配置记录器
- 设置 Config 传输通道
- 启动配置记录器
- 验证 Config 设置

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/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!"
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [AttachRolePolicy](#)
- [:CreateBucket](#)
- [CreateRole](#)
- [CreateTopic](#)
- [:DeleteBucket](#)
- [DeleteConfigurationRecorder](#)
- [DeleteDeliveryChannel](#)
- [DeleteRole](#)
- [DeleteRolePolicy](#)
- [DeleteTopic](#)
- [DescribeConfigurationRecorderStatus](#)
- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)
- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

文档文本提取入门

以下代码示例展示了如何：

- 创建 S3 存储桶
- 将文档上传到 S3
- 清理资源

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====="
echo "Amazon Textract Getting Started Tutorial"
echo "====="
echo "This script will guide you through using Amazon Textract to analyze document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
```

```

    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-Security-Token\).*\/1=***REDACTED***\/g'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ]; then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then

```

```

        echo "ERROR: AWS CLI is not installed."
        exit 1
    fi

    AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
    AWS_CONFIG_STATUS=$?
    if [ $AWS_CONFIG_STATUS -ne 0 ]; then
        echo "ERROR: AWS CLI is not properly configured."
        echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*\/1=***REDACTED***/'
        exit 1
    fi

    # Verify AWS region is configured and supports Textract
    AWS_REGION=$(aws configure get region)
    if [ -z "$AWS_REGION" ]; then
        echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
        default region."
        exit 1
    fi

    # Check if Textract is available in the configured region
    echo "Checking if Amazon Textract is available in region $AWS_REGION..."
    TEXTTRACT_CHECK=$(aws textract help 2>&1)
    TEXTTRACT_CHECK_STATUS=$?
    if [ $TEXTTRACT_CHECK_STATUS -ne 0 ]; then
        echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
        exit 1
    fi

    # Generate a random identifier for S3 bucket
    RANDOM_ID=$(openssl rand -hex 6)
    # Check for shared prereq bucket
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
    bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null || echo "")
    if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
        BUCKET_NAME="$PREREQ_BUCKET"
        BUCKET_IS_SHARED=true
        echo "Using shared bucket: $BUCKET_NAME"
    else
        BUCKET_IS_SHARED=false
        BUCKET_NAME="textract-${RANDOM_ID}"
    fi

```

```

fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAyAAAFfcSjAAAAADULEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"

```

```
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp "$DOCUMENT_NAME" s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
    "S3Object": {
        "Bucket": "BUCKET_PLACEHOLDER",
        "Name": "DOCUMENT_PLACEHOLDER"
    }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES", "FORMS", "SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?
```

```

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*\/\1=***REDACTED***\/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "=====
echo "Analysis Summary"
echo "=====
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
'{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
| wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"

```



```

echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "====="
echo "RESOURCES CREATED"
echo "====="
for resource in "${RESOURCES_CREATED[@]}; do
    echo "- $resource"
done
echo ""
echo "====="
echo "CLEANUP CONFIRMATION"
echo "====="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

```

```
echo ""
echo "=====
echo "Tutorial complete!"
echo "=====
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
 - [AnalyzeDocument](#)
 - [Cp](#)
 - [帮助](#) !
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

机器学习特征存放区入门

以下代码示例展示了如何：

- 设置 IAM 权限
- 创建 SageMaker 执行角色
- 创建特征组
- 清理资源

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
```

```

wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
"$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null

```

```

        ;;
        "IAMRole")
            echo "Detaching policies from role: $resource_name"
            aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
            aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
            echo "Deleting IAM role: $resource_name"
            aws iam delete-role --role-name "$resource_name" 2>/dev/null
            ;;
        *)
            echo "Unknown resource type: $resource_type"
            ;;
    esac
done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then

```

```
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
    Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}
```

```
# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
```

```
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1
```



```

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'$S3_BUCKET_NAME'/'$PREFIX'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \

```

```

    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'$S3_BUCKET_NAME'/'$PREFIX'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1

```

```
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "name", "ValueAsString": "John Doe"},
    {"FeatureName": "age", "ValueAsString": "35"},
    {"FeatureName": "address", "ValueAsString": "123 Main St"},
    {"FeatureName": "membership_type", "ValueAsString": "premium"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "109382"},
    {"FeatureName": "name", "ValueAsString": "Jane Smith"},
    {"FeatureName": "age", "ValueAsString": "28"},
    {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
    {"FeatureName": "membership_type", "ValueAsString": "standard"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)
```

```
if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
```

```
        cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
| grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi
```

```
echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "=====
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "=====
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:'
[^\[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "=====
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
```

```

echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Skipping cleanup. Resources will remain in your account."
    echo "To clean up later, delete the following resources:"
    echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
$ORDERS_FEATURE_GROUP_NAME"
    echo "- S3 Bucket: $S3_BUCKET_NAME"
    if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
        echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
'IAMRole:[^[:space:]]*' | cut -d: -f2)"
    fi
    echo ""
    echo "Estimated ongoing cost: ~$0.01 per month for online store"
fi
fi

echo "Script completed at $(date)"

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [AttachRolePolicy](#)
- [:CreateBucket](#)
- [CreateFeatureGroup](#)
- [CreateRole](#)
- [:DeleteBucket](#)
- [DeleteFeatureGroup](#)
- [DeleteRole](#)
- [DescribeFeatureGroup](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [GetRole](#)

- [ListFeatureGroups](#)
- [PutPublicAccessBlock](#)
- [Rm](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

对象存储入门

以下代码示例展示了如何：

- 创建您的第一个 S3 存储桶
- 上传对象
- 启用版本控制
- 配置默认加密
- 向存储桶添加标签
- 列出对象和版本
- 清理资源

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail
```



```

# =====
# Prerequisites check
# =====

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
"$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
fi

# =====
# Setup: logging, temp directory, resource tracking
# =====

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"
fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="$TEMP_DIR/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

```

```

echo "=====
echo "S3 Getting Started"
echo "=====
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# =====
# Helper functions
# =====

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
    then
        return 0
    fi

    echo "$versions" | jq -r '.[ ] | "\(.Key)\t\(.VersionId)'" 2>/dev/null | while
IFS=$'\t' read -r key version_id; do
        if [ -n "$key" ] && [ "$key" != "null" ]; then
            aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
"$version_id" >/dev/null 2>&1 || true
        fi
    done

    return 0
}

# =====
# Error handling and cleanup functions

```

```
# =====

cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP"
    echo "=====

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[.
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[.
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "=====
    echo "ERROR on line ${line_number}"
}
```

```

    echo "======"
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# =====
# Step 1: Create a bucket
# =====

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
    then
        echo "ERROR: Failed to create bucket $BUCKET_NAME"
        exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
            fi
        fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

```

```

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag bucket"
    fi
fi
echo ""

# =====
# Step 2: Upload a sample text file
# =====

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# =====
# Step 3: Download the object

```

```
# =====

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""

# =====
# Step 4: Copy the object to a folder prefix
# =====

echo "Step 4: Copying object to a folder prefix..."

if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# =====
# Step 5: Enable versioning and upload a second version
# =====

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
```

```

    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# =====
# Step 6: Configure SSE-S3 encryption
# =====

echo "Step 6: Configuring SSE-S3 default encryption..."

if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
echo "SSE-S3 encryption configured."

```

```

echo ""

# =====
# Step 7: Block all public access
# =====

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# =====
# Step 8: Configure bucket logging
# =====

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \
        --bucket "$LOG_TARGET_BUCKET" \

```



```

        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag log bucket"
    fi

    aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

    if ! aws s3api put-bucket-logging \
        --bucket "$BUCKET_NAME" \
        --bucket-logging-status '{
            "LoggingEnabled": {
                "TargetBucket": "'$LOG_TARGET_BUCKET'",
                "TargetPrefix": "logs/"
            }
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to configure bucket logging"
    else
        echo "Bucket logging configured."
    fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# =====
# Step 9: Tag the bucket
# =====

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{

```

```

        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
            {
                "Key": "Environment",
                "Value": "Tutorial"
            },
            {
                "Key": "Project",
                "Value": "S3-GettingStarted"
            },
            {
                "Key": "ManagedBy",
                "Value": "Bash-Tutorial"
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to tag bucket"
    exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# =====
# Step 10: List objects and versions
# =====

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

```

```

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# =====
# Step 11: Cleanup
# =====

echo ""
echo "=====
echo "TUTORIAL COMPLETE"
echo "=====
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

- [HeadObject](#)
- [ListObjectVersions](#)
- [ListObjectsV2](#)
- [PutBucketEncryption](#)
- [PutBucketTagging](#)
- [PutBucketVersioning](#)
- [PutObject](#)
- [PutPublicAccessBlock](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

查询分析入门

以下代码示例展示了如何：

- 为查询结果创建 S3 存储桶
- 创建一个数据库
- 创建表
- 运行查询
- 创建和使用命名查询
- 清理资源

Bash

AWS CLI 及 Bash 脚本

Note

查看 GitHub，了解更多信息。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/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."
            fi
        done
    fi
fi

```

```

        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
        echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
        break
    fi
    echo "Database deletion in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done
fi
fi

# Empty and delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Emptying S3 bucket: $S3_BUCKET"
    EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
    if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
    else
        echo "S3 bucket emptied successfully."
    fi

    echo "Deleting S3 bucket: $S3_BUCKET"
    DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    if echo "$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!"
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
 - [Cp](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [StartQueryExecution](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 AWS SDK 使用 Amazon S3 对象锁定功能

以下代码示例显示了如何使用 S3 对象锁定特征。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行演示 Amazon S3 对象锁定功能的交互式场景。

```
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;
    }
}

```

S3 函数的包装器类。

```

using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper

```

```
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
    bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
    enableObjectLock)
    {
        Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
    {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

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



```
/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}

/// <summary>
```

```

    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>

```

```
public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
{
    var enabledString = enableObjectLock ? "Enabled" : "Disabled";
    var timeDifference = retainUntilDate.Subtract(DateTime.Now);
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                Rule = new ObjectLockRule()
                {
                    DefaultRetention = new DefaultRetention()
                    {
                        Mode = retention,
                        Days = timeDifference.Days // Can be specified in
days or years but not both.
                    }
                }
            }
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{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;
        }
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的以下主题。
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)

- [PutObjectRetention](#)

Go

适用于 Go 的 SDK V2

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行演示 Amazon S3 对象锁定功能的交互式场景。

```
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)
                }
            }
        }
    }
}
```

```

    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))
}

```

定义一个结构来包装此示例中使用的 S3 操作。

```

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
}
```

清理资源。

```
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)
    }
}
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 AWS SDK API Reference》中的以下主题。
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)

- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行演示 Amazon S3 对象锁定功能的交互式场景。

```
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;
    }
}

```

S3 函数的包装器类。

```

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 retention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(retention)
    .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());
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

JavaScript

适用于 JavaScript 的 SDK (v3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

场景的入口点 (index.js)。这用于编排所有步骤。访问 GitHub，来查看 Scenario、ScenarioInput、ScenarioOutput 和 ScenarioAction 的实现详细信息。

```
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]",
  });
}

```

向控制台输出欢迎消息 (`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 };

```

部署存储桶、对象和文件设置 (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,
};

```

查看和删除存储桶中的文件 (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 };

```

销毁所有创建的资源 (`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 };
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的以下主题。
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 发起 Amazon S3 条件请求

以下代码示例演示如何向 Amazon S3 请求添加前提条件。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行一个交互式场景，演示 Amazon S3 条件请求的功能。

```
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;
}
}
```

S3 函数的包装器类。

```
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;
    }
}
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的以下主题。
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

工作流程的入口点 (`index.js`)。这用于编排所有步骤。访问 [GitHub](#)，来查看 `Scenario`、`ScenarioInput`、`ScenarioOutput` 和 `ScenarioAction` 的实现详细信息。

```
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]",
    });
  }
}

```

向控制台输出欢迎消息 (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 };

```

部署存储桶和对象 (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,
};
```

使用 S3 条件请求获取、复制和放置对象 (repl.steps.js)。

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,
  PutObjectCommand,
```

```
} from "@aws-sdk/client-s3";
import data from "./object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
      ],
    }
  )
```

```

        name: "Perform a conditional copy. These examples use the key name
prefix defined in ./object_name.json.",
        value: choices.CONDITIONAL_COPY,
    },
    {
        name: "Perform a conditional write. This example use the sample file ./
text02.txt.",
        value: choices.CONDITIONAL_WRITE,
    },
    { name: "Finish the workflow.", value: choices.EXIT },
],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket })),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key } = version;
            files.push({ bucket, key: Key });
        }
    }
    return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
    const objectsResponse = await client.send(
        new GetObjectCommand({
            Bucket: bucket,
            Key: key,
        })),
    );

```



```

    );
    return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",

```

```

        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
    ],
},
);
const condCopyOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional copy action would you like to take?",
    {
        type: "select",
        choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ],
    },
);
const condWriteOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional write action would you like to take?",
    {
        type: "select",
        choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail.",
        ],
    },
);

const { replChoice } = state;

switch (replChoice) {
    case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,
        ]);
        state.replOutput = files

```

```

        .map(
            (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
        )
        .join("\n");
    break;
}
case choices.CONDITIONAL_READ:
{
    const selectedCondRead = await condReadOptions.handle(state);
    if (
        selectedCondRead ===
        "If-Match: using the object's ETag. This condition should succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        try {
            await client.send(
                new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                }),
            );
            state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
        } catch (err) {
            state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
        }
        break;
    }
    if (
        selectedCondRead ===
        "If-None-Match: using the object's ETag. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        try {
            await client.send(

```

```

        new GetObjectCommand({
            Bucket: bucket,
            Key: key,
            IfNoneMatch: ETag,
        }),
    );
    state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
    } catch (err) {
        state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
    }
    break;
}
if (
    selectedCondRead ===
    "If-Modified-Since: using yesterday's date. This condition should
succeed."
) {
    const date = new Date();
    date.setDate(date.getDate() - 1);

    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    try {
        await client.send(
            new GetObjectCommand({
                Bucket: bucket,
                Key: key,
                IfModifiedSince: date,
            }),
        );
        state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
    } catch (err) {
        state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondRead ===
    "If-Unmodified-Since: using yesterday's date. This condition should
fail."

```

```

    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";

      const date = new Date();
      date.setDate(date.getDate() - 1);
      try {
        await client.send(
          new GetObjectCommand({
            Bucket: bucket,
            Key: key,
            IfUnmodifiedSince: date,
          }),
        );
        state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
      } catch (err) {
        state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
      }
      break;
    }
  }
  break;
case choices.CONDITIONAL_COPY: {
  const selectedCondCopy = await condCopyOptions.handle(state);
  if (
    selectedCondCopy ===
    "If-Match: using the object's ETag. This condition should succeed."
  ) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);

    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;
    try {
      await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: state.destinationBucketName,

```

```

        Key: copiedKey,
        CopySourceIfMatch: ETag,
    )),
    );
    state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
    } catch (err) {
        state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondCopy ===
    "If-None-Match: using the object's ETag. This condition should fail."
) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    try {
        await client.send(
            new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfNoneMatch: ETag,
            )),
        );
        state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
    } catch (err) {
        state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (

```

```

        selectedCondCopy ===
        "If-Modified-Since: using yesterday's date. This condition should
succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const copySource = `${bucket}/${key}`;
        // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
        const name = data.name;
        const copiedKey = `${name}${key}`;

        const date = new Date();
        date.setDate(date.getDate() - 1);

        try {
            await client.send(
                new CopyObjectCommand({
                    CopySource: copySource,
                    Bucket: state.destinationBucketName,
                    Key: copiedKey,
                    CopySourceIfModifiedSince: date,
                }),
            );
            state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
        } catch (err) {
            state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
        }
        break;
    }
    if (
        selectedCondCopy ===
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const copySource = `${bucket}/${key}`;
        // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
        const name = data.name;

```

```

const copiedKey = `${name}${key}`;

const date = new Date();
date.setDate(date.getDate() - 1);

try {
  await client.send(
    new CopyObjectCommand({
      CopySource: copySource,
      Bucket: state.destinationBucketName,
      Key: copiedKey,
      CopySourceIfUnmodifiedSince: date,
    }),
  );
  state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
} catch (err) {
  state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
}
break;
}
case choices.CONDITIONAL_WRITE:
{
  const selectedCondWrite = await condWriteOptions.handle(state);
  if (
    selectedCondWrite ===
    "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
  ) {
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const key = "text02.txt";
    const __filename = fileURLToPath(import.meta.url);
    const __dirname = dirname(__filename);
    const filePath = path.join(__dirname, "text02.txt");
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: `${state.destinationBucketName}`,
          Key: `${key}`,
          Body: await readFile(filePath),

```



```

                IfNoneMatch: "*",
            )),
        );
        state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
        } catch (err) {
            state.replOutput = `Unable to upload object to bucket
${state.destinationBucketName}:${err.message}`;
        }
        break;
    }
}
break;

default:
    throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
            preformatted: true,
        }),
    },
},
);

export { replInput, choices };

```

销毁所有创建的资源 (`clean.steps.js`) 。

```

import {
    DeleteObjectCommand,
    DeleteBucketCommand,
    ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

```

```
/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
            console.log(`An error occurred: ${err.message} `);
          }
        }
      }
    }
  })
```

```
    } catch (e) {
      if (e instanceof Error && e.name === "NoSuchBucket") {
        console.log("Objects and buckets have already been deleted.");
        continue;
      }
      throw e;
    }

    await client.send(new DeleteBucketCommand({ Bucket: bucket }));
    console.log(`Delete for ${bucket} complete.`);
  }
});

export { confirmCleanup, cleanupAction };
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的以下主题。
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行一个交互式场景，演示 Amazon S3 条件请求。

```
"""
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()

```

定义条件请求操作的包装器类。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 管理 Amazon S3 存储桶的访问控制列表 (ACL)

以下代码示例显示如何管理 Amazon S3 存储桶的访问控制列表 (ACL)。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
```

```

    /// </summary>
    public class ManageACLs
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket1";
            string newBucketName = "amzn-s3-demo-bucket2";
            string keyName = "sample-object.txt";
            string emailAddress = "someone@example.com";

            // If the AWS Region where your bucket is located is different from
            // the Region defined for the default user, pass the Amazon S3
bucket's
            // name to the client constructor. It should look like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,

```

```

        string bucketName,
        string newBucketName,
        string keyName,
        string emailAddress)
    {
        try
        {
            // Create a new Amazon S3 bucket and specify canned ACL.
            var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

            // Get the ACL on a bucket.
            await GetBucketACLAsync(client, bucketName);

            // Add (replace) the ACL on an object in a bucket.
            await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Exception: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL attached.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="newBucketName">A string representing the name of the
    /// new Amazon S3 bucket.</param>
    /// <returns>Returns a boolean value indicating success or failure.</
returns>
    public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
    {
        var request = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = S3Region.EUWest1,

            // Add a canned ACL.
            CannedACL = S3CannedACL.LogDeliveryWrite,
        };
    }

```



```

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

    /// <summary>
    /// Retrieves the ACL associated with the Amazon S3 bucket name in the
    /// bucketName parameter.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
    /// ACL list.</param>
    /// <returns>Returns an S3AccessControlList returned from the call to
    /// GetACLAsync.</returns>
    public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
    {
        GetACLResponse response = await client.GetACLAsync(new GetACLRequest
        {
            BucketName = bucketName,
        });

        return response.AccessControlList;
    }

    /// <summary>
    /// Adds a new ACL to an existing object in the Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
S3
    /// bucket containing the object to which we want to apply a new ACL.</
param>
    /// <param name="keyName">A string representing the name of the object
    /// to which we want to apply the new ACL.</param>
    /// <param name="emailAddress">The email address of the person to whom
    /// we will be applying to whom access will be granted.</param>

```

```
public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
string bucketName, string keyName, string emailAddress)
{
    // Retrieve the ACL for an object.
    GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest
    {
        BucketName = bucketName,
        Key = keyName,
    });

    S3AccessControlList acl = aclResponse.AccessControlList;

    // Retrieve the owner.
    Owner owner = acl.Owner;

    // Clear existing grants.
    acl.Grants.Clear();

    // Add a grant to reset the owner's full permission
    // (the previous clear statement removed all permissions).
    var fullControlGrant = new S3Grant
    {
        Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
    };
    acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

    // Specify email to identify grantee for granting permissions.
    var grantUsingEmail = new S3Grant
    {
        Grantee = new S3Grantee { EmailAddress = emailAddress },
        Permission = S3Permission.WRITE_ACP,
    };

    // Specify log delivery group as grantee.
    var grantLogDeliveryGroup = new S3Grant
    {
        Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
        Permission = S3Permission.WRITE,
    };

    // Create a new ACL.
    var newAcl = new S3AccessControlList
```

```
        {  
            Grants = new List<S3Grant> { grantUsingEmail,  
grantLogDeliveryGroup },  
            Owner = owner,  
        };  
  
        // Set the new ACL. We're throwing away the response here.  
        _ = await client.PutACLAsync(new PutACLRequest  
        {  
            BucketName = bucketName,  
            Key = keyName,  
            AccessControlList = newAcl,  
        });  
    }  
  
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的以下主题。
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 的 Amazon S3 管理大型 Amazon SQS 消息

以下代码示例演示如何使用 Amazon SQS 扩展型客户端库来处理大型 Amazon SQS 消息。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
    }
}
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x 开发人员指南》<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-s3-messages.html>。
- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
 - [:CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 AWS SDK 使用 Lambda 函数批量管理版本控制的 Amazon S3 对象

以下代码示例显示了如何使用 Lambda 函数批量管理版本控制的 S3 对象。

Python

SDK for Python (Boto3)

显示如何通过创建调用 AWS Lambda 函数来执行处理的任务来批量操作 Amazon Simple Storage Service (Amazon S3) 版本控制对象。此示例将创建了一个启用版本控制的桶，上传 Lewis Carroll 所写的诗歌《You Are Old, Father William》中的诗节，并使用 Amazon S3 批处理任务以各种方式调整这首诗。

了解如何：

- 创建对版本控制对象运行的 Lambda 函数。
- 创建要更新的对象清单。

- 创建调用 Lambda 函数来更新对象的批处理任务。
- 删除 Lambda 函数。
- 清空并删除版本控制的桶。

此示例最好在 GitHub 上查看。有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 解析 Amazon S3 URI

以下代码示例显示如何解析 Amazon S3 URI 以提取诸如桶名称和对象密钥等重要组件。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 [S3Uri](#) 类解析 Amazon S3 URI。

```
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);
    }
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 执行 Amazon S3 对象的分段复制

以下代码示例显示如何执行 Amazon S3 对象的分段复制。

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的以下主题。
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [GetObjectMetadata](#)
 - [UploadPartCopy](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 接收和处理 Amazon S3 事件通知

以下代码示例显示了如何以面向对象的方式处理 S3 事件通知。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例显示了如何使用 Amazon SQS 处理 S3 通知事件。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

- [DeleteMessageBatch](#)
- [GetQueueAttributes](#)
- [PutBucketNotificationConfiguration](#)
- [ReceiveMessage](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 保存 EXIF 和其他图像信息

以下代码示例展示了如何：

- 从 JPG、JPEG 或 PNG 文件中获取 EXIF 信息。
- 将图像文件上传到 Amazon S3 存储桶。
- 使用 Amazon Rekognition 识别文件中的三个主要属性（标签）。
- 将 EXIF 和标签信息添加到该区域的 Amazon DynamoDB 表中。

Rust

适用于 Rust 的 SDK

从 JPG、JPEG 或 PNG 文件中获取 EXIF 信息，将图像文件上传到 Amazon S3 存储桶，使用 Amazon Rekognition 识别文件中的三个主要属性（Amazon Rekognition 中的标签），然后将 EXIF 和标签信息添加到该区域的 Amazon DynamoDB 表中。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- DynamoDB
- Amazon Rekognition
- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 向 Amazon EventBridge 发送 S3 事件通知

以下代码示例显示了如何启用存储桶来将 S3 事件通知发送到 EventBridge，并将通知路由到 Amazon SNS 主题和 Amazon SQS 队列。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/** 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;
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

跟踪使用 AWS SDK 执行的 Amazon S3 对象上传或下载操作

下面的代码示例展示了如何跟踪 Amazon S3 对象的上传或下载。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

跟踪文件上传进度。

```
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!

  */
}
```

跟踪文件下载进度。

```
public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFileWithPath) {
  DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
    .getObjectRequest(b -> b.bucket(bucketName).key(key))
    .addTransferListener(LoggingTransferListener.create()) // Add
listener.

    .destination(Paths.get(downloadedFileWithPath))
    .build();
```

```

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
            <Configuration status="WARN">
                <Appenders>
                    <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                        <PatternLayout pattern="%m%n"/>
                    </Console>
                </Appenders>

                <Loggers>
                    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                        <AppenderRef ref="AlignedConsoleAppender"/>
                    </logger>
                </Loggers>
            </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file download.
            Transfer initiated...
            |=====| 39.4%
            |=====| 78.8%
            |=====| 100.0%
            Transfer complete!

        */
    }

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
 - [GetObject](#)
 - [PutObject](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 S3 对象 Lambda 转换应用程序的数据

下面的代码示例显示如何使用 S3 对象 Lambda 转换应用程序的数据。

.NET

适用于 .NET 的 SDK

显示如何将自定义代码添加到标准 S3 GET 请求中，来修改从 S3 检索的请求对象，从而使该对象适合发出请求的客户端或应用程序的需要。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Lambda
- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 进行单元测试和集成测试的示例方法

以下代码示例显示了使用 AWS SDK 编写单元测试和集成测试时的最佳实践方法示例。

Rust

适用于 Rust 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

Cargo.toml 用于测试示例。

```
[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"

```

使用 automock 和服务包装器的单元测试示例。

```

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);
}
}

```

使用 StaticReplayClient 的集成测试示例。

```

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(&[]);
    }
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

以递归方式将本地目录上传到 Amazon Simple Storage Service (Amazon S3) 桶

以下代码示例显示如何以递归方式将本地目录上传到 Amazon Simple Storage Service (Amazon S3) 桶。

Java

适用于 Java 的 SDK 2.x

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 [S3TransferManager](#) 以[上传本地目录](#)。查看[完整文件](#)并[进行测试](#)。

```
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();
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 [UploadDirectory](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

借助 AWS SDK 向 Amazon S3 上传大文件或从 Amazon S3 下载大文件

下面的代码示例说明如何向 Amazon S3 上传大文件或从 Amazon S3 下载大文件。

有关更多信息，请参阅[使用分段上传操作上传对象](#)。

.NET

适用于 .NET 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

调用使用 Amazon S3 TransferUtility 将文件传入和传出 S3 存储桶的函数。

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");
```

```
var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
    bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
    {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
    bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
    {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");
```

```
success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
    keyName, localPath);
if (success)
{
    Console.WriteLine($"Successfully downloaded the file, {keyName} from
    {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
    bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
    {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}
```

```
// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
an\n\t S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
directory.");
    Console.WriteLine($"{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
```

```
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
    s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

上传单个文件。

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

上传整个本地目录。

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

下载单个文件。

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

下载 S3 存储桶的内容。

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

使用 TransferUtility 跟踪上传的进度。

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

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();
```

```
        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }

    /// <summary>
    /// Starts an Amazon S3 multipart upload and assigns an event handler to
    /// track the progress of the upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// perform the multipart upload.</param>
    /// <param name="bucketName">The name of the bucket to which to upload
    /// the file.</param>
    /// <param name="filePath">The path, including the file name of the
    /// file to be uploaded to the Amazon S3 bucket.</param>
    /// <param name="keyName">The file name to be used in the
    /// destination Amazon S3 bucket.</param>
    public static async Task TrackMPUAsync(
        IAmazonS3 client,
        string bucketName,
        string filePath,
        string keyName)
    {
        try
        {
            var fileTransferUtility = new TransferUtility(client);

            // Use TransferUtilityUploadRequest to configure options.
            // In this example we subscribe to an event.
            var uploadRequest =
                new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    Key = keyName,
                };

            uploadRequest.UploadProgressEvent +=
                new EventHandler<UploadProgressArgs>(
                    UploadRequest_UploadPartProgressEvent);

            await fileTransferUtility.UploadAsync(uploadRequest);
            Console.WriteLine("Upload completed");
        }
        catch (AmazonS3Exception ex)
        {

```

```

        Console.WriteLine($"Error:: {ex.Message}");
    }
}

/// <summary>
/// Event handler to check the progress of the multipart upload.
/// </summary>
/// <param name="sender">The object that raised the event.</param>
/// <param name="e">The object that contains multipart upload
/// information.</param>
public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
{
    // Process event.
    Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
}
}

```

上传经过加密的对象。

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
/// perform a multipart upload to an Amazon S3 bucket.
/// </summary>
public class SSECLowLevelMPUCopyObject
{
    public static async Task Main()
    {
        string existingBucketName = "amzn-s3-demo-bucket";
        string sourceKeyName = "sample_file.txt";
        string targetKeyName = "sample_file_copy.txt";
        string filePath = $"sample\\{targetKeyName}";
    }
}

```

```

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USEast1.
        IAmazonS3 client = new AmazonS3Client();

        // Create the encryption key.
        var base64Key = CreateEncryptionKey();

        await CreateSampleObjUsingClientEncryptionKeyAsync(
            client,
            existingBucketName,
            sourceKeyName,
            filePath,
            base64Key);
    }

    /// <summary>
    /// Creates the encryption key to use with the multipart upload.
    /// </summary>
    /// <returns>A string containing the base64-encoded key for encrypting
    /// the multipart upload.</returns>
    public static string CreateEncryptionKey()
    {
        Aes aesEncryption = Aes.Create();
        aesEncryption.KeySize = 256;
        aesEncryption.GenerateKey();
        string base64Key = Convert.ToBase64String(aesEncryption.Key);
        return base64Key;
    }

    /// <summary>
    /// Creates and uploads an object using a multipart upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object used to
    /// initialize and perform the multipart upload.</param>
    /// <param name="existingBucketName">The name of the bucket to which
    /// the object will be uploaded.</param>
    /// <param name="sourceKeyName">The source object name.</param>
    /// <param name="filePath">The location of the source object.</param>
    /// <param name="base64Key">The encryption key to use with the upload.</
param>
    public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
        IAmazonS3 client,

```

```
        string existingBucketName,
        string sourceKeyName,
        string filePath,
        string base64Key)
    {
        List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

        InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        InitiateMultipartUploadResponse initResponse =
            await client.InitiateMultipartUploadAsync(initiateRequest);

        long contentLength = new FileInfo(filePath).Length;
        long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

        try
        {
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++)
            {
                UploadPartRequest uploadRequest = new UploadPartRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    PartNumber = i,
                    PartSize = partSize,
                    FilePosition = filePosition,
                    FilePath = filePath,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key,
                };

                // Upload part and add response to our list.
```

```
        uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

        filePosition += partSize;
    }

    CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
    {
        BucketName = existingBucketName,
        Key = sourceKeyName,
        UploadId = initResponse.UploadId,
    };
    completeRequest.AddPartETags(uploadResponses);

    CompleteMultipartUploadResponse completeUploadResponse =
        await client.CompleteMultipartUploadAsync(completeRequest);
    }
catch (Exception exception)
{
    Console.WriteLine($"Exception occurred: {exception.Message}");

    // If there was an error, abort the multipart upload.
    AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
    {
        BucketName = existingBucketName,
        Key = sourceKeyName,
        UploadId = initResponse.UploadId,
    };

    await client.AbortMultipartUploadAsync(abortMPURequest);
}
}
```


Go

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建使用上传和下载管理器来将数据分解为多个分段并同时传输的函数。

```
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
}

```

运行一个交互式场景，演示如何在上下文中使用上传和下载管理器。

```

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

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

调用使用 `S3TransferManager` 将文件传入和传出 S3 存储桶的函数。

```
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();  
}
```

上传整个本地目录。

```
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();
}
```

上传单个文件。

```
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();
}
```

代码示例使用以下导入。

```
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;
```

在[基于 AWS CRT 的 S3 客户端](#)上使用 [S3 Transfer Manager](#)，以在内容大小超过阈值时透明地执行分段上传。默认阈值大小为 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();
    }
```

使用 [S3Client API](#) 执行分段上传。

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

```

使用启用分段支持的 [S3AsyncClient API](#) 执行分段上传。

```

/**
 * 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

适用于 JavaScript 的 SDK (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

上传大文件。

```
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;
    }
  }
};
```

下载大文件。

```
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

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用多种可用的传输管理器设置创建传输文件的功能。在文件传输过程中，使用回调类写入回调进度。

```
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
```

演示传输管理器的功能并报告结果。

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
```

```

        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
"demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

```

```

Ask the user if they want to run this specific demonstration.
If they say yes, create a file on the local path, upload it
using the specified upload function, then download it using the
specified download function.
"""
if download_args is None:
    download_args = {}
if upload_args is None:
    upload_args = {}
question = question.format(self.file_size_mb)
answer = input(f"{question} (y/n)")
if answer.lower() == "y":
    local_file_path, object_key, download_file_path =
self._create_demo_file()

    file_transfer.TransferConfig = self._config_wrapper(
        TransferConfig, CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Uploading", local_file_path, object_key, **upload_args
    )
    start_time = time.perf_counter()
    thread_info = upload_func(
        local_file_path,
        self.demo_bucket,
        object_key,
        self.file_size_mb,
        **upload_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

    file_transfer.TransferConfig = self._config_wrapper(
        TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Downloading", object_key, download_file_path, **download_args
    )
    start_time = time.perf_counter()
    thread_info = download_func(
        self.demo_bucket,
        object_key,
        download_file_path,
        self.file_size_mb,

```

```

        **download_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

def last_name_set(self):
    """Get the name set used for the last demo."""
    return self._chore_list[-1]

def cleanup(self):
    """
    Remove files from the demo folder, and uploaded objects from the
    Amazon S3 bucket.
    """
    print("-" * self._terminal_width)
    for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
        print(f"Removing {local_file_path}")
        try:
            os.remove(local_file_path)
        except FileNotFoundError as err:
            print(err)

        print(f"Removing {downloaded_file_path}")
        try:
            os.remove(downloaded_file_path)
        except FileNotFoundError as err:
            print(err)

        if self.demo_bucket:
            print(f"Removing {self.demo_bucket}:{s3_object_key}")
            try:
self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}"
        self._size_multiplier = MB
    elif platform.system() == "Linux" or platform.system() == "Darwin":

```

```

        self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported."
        )

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

    Returns:
    A tuple that contains the local file path, object name, and download
    file path.
    """
    file_name_template = "TestFile{}-{}.demo"
    local_suffix = "local"
    object_suffix = "s3object"
    download_suffix = "downloaded"
    file_tag = len(self._chore_list) + 1

    local_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag, local_suffix)
    )

    s3_object_key = file_name_template.format(file_tag, object_suffix)

    downloaded_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag,
download_suffix)
    )

    filled_cmd = self._create_file_cmd.format(
        local_file_path, self.file_size_mb * self._size_multiplier
    )

    print(
        f"Creating file of size {self.file_size_mb} MB "
        f"in {self.demo_folder} by running:"
    )

```



```

    )
    print(f"{'':4}{filled_cmd}")
    os.system(filled_cmd)

    chore = (local_file_path, s3_object_key, downloaded_file_path)
    self._chore_list.append(chore)
    return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print("-" * self._terminal_width)
    print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
    if kwargs:
        print("With extra args:")
        for arg, value in kwargs.items():
            print(f"{'':4}{arg:<20}: {value}")

@staticmethod
def ask_user(question):
    """
    Ask the user a yes or no question.

    Returns:
    True when the user answers 'y' or 'Y'; otherwise, False.
    """
    answer = input(f"{question} (y/n) ")
    return answer.lower() == "y"

@staticmethod
def _config_wrapper(func, config_attrs):
    def wrapper(*args, **kwargs):
        config = func(*args, **kwargs)
        print("With configuration:")
        for attr in config_attrs:
            print(f"{'':4}{attr:<20}: {getattr(config, attr)}")
        return config

    return wrapper

@staticmethod
def _report_transfer_result(thread_info, elapsed):
    """Report the result of a transfer, including per-thread data."""
    print(f"\nUsed {len(thread_info)} threads.")
    for ident, byte_count in thread_info.items():

```

```
        print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
    print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        }
    )
```

```

    },
)

# Upload using server-side encryption with customer-provided
# encryption keys.
# Generate a 256-bit key from a passphrase.
sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
demo_manager.demo(
    "Do you want to upload and download a {} MB file using "
    "server-side encryption?",
    file_transfer.upload_with_sse,
    file_transfer.download_with_sse,
    upload_args={"sse_key": sse_key},
    download_args={"sse_key": sse_key},
)

# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user(
    "Do you want to try to download the encrypted "
    "object without sending the required key?"
):
    try:
        _, object_key, download_file_path = demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket,
            object_key,
            download_file_path,
            demo_manager.file_size_mb,
        )
    except ClientError as err:
        print(
            "Got expected error when trying to download an encrypted "
            "object without specifying encryption info:"
        )
        print(f"{'':4}{err}")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
    "Demonstration complete. Do you want to remove local files " "and S3
objects?"
):
    demo_manager.cleanup()

```

```
if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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(())
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 将未知大小的流上传到 Amazon S3 对象

下面的代码示例演示了如何将未知大小的流上传到 Amazon S3 对象。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 [AWS 基于 CRT 的 S3 客户端](#)。

```
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;
    }
}

```

使用标准的[启用了分段上传的异步 S3 客户端](#)。

```

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;
    }
}

```

使用 [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

适用于 Swift 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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).")  
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS 开发工具包，对 Amazon S3 对象使用校验和

以下代码示例显示如何对 Amazon S3 对象使用校验和。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

代码示例使用以下导入的子集。

```
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;
```

在[构建 PutObjectRequest](#) 时为 putObject 方法指定校验和算法。

```
public void putObjectWithChecksum(String bucketName, String key) {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

在[构建 GetObjectRequest](#) 时为 getObject 方法验证校验和。

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

在[构建 PutObjectRequest](#) 时为 putObject 方法预先计算校验和。

```
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)));
}
```

在[基于 AWS CRT 的 S3 客户端](#)上使用 [S3 Transfer Manager](#)，以在内容大小超过阈值时透明地执行分段上传。默认阈值大小为 8 MB。

您可以为要使用的开发工具包指定校验和算法。默认情况下，开发工具包使用 CRC32 算法。

```
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();
}
```

```
}
```

使用 [S3Client API](#) 或 S3AsyncClient API 执行分段上传。如果指定其他校验和，则您必须指定在启动上传时使用的算法。您还必须为每个分段请求指定算法，并提供每个分段在上传后计算得出的校验和。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。
- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [UploadPart](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 AWS SDK 使用 Amazon S3 对象完整性功能

以下代码示例显示了如何使用 S3 对象完整性功能。

C++

SDK for C++

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行演示 Amazon S3 对象完整性功能的交互式场景。

```
#!/ 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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 AWS SDK API Reference》中的以下主题。
 - [AbortMultipartUpload](#)
 - [CompleteMultipartUpload](#)

- [CreateMultipartUpload](#)
- [DeleteObject](#)
- [GetObjectAttributes](#)
- [PutObject](#)
- [UploadPart](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 使用 Amazon S3 版本控制对象

以下代码示例展示了如何：

- 创建版本控制的 S3 存储桶。
- 获取对象的所有版本。
- 将对象回滚到以前的版本。
- 删除并还原版本控制的对象。
- 永久删除对象的所有版本。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建包装 S3 操作的函数。

```
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

```

将诗节上传到版本控制的对象并对其执行一系列操作。

```

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!")
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。
 - [:CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon S3 无服务器示例

以下代码示例演示了如何将 Amazon S3 与 AWS SDK 一起使用。

示例

- [通过 Amazon S3 触发器调用 Lambda 函数](#)

通过 Amazon S3 触发器调用 Lambda 函数

以下代码示例演示了如何实现一个 Lambda 函数，该函数接收通过将对象上传到 S3 存储桶而触发的事件。该函数从事件参数中检索 S3 存储桶名称和对象密钥，并调用 Amazon S3 API 来检索和记录对象的内容类型。

.NET

适用于 .NET 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 .NET 将 S3 事件与 Lambda 结合使用。

```
// 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

适用于 Go 的 SDK V2

Note

查看 GitHub，了解更多信息。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Go 将 S3 事件与 Lambda 结合使用。

```
// 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

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 S3 事件与 Lambda 结合使用。

```
// 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

适用于 JavaScript 的 SDK (v3)

Note

查看 [GitHub](#)，了解更多信息。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 JavaScript 将 S3 事件与 Lambda 结合使用。

```
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);
    }
};
```

使用 TypeScript 将 S3 事件与 Lambda 结合使用。

```
// 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

适用于 PHP 的 SDK

Note

查看 [GitHub](#)，了解更多信息。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 PHP 将 S3 事件与 Lambda 结合使用。

```
<?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

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub , 了解更多信息。在[无服务器示例](#)存储库中查找完整示例 , 并了解如何进行设置和运行。

使用 Python 将 S3 事件与 Lambda 结合使用。

```
# 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

适用于 Ruby 的 SDK

Note

查看 GitHub，了解更多信息。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 S3 事件与 Lambda 结合使用。

```
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
```

Rust

适用于 Rust 的 SDK

Note

查看 GitHub，了解更多信息。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Rust 将 S3 事件与 Lambda 结合使用。

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

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 的 Amazon S3 Control 代码示例

以下代码示例演示如何将 Amazon S3 Control 与 AWS 软件开发工具包 (SDK) 结合使用。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon S3 控件

- [使用 AWS SDK 的 Amazon S3 Control 的基本功能示例](#)
 - [Hello Amazon S3 Control](#)
 - [了解使用 AWS SDK 的 Amazon S3 Control 的基本功能](#)
 - [使用 AWS SDK 对 Amazon S3 控件执行的操作](#)
 - [将 CreateJob 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteJobTagging 和 AWS SDK 搭配使用](#)
 - [将 DescribeJob 与 AWS SDK 或 CLI 配合使用](#)
 - [将 GetJobTagging 和 AWS SDK 搭配使用](#)
 - [将 PutJobTagging 和 AWS SDK 搭配使用](#)
 - [将 UpdateJobPriority 与 AWS SDK 或 CLI 配合使用](#)
 - [将 UpdateJobStatus 与 AWS SDK 或 CLI 配合使用](#)

使用 AWS SDK 的 Amazon S3 Control 的基本功能示例

以下代码示例展示了如何将 Amazon S3 Control 的基本功能与 AWS SDK 结合使用。

示例

- [Hello Amazon S3 Control](#)
- [了解使用 AWS SDK 的 Amazon S3 Control 的基本功能](#)
- [使用 AWS SDK 对 Amazon S3 控件执行的操作](#)
 - [将 CreateJob 与 AWS SDK 或 CLI 配合使用](#)
 - [将 DeleteJobTagging 和 AWS SDK 搭配使用](#)
 - [将 DescribeJob 与 AWS SDK 或 CLI 配合使用](#)
 - [将 GetJobTagging 和 AWS SDK 搭配使用](#)
 - [将 PutJobTagging 和 AWS SDK 搭配使用](#)
 - [将 UpdateJobPriority 与 AWS SDK 或 CLI 配合使用](#)
 - [将 UpdateJobStatus 与 AWS SDK 或 CLI 配合使用](#)

Hello Amazon S3 Control

以下代码示例展示了如何开始使用 Amazon S3 Control。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
    });
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListJobs](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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
```

- 有关 API 的详细信息，请参阅《适用于 Python 的 AWS SDK (Boto3) API 参考》中的 [ListJobs](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

了解使用 AWS SDK 的 Amazon S3 Control 的基本功能

以下代码示例演示如何了解 Amazon S3 Control 的核心操作。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

了解核心操作。

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

```

封装操作的操作类。

```

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();
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
 - [CreateJob](#)。
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub , 了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例 , 了解如何进行设置和运行。

了解 S3 批量基础知识场景。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。
 - [CreateJob](#)。
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 对 Amazon S3 控件执行的操作

以下代码示例演示了如何使用 AWS SDK 来执行各个 Amazon S3 控制操作。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅《[Amazon S3 Control API Reference](#)》。

示例

- [将 CreateJob 与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteJobTagging 和 AWS SDK 搭配使用](#)
- [将 DescribeJob 与 AWS SDK 或 CLI 配合使用](#)
- [将 GetJobTagging 和 AWS SDK 搭配使用](#)
- [将 PutJobTagging 和 AWS SDK 搭配使用](#)
- [将 UpdateJobPriority 与 AWS SDK 或 CLI 配合使用](#)
- [将 UpdateJobStatus 与 AWS SDK 或 CLI 配合使用](#)

将 **CreateJob** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateJob。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

CLI

AWS CLI

创建 Amazon S3 批量操作任务

以下 create-job 示例创建一个 Amazon S3 批量操作任务，将对象标记为 confidential 在 the bucket ``employee-records。

```
aws s3control create-job \
  --account-id 123456789012 \
  --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
    "Value":"true"}] }}' \
  --report '{"Bucket":"arn:aws:s3:::employee-records-logs","Prefix":"batch-op-
create-job",
    "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
  --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
    ["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
inv-report/7a6a9be4-072c-407e-85a2-
ec3e982f773e.csv","ETag":"69f52a4e9f797e987155d9c8f5880897"}}' \
  --priority 42 \
  --role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```


输出：

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CreateJob](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建异步 S3 作业。

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

```

创建合规性保留作业。

```

/**
 * 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();
    }

```

创建法定保留作业。

```

/**
 * 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();
    }

```

创建新的治理保留作业。

```

/**
 * 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();
}
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [CreateJob](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [CreateJob](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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.

```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [CreateJob](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteJobTagging** 和 AWS SDK 搭配使用

以下代码示例演示如何使用 DeleteJobTagging。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteJobTagging](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [DeleteJobTagging](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DeleteJobTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeJob** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeJob。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

CLI

AWS CLI

描述 Amazon S3 批量操作任务

以下 describe-job 命令可为指定的批量操作任务提供配置参数和状态。

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

输出：

```
{  
  "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"
  }
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [DescribeJob](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DescribeJob](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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

```

- 有关 API 的详细信息，请参阅《适用于 Python 的 AWS SDK (Boto3) API 参考》中的 [DescribeJob](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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.

```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [DescribeJob](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetJobTagging** 和 AWS SDK 搭配使用

以下代码示例演示如何使用 GetJobTagging。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

```
    });  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetJobTagging](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```


- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [GetJobTagging](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [GetJobTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutJobTagging** 和 AWS SDK 搭配使用

以下代码示例演示如何使用 PutJobTagging。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutJobTagging](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [PutJobTagging](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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.

```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [PutJobTagging](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateJobPriority** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 UpdateJobPriority。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

CLI

AWS CLI

更新 Amazon S3 批量操作任务的任务优先级

以下 update-job-priority 示例会将指定的任务更新为新的优先级。

```

aws s3control update-job-priority \
  --account-id 123456789012 \

```

```
--job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \  
--priority 52
```

输出：

```
{  
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",  
  "Priority": 52  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [UpdateJobPriority](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/**  
 * 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;
    }

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [UpdateJobPriority](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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
```


- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [UpdateJobPriority](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [UpdateJobPriority](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateJobStatus** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 UpdateJobStatus。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

CLI

AWS CLI

更新 Amazon S3 批量操作任务的状态

以下 update-job-status 示例取消正在等待批准的指定任务。

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled
```

输出：

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

以下 update-job-status 示例确认并运行正在等待批准的指定任务。

```
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"  
}
```

以下 update-job-status 示例取消正在运行的指定任务。

```
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"  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [UpdateJobStatus](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [UpdateJobStatus](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [UpdateJobStatus](#)。

SAP ABAP

适用于 SAP ABAP 的 SDK

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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.

```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API Reference》中的 [UpdateJobStatus](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 的 S3 目录存储桶代码示例

以下代码示例显示如何将 S3 目录存储桶与 AWS 软件开发工具包 (SDK) 结合使用。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务 服务结合来完成特定任务的代码示例。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

S3 目录存储桶

- [使用 AWS SDK 的 S3 目录存储桶基本功能示例](#)
 - [Hello Amazon S3 目录存储桶](#)
 - [通过 AWS SDK 了解 S3 目录存储桶基础功能](#)
 - [适用于使用 AWS SDK 的 S3 目录存储桶的操作](#)
 - [将 AbortMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CompleteMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CopyObject 和 AWS SDK 搭配使用](#)
 - [将 CreateBucket 和 AWS SDK 搭配使用](#)
 - [将 CreateMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CreateSession 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucket 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucketEncryption 和 AWS SDK 搭配使用](#)
 - [将 DeleteBucketPolicy 和 AWS SDK 搭配使用](#)
 - [将 DeleteObject 和 AWS SDK 搭配使用](#)
 - [将 DeleteObjects 和 AWS SDK 搭配使用](#)
 - [将 GetBucketEncryption 和 AWS SDK 搭配使用](#)

- [将 GetBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 GetObject 和 AWS SDK 搭配使用](#)
- [将 GetObjectAttributes 和 AWS SDK 搭配使用](#)
- [将 HeadBucket 和 AWS SDK 搭配使用](#)
- [将 HeadObject 和 AWS SDK 搭配使用](#)
- [将 ListDirectoryBuckets 和 AWS SDK 搭配使用](#)
- [将 ListMultipartUploads 和 AWS SDK 搭配使用](#)
- [将 ListObjectsV2 和 AWS SDK 搭配使用](#)
- [将 ListParts 和 AWS SDK 搭配使用](#)
- [将 PutBucketEncryption 和 AWS SDK 搭配使用](#)
- [将 PutBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 PutObject 和 AWS SDK 搭配使用](#)
- [将 UploadPart 和 AWS SDK 搭配使用](#)
- [将 UploadPartCopy 和 AWS SDK 搭配使用](#)
- [使用 AWS SDK 的 S3 目录存储桶场景](#)
 - [使用 AWS SDK 为 Amazon S3 目录存储桶创建获取对象的预签名 URL](#)

使用 AWS SDK 的 S3 目录存储桶基本功能示例

以下代码示例展示了如何将 Amazon S3 Directory Buckets 的基本功能与 AWS SDK 结合使用。

示例

- [Hello Amazon S3 目录存储桶](#)
- [通过 AWS SDK 了解 S3 目录存储桶基础功能](#)
- [适用于使用 AWS SDK 的 S3 目录存储桶的操作](#)
 - [将 AbortMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CompleteMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CopyObject 和 AWS SDK 搭配使用](#)
 - [将 CreateBucket 和 AWS SDK 搭配使用](#)
 - [将 CreateMultipartUpload 和 AWS SDK 搭配使用](#)
 - [将 CreateSession 和 AWS SDK 搭配使用](#)

- [将 DeleteBucket 和 AWS SDK 搭配使用](#)
- [将 DeleteBucketEncryption 和 AWS SDK 搭配使用](#)
- [将 DeleteBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 DeleteObject 和 AWS SDK 搭配使用](#)
- [将 DeleteObjects 和 AWS SDK 搭配使用](#)
- [将 GetBucketEncryption 和 AWS SDK 搭配使用](#)
- [将 GetBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 GetObject 和 AWS SDK 搭配使用](#)
- [将 GetObjectAttributes 和 AWS SDK 搭配使用](#)
- [将 HeadBucket 和 AWS SDK 搭配使用](#)
- [将 HeadObject 和 AWS SDK 搭配使用](#)
- [将 ListDirectoryBuckets 和 AWS SDK 搭配使用](#)
- [将 ListMultipartUploads 和 AWS SDK 搭配使用](#)
- [将 ListObjectsV2 和 AWS SDK 搭配使用](#)
- [将 ListParts 和 AWS SDK 搭配使用](#)
- [将 PutBucketEncryption 和 AWS SDK 搭配使用](#)
- [将 PutBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 PutObject 和 AWS SDK 搭配使用](#)
- [将 UploadPart 和 AWS SDK 搭配使用](#)
- [将 UploadPartCopy 和 AWS SDK 搭配使用](#)

Hello Amazon S3 目录存储桶

以下代码示例演示如何开始使用 Amazon S3 目录存储桶。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 AWS SDK 了解 S3 目录存储桶基础功能

以下代码示例演示了如何：

- 设置 VPC 和 VPC 端点。
- 设置策略、角色和用户，以使用 S3 目录存储桶和 S3 Express One Zone 存储类别。
- 创建两个 S3 客户端。
- 创建两个存储桶。
- 创建一个对象并复制它。
- 演示性能差异。
- 填充存储桶以显示按字典顺序的差异。
- 提示用户查看他们是否要清除资源。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行一个交互式场景，演示 Amazon S3 的功能。

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
    UUID.randomUUID();

    private static String regularUser = "";
    private static String vpcId = "";
    private static String expressUser = "";

    private static String vpcEndpointId = "";

    private static final S3DirectoriesActions s3DirectoriesActions = new
    S3DirectoriesActions();

    public static void main(String[] args) {
        try {
            s3ExpressScenario();
        } catch (RuntimeException e) {
```

```
        logger.info(e.getMessage());
    }
}

// Runs the scenario.
private static void s3ExpressScenario() {
    logger.info(DASHES);
    logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
for Java V2.");
    logger.info("""
        Let's get started! First, please note that S3 Express One Zone works
        best when working within the AWS infrastructure,
        specifically when working in the same Availability Zone (AZ). To see
        the best results in this example and when you implement
        directory buckets into your infrastructure, it is best to put your
        compute resources in the same AZ as your directory
        bucket.
        """);
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    // Create an optional VPC and create 2 IAM users.
    UserNames userNames = createVpcUsers();
    String expressUserName = userNames.getExpressUserName();
    String regularUserName = userNames.getRegularUserName();

    // Set up two S3 clients, one regular and one express,
    // and two buckets, one regular and one directory.
    setupClientsAndBuckets(expressUserName, regularUserName);

    // Create an S3 session for the express S3 client and add objects to the
    buckets.
    logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
    waitForInputToContinue(scanner);
    String bucketObject = createSessionAddObjects();

    // Demonstrate performance differences between regular and directory
    buckets.
    demonstratePerformance(bucketObject);

    // Populate the buckets to show the lexicographical difference between
    // regular and express buckets.
    showLexicographicalDifferences(bucketObject);
}
```



```

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /**
     Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {

```

```

        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
            becomes.

            This structural difference also causes `ListObject` operations to
            behave
            differently, which can cause unexpected results. Let's add a few
            more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        // Populate a few more files in each bucket so that we can use
        // ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();

```

```

        s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchBucketException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    try {
        // List objects in both S3 buckets.
        List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

        logger.info("Directory bucket content");
        for (String obj : dirBucketObjects) {
            logger.info(obj);
        }

        logger.info("Regular bucket content");
        for (String obj : regBucketObjects) {
            logger.info(obj);
        }
    } catch (CompletionException e) {
        logger.error("Async operation failed: {} ",
e.getCause().getMessage());
        return;
    }

    logger.info("""
        Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
        because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually

```

```

        creates directories and uses the object "key" as a path to the
object.
        """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from
a directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.

            Note: the performance difference will be much more pronounced if this
            example is run in an EC2 instance in the same Availability Zone as
            the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");
    }

```

```
// Ask if the user wants to download a different number.
logger.info("Would you like to download the file a different number of
times? (y/n): ");
String response = scanner.next().trim().toLowerCase();
if (response.equals("y")) {
    int maxDownloads = 1_000_000;

    // Ask for a valid number of downloads.
    while (true) {
        logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
        if (scanner.hasNextInt()) {
            downloads = scanner.nextInt();
            if (downloads >= 1 && downloads <= maxDownloads) {
                break;
            } else {
                logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
            }
        } else {
            logger.info("Invalid input. Please enter a valid integer.");
            scanner.next();
        }
    }

    logger.info("You have chosen to download {} items.", downloads);
} else {
    logger.info("No changes made. Using default downloads: {}",
downloads);
}
// Simulating the download process for the directory bucket.
logger.info("Downloading from the directory bucket.");
long directoryTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        // Get the object from the directory bucket.
        s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
```

```

        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}
long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

// Download from the regular bucket.
logger.info("Downloading from the regular bucket.");
long normalTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}

long normalTimeDifference = System.nanoTime() - normalTimeStart;
logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
long difference = normalTimeDifference - directoryTimeDifference;
logger.info("That's a difference of " + difference + " nanoseconds, or");
logger.info(difference / 1_000_000_000.0 + " seconds.");

```

```

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAdd0Objects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with
permission to
            work with directory buckets.
        """);
    }

```

```

        It's important to remember the user permissions when interacting
with
        directory buckets. Instead of validating permissions on every call
as
        regular buckets do, directory buckets utilize the user credentials
and session
        token to validate. This allows for much faster connection speeds on
every call.
        For single calls, this is low, but for many concurrent calls
        this adds up to a lot of time saved.
        """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
        /*
         * Optionally create a VPC.
         * Create two IAM users, one with S3 Express One Zone permissions and one
without.
         */
        logger.info(DASHES);
        logger.info("""
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);
    }

```



```

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                ");
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            ");
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

```

```

/**
 * Creates users using AWS CloudFormation.
 *
 * @return a {@link Map} of String keys and String values representing the
stack outputs,
 * which may include user-related information such as user names and IDs.
 */
public static Map<String, String> createUsersUsingCDK(String stackName) {
    logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
    CloudFormationHelper.deployCloudFormationStack(stackName);
    return CloudFormationHelper.getStackOutputsAsync(stackName).join();
}

/**
 * Sets up the necessary clients and buckets for the S3 Express service.
 *
 * @param expressUserName the username for the user with S3 Express
permissions
 * @param regularUserName the username for the user with regular S3
permissions
 */
public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
    Scanner locscanner = new Scanner(System.in);
    String accessKeyIdforRegUser;
    String secretAccessforRegUser;
    try {
        CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
        accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
        secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IamException) {
            logger.error("IamException occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
    }
    return;
}

```

```

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
    }

```

```

        return;
    }

    logger.info("""
        We can now use the ExpressUser client to make calls to S3 Express
        operations.
        """);
    waitForInputToContinue(locscanner);
    logger.info(DASHES);
    logger.info("""
        4. Create two buckets.
        Now we will create a directory bucket which is the linchpin of the S3
        Express One Zone service. Directory buckets
        behave differently from regular S3 buckets which we will explore
        here. We'll also create a regular bucket, put
        an object into the regular bucket, and copy it to the directory
        bucket.
        """);

    logger.info("""
        Now, let's choose an availability zone (AZ) for the directory
        bucket.
        We'll choose one that is supported.
        """);
    String zoneId;
    String regularBucketName;
    try {
        zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
        regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof Ec2Exception) {
            logger.error("EC2Exception occurred: {}", cause.getMessage(),
            ce);
        } else {
            logger.error("An unexpected error occurred: {}",
            cause.getMessage(), ce);
        }
        return;
    }
    logger.info("""
        Now, let's create the actual directory bucket, as well as a regular
        bucket."

```

```

        "");

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**

```

```

    * Creates an asynchronous S3 client with the specified access key and secret
    access key.
    *
    * @param accessKeyId      the AWS access key ID
    * @param secretAccessKey the AWS secret access key
    * @return a {@link CompletableFuture} that asynchronously creates the S3
    client
    * @throws IllegalArgumentException if the access key ID or secret access key
    is null
    */
    public static CompletableFuture<S3AsyncClient>
    createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
            == null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret
            Access Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
            AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

            .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();

        });
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}

```

```
    }  
}
```

Amazon S3 SDK 方法的包装器类。

```
public class S3DirectoriesActions {  
  
    private static IamAsyncClient iamAsyncClient;  
  
    private static Ec2AsyncClient ec2AsyncClient;  
    private static final Logger logger =  
        LoggerFactory.getLogger(S3DirectoriesActions.class);  
  
    private static IamAsyncClient getIAMAsyncClient() {  
        if (iamAsyncClient == null) {  
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()  
                .maxConcurrency(100)  
                .connectionTimeout(Duration.ofSeconds(60))  
                .readTimeout(Duration.ofSeconds(60))  
                .writeTimeout(Duration.ofSeconds(60))  
                .build();  
  
            ClientOverrideConfiguration overrideConfig =  
                ClientOverrideConfiguration.builder()  
                    .apiCallTimeout(Duration.ofMinutes(2))  
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))  
                    .retryStrategy(RetryMode.STANDARD)  
                    .build();  
  
            iamAsyncClient = IamAsyncClient.builder()  
                .httpClient(httpClient)  
                .overrideConfiguration(overrideConfig)  
                .build();  
        }  
        return iamAsyncClient;  
    }  
  
    private static Ec2AsyncClient getEc2AsyncClient() {  
        if (ec2AsyncClient == null) {  
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()  
                .maxConcurrency(100)  
                .connectionTimeout(Duration.ofSeconds(60))
```

```

        .readTimeout(Duration.ofSeconds(60))
        .writeTimeout(Duration.ofSeconds(60))
        .build();

    ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))
        .apiCallAttemptTimeout(Duration.ofSeconds(90))
        .retryStrategy(RetryMode.STANDARD)
        .build();

    ec2AsyncClient = Ec2AsyncClient.builder()
        .httpClient(httpClient)
        .region(Region.US_WEST_2)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return ec2AsyncClient;
}

/**
 * Deletes the specified S3 bucket and all the objects within it
asynchronously.
 *
 * @param s3AsyncClient the S3 asynchronous client to use for the operations
 * @param bucketName the name of the S3 bucket to be deleted
 * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
 *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
 * @throws CompletionException if there was an error deleting the bucket or
its objects
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
    ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3AsyncClient.listObjectsV2(listRequest)
        .thenCompose(listResponse -> {
            if (!listResponse.contents().isEmpty()) {
                List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()

```



```

        .map(s3object ->
ObjectIdentifier.builder().key(s3object.key()).build())
        .collect(Collectors.toList());

        DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
        .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
        .build();

        return s3AsyncClient.deleteObjects(deleteRequest)
            .thenAccept(deleteResponse -> {
                if (!deleteResponse.errors().isEmpty()) {
                    deleteResponse.errors().forEach(error ->
                        logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                }
            });
    }
    return CompletableFuture.completedFuture(null);
})
.thenCompose(ignored -> {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
        return s3AsyncClient.deleteBucket(deleteBucketRequest);
    })
    .thenCompose(ignored -> {
        S3AsyncWaiter waiter = s3AsyncClient.waiter();
        HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
        return waiter.waitUntilBucketNotExists(headBucketRequest);
    })
    .whenComplete((ignored, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof S3Exception) {
                throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
            }
            throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
        }
    });
}

```

```

        }
        logger.info("Bucket deleted successfully: " + bucketName);
    });
}

/**
 * Lists the objects in an S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the objects to list
 * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
 */
public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
    ListObjectsV2Request request = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3Client.listObjectsV2(request)
        .thenApply(response -> response.contents().stream()
            .map(S3Object::key)
            .toList())
        .whenComplete((result, exception) -> {
            if (exception != null) {
                throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
            }
        });
}

/**
 * Retrieves an object from an Amazon S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName the unique identifier (key) of the object to retrieve
 * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
 */
public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)

```

```

        .bucket(bucketName)
        .build();

        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest,
            AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client          the S3 async client to use for the copy
operation
     * @param sourceBucket      the name of the source bucket
     * @param sourceKey         the key of the object to be copied in the source
bucket
     * @param destinationBucket the name of the destination bucket
     * @param destinationKey    the key of the copied object in the destination
bucket
     * @return a {@link CompletableFuture} that completes when the copy operation
is finished
     */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
    {
        CopyObjectRequest copyRequest = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
    }

```

```

        .whenComplete((ignored, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to copy object",
exception);
            }
        });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client the S3 asynchronous client to use for creating the
session
     * @param bucketName the name of the S3 bucket for which to create the
session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse>
createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });
    }

```

```

    });

}

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The asynchronous S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone        The Availability Zone where the bucket will be created
 * @throws CompletionException if there's an error creating the bucket
 */
public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
    logger.info("Creating bucket: " + bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone)
            .build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .createBucketConfiguration(bucketConfiguration)
        .build();

    return s3Client.createBucket(bucketRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
            }
        })
}

```

```

        throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
    }
    logger.info("Bucket created successfully with location: " +
response.location());
    });
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the bucket creation
 * @param bucketName  the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
 *         when the bucket is successfully created
 * @throws CompletionException if there's an error creating the bucket
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    return s3Client.createBucket(bucketRequest)
        .thenCompose(response -> {
            S3AsyncWaiter s3Waiter = s3Client.waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();
            return s3Waiter.waitUntilBucketExists(bucketRequestWait);
        })
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                }
            }
            logger.info(bucketName + " is ready");
        });
}

```

```

    });
}

/**
 * Uploads an object to an Amazon S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the upload
 * @param bucketName  the destination S3 bucket name
 * @param bucketObject the name of the object to be uploaded
 * @param text        the content to be uploaded as the object
 */
public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(bucketObject)
        .build();

    return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchBucketException) {
                    throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                }
            }
        });
}

/**
 * Creates an AWS IAM access key asynchronously for the specified user name.
 *
 * @param userName the name of the IAM user for whom to create the access key
 * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
 */
public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
    CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()

```

```

        .userName(userName)
        .build();

    return getIAMAsyncClient().createAccessKey(request)
        .whenComplete((response, exception) -> {
            if (response != null) {
                logger.info("Access Key Created.");
            } else {
                if (exception == null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof IamException) {
                        throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                    }
                }
            }
        });
    }

    /**
     * Asynchronously selects an Availability Zone ID from the available EC2
     zones.
     *
     * @return A {@link CompletableFuture} that resolves to the selected
     Availability Zone ID.
     * @throws CompletionException if an error occurs during the request or
     processing.
     */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
        DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return
null if no zones are found
                }
            })
    }

```



```

        List<String> zoneIds = zonesList.stream()
            .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
            .toList();

        return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
            .thenApply(selectedZone -> {
                // Return only the selected Zone ID (e.g., "usw2-az1").
                return selectedZone.zoneId();
            });
    })
    .whenComplete((result, exception) -> {
        if (exception == null) {
            if (result != null) {
                logger.info("Selected Availability Zone ID: " + result);
            } else {
                logger.info("No availability zone selected.");
            }
        } else {
            Throwable cause = exception.getCause();
            if (cause instanceof Ec2Exception) {
                throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
            }
            throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
        }
    });
}

/**
 * Prompts the user to select an Availability Zone from the given list.
 *
 * @param zonesList the list of Availability Zones
 * @param zoneIds the list of zone IDs
 * @return the selected Availability Zone
 */
private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
    Scanner scanner = new Scanner(System.in);
    int index = -1;

```

```

while (index < 0 || index >= zoneIds.size()) {
    logger.info("Select an availability zone:");
    IntStream.range(0, zoneIds.size()).forEach(i ->
        logger.info(i + ": " + zoneIds.get(i))
    );

    logger.info("Enter the number corresponding to your choice: ");
    if (scanner.hasNextInt()) {
        index = scanner.nextInt();
    } else {
        scanner.next();
    }
}

AvailabilityZone selectedZone = zonesList.get(index);
logger.info("You selected: " + selectedZone.zoneId());
return selectedZone;
}

/**
 * Asynchronously sets up a new VPC, including creating the VPC, finding the
 * associated route table, and
 * creating a VPC endpoint for the S3 service.
 *
 * @return a {@link CompletableFuture} that, when completed, contains a
 * AbstractMap with the
 *      VPC ID and VPC endpoint ID.
 */
public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
    String cidr = "10.0.0.0/16";
    CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
        .cidrBlock(cidr)
        .build();

    return getEc2AsyncClient().createVpc(vpcRequest)
        .thenCompose(vpcResponse -> {
            String vpcId = vpcResponse.vpc().vpcId();
            logger.info("VPC Created: {}", vpcId);

            Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .vpcIds(vpcId)

```

```

        .build();

        return waiter.waitUntilVpcAvailable(request)
            .thenApply(waiterResponse -> vpcId);
    })
    .thenCompose(vpcId -> {
        Filter filter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
            .filters(filter)
            .build();

        return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
            .thenApply(routeTablesResponse -> {
                if (routeTablesResponse.routeTables().isEmpty()) {
                    throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
                }
                String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                logger.info("Route table found: {}", routeTableId);
                return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
            });
    })
    .thenCompose(vpcAndRouteTable -> {
        String vpcId = vpcAndRouteTable.getKey();
        String routeTableId = vpcAndRouteTable.getValue();
        Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
        String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

        CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
            .vpcId(vpcId)
            .routeTableIds(routeTableId)
            .serviceName(serviceName)
            .build();
    });

```

```

        return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
            .thenApply(vpcEndpointResponse -> {
                String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                logger.info("VPC Endpoint created: {}", vpcEndpointId);
                return new AbstractMap.SimpleEntry<>(vpcId,
vpcEndpointId);
            });
    })
    .exceptionally(exception -> {
        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
        if (cause instanceof Ec2Exception) {
            logger.error("EC2 error during VPC setup: {}",
cause.getMessage(), cause);
            throw new CompletionException("EC2 error during VPC setup: "
+ cause.getMessage(), cause);
        }

        logger.error("VPC setup failed: {}", cause.getMessage(), cause);
        throw new CompletionException("VPC setup failed: " +
cause.getMessage(), cause);
    });
}
}
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

PHP

适用于 PHP 的 SDK

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行一个用于演示 Amazon S3 目录存储桶和 S3 Express One Zone 的基础知识的场景。

```
echo "\n";
echo "-----\n";
echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
echo "-----\n";

// Change these both of these values to use a different region/
availability zone.
$region = "us-west-2";
$az = "usw2-az1";

$this->s3Service = new S3Service(new S3Client(['region' => $region]));
$this->iamService = new IAMService(new IamClient(['region' => $region]));

$uuid = uniqid();

echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
resources in the same AZ as your Directory
bucket.\n
INTRO;

    pressEnter();
    // 1. Configure a gateway VPC endpoint. This is the recommended method to
allow S3 Express One Zone traffic without
    // the need to pass through an internet gateway or NAT device.
echo "\n";
```

```

        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
        is running in an EC2 instance in the same AZ as your Directory buckets will be.
        \n";

        $ec2Choice = testable_readline("Are you running this in an EC2 instance
        located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
        VPC Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it,
            and create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
            $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
            ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
            $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
            \n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
        only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();

```

```

    $file = file_get_contents(__DIR__ . "/../../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
    $result = $this->cloudFormationClient->createStack([
        'StackName' => $stackName,
        'TemplateBody' => $file,
        'Capabilities' => ['CAPABILITY_IAM'],
    ]);
    $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
['StackName' => $stackName]);
    try {
        $waiter->promise()->wait();
    } catch (CloudFormationException $caught){
        echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
    $this->resources['stackName'] = $stackName;
    $stackInfo = $this->cloudFormationClient->describeStacks([
        'StackName' => $result['StackId'],
    ]);

    $expressUserName = "";
    $regularUserName = "";
    foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
        if ($output['OutputKey'] == "RegularUser") {
            $regularUserName = $output['OutputValue'];
        }
        if ($output['OutputKey'] == "ExpressUser") {
            $expressUserName = $output['OutputValue'];
        }
    }
    $regularKey = $this->iamService->createAccessKey($regularUserName);
    $regularCredentials = new Credentials($regularKey['AccessKeyId'],
    $regularKey['SecretAccessKey']);
    $expressKey = $this->iamService->createAccessKey($expressUserName);
    $expressCredentials = new Credentials($expressKey['AccessKeyId'],
    $expressKey['SecretAccessKey']);

    // 3. Create an additional client using the credentials with S3 Express
    permissions.
    echo "\n";
    echo "3. Create an additional client using the credentials with S3
    Express permissions.\n";

```

```

    echo "This client is created with the credentials associated with the
    user account with the S3 Express policy attached, so it can perform S3 Express
    operations.\n";
    pressEnter();
    $s3RegularClient = new S3Client([
        'Region' => $region,
        'Credentials' => $regularCredentials,
    ]);
    $s3RegularService = new S3Service($s3RegularClient);
    $s3ExpressClient = new S3Client([
        'Region' => $region,
        'Credentials' => $expressCredentials,
    ]);
    $s3ExpressService = new S3Service($s3ExpressClient);
    echo "All the roles and policies were created an attached to the user.
    Then, a new S3 Client and Service were created using that user's credentials.
    \n";

    echo "We can now use this client to make calls to S3 Express operations.
    Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
    Express.\n";
    pressEnter();

    // 4. Create two buckets.
    echo "\n";
    echo "3. Create two buckets.\n";
    echo "Now we will create a Directory bucket, which is the linchpin of the
    S3 Express One Zone service.\n";
    echo "Directory buckets behave in different ways from regular S3 buckets,
    which we will explore here.\n";
    echo "We'll also create a normal bucket, put an object into the normal
    bucket, and copy it over to the Directory bucket.\n";
    pressEnter();

    // Create a directory bucket. These are different from normal S3 buckets
    in subtle ways.
    $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
    echo "Now, let's create the actual Directory bucket, as well as a regular
    bucket.\n";
    pressEnter();
    $s3ExpressService->createBucket($directoryBucketName, [
        'CreateBucketConfiguration' => [
            'Bucket' => [

```



```

        'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
        'DataRedundancy' => "SingleAvailabilityZone",
    ],
    'Location' => [
        'Name' => $az,
        'Type' => "AvailabilityZone",
    ],
],
]);
$this->resources['directoryBucketName'] = $directoryBucketName;

// Create a normal bucket.
$normalBucketName = "normal-bucket-$uuid";
$s3RegularService->createBucket($normalBucketName);
$this->resources['normalBucketName'] = $normalBucketName;
echo "Great! Both buckets were created.\n";
pressEnter();

// 5. Create an object and copy it over.
echo "\n";
echo "5. Create an object and copy it over.\n";
echo "We'll create a basic object consisting of some text and upload it
to the normal bucket.\n";
echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
pressEnter();

$objectKey = "basic-text-object";
$s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
$this->resources['objectKey'] = $objectKey;

// Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
$s3ExpressService->createSession($directoryBucketName);
$s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";

```

```
    echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";
    echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
    pressEnter();

    // 6. Demonstrate performance difference.
    echo "\n";
    echo "6. Demonstrate performance difference.\n";
    $downloads = 1000;
    echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
    $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
    if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
        $downloads = $downloadChoice;
    }

    // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
    $directoryStartTime = hrtime(true);
    for($i = 0; $i < $downloads; ++$i){
        $s3ExpressService->getObject($directoryBucketName, $objectKey);
    }
    $directoryEndTime = hrtime(true);
    $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

    $normalStartTime = hrtime(true);
    for($i = 0; $i < $downloads; ++$i){
        $s3RegularService->getObject($normalBucketName, $objectKey);
    }
    $normalEndTime = hrtime(true);
    $normalTimeDiff = $normalEndTime - $normalStartTime;

    echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
```

```

    echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
    " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
    seconds.\n";
    pressEnter();

    // 7. Populate the buckets to show the lexicographical difference.
    echo "\n";
    echo "7. Populate the buckets to show the lexicographical difference.\n";
    echo "Now let's explore how Directory buckets store objects in a
different manner to regular buckets.\n";
    echo "The key is in the name \"Directory!\".\n";
    echo "Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual directories/folders.\n";
    echo "This allows for more rapid indexing, traversing, and therefore
retrieval times!\n";
    echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
    echo "This structural difference also causes ListObjects to behave
differently, which can cause unexpected results.\n";
    echo "Let's add a few more objects with layered directories as see how
the output of ListObjects changes.\n";
    pressEnter();

    // Populate a few more files in each bucket so that we can use
ListObjects and show the difference.
    $otherObject = "other/$objectKey";
    $altObject = "alt/$objectKey";
    $otherAltObject = "other/alt/$objectKey";
    $s3ExpressService->putObject($directoryBucketName, $otherObject);
    $s3RegularService->putObject($normalBucketName, $otherObject);
    $this->resources['otherObject'] = $otherObject;
    $s3ExpressService->putObject($directoryBucketName, $altObject);
    $s3RegularService->putObject($normalBucketName, $altObject);
    $this->resources['altObject'] = $altObject;
    $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
    $s3RegularService->putObject($normalBucketName, $otherAltObject);
    $this->resources['otherAltObject'] = $otherAltObject;

    $listDirectoryBucket = $s3ExpressService-
>listObjects($directoryBucketName);
    $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

    // Directory bucket content
    echo "Directory bucket content\n";

```

```

        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical
order, while the directory bucket does not. This is because the normal bucket
considers the whole \"key\" to be the object identifies, while the directory
bucket actually creates directories and uses the object \"key\" as a path to the
object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }
    }

namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)

```

```
{
    if ($client) {
        $this->client = $client;
    } else {
        $this->client = new S3Client([
            'version' => 'latest',
            'region' => 'us-west-2',
        ]);
    }
    $this->verbose = $verbose;
}

public function setVerbose($verbose)
{
    $this->verbose = $verbose;
}

public function isVerbose(): bool
{
    return $this->verbose;
}

public function getClient(): S3Client
{
    return $this->client;
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
```

```

        echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
        echo "\nPlease fix error with bucket deletion before continuing.
\n";
    }
    throw $exception;
}

}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {

```

```

        echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
        echo "Please fix error with object uploading before continuing.";
    }
    throw $exception;
}
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";

```

```

    }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with list objects before continuing.";
        }
        throw $exception;
    }
    return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {

```



```

        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }

    $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
    try {
        $this->client->deleteObjects($parameters);
        if ($this->verbose) {
            echo "Deleted the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}
}

```

```
public function deleteBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->deleteBucket($parameters);
        if ($this->verbose) {
            echo "Deleted the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function listBuckets(array $args = [])
{

```

```

        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }

    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
$options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }

    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
        }
    }

```

```
        return $result;
    }catch(S3Exception $caught){
        if($caught->getAwsErrorType() == "NoSuchBucket"){
            echo "The specified bucket does not exist.";
        }
        throw $caught;
    }
}
```

- 有关 API 详细信息，请参阅《适用于 PHP 的 AWS SDK API Reference》中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

Python

适用于 Python 的 SDK (Boto3)

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行一个用于演示 Amazon S3 目录存储桶和 S3 Express One Zone 的基础知识的场景。

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
    Express."""

    def __init__(
```

```

        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
        print("")
        print_dashes()
        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
3)!")
        print_dashes()
        print(
            """
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example and when you implement
Directory buckets into your infrastructure, it is best to put your compute
resources in the same AZ as your Directory
bucket.
            """
        )
        press_enter_to_continue()

        # Create an optional VPC and create 2 IAM users.
        express_user_name, regular_user_name = self.create_vpc_and_users()

        # Set up two S3 clients, one regular and one express, and two buckets,
        one regular and one express.

```

```

        self.setup_clients_and_buckets(express_user_name, regular_user_name)

        # Create an S3 session for the express S3 client and add objects to the
        buckets.
        bucket_object = self.create_session_and_add_objects()

        # Demonstrate performance differences between regular and express
        buckets.
        self.demonstrate_performance(bucket_object)

        # Populate the buckets to show the lexicographical difference between
        regular and express buckets.
        self.show_lexicographical_differences(bucket_object)

        print("")
        print("That's it for our tour of the basic operations for S3 Express One
        Zone.")

        if q.ask(
            "Would you like to delete all the resources created during this demo
            (y/n)? ",
            q.is_yesno,
        ):
            self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
        without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
        allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(

```

```

        "Great! Let's set up a VPC, retrieve the Route Table from it, and
        create a VPC Endpoint to connect the S3 Client to."
    )
    self.setup_vpc()
    press_enter_to_continue()
else:
    print("Skipping the VPC setup. Don't forget to use this in
production!")
    print(
        """

```

2. Policies, users, and roles with CDK.

Now, we'll set up some policies, roles, and a user. This user will only have permissions to do S3 Express One Zone actions.

```

        """
    )
    press_enter_to_continue()
    stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
    template_as_string = S3ExpressScenario.get_template_as_string()
    self.stack = self.deploy_cloudformation_stack(stack_name,
template_as_string)
    regular_user_name = None
    express_user_name = None
    outputs = self.stack.outputs
    for output in outputs:
        if output.get("OutputKey") == "RegularUser":
            regular_user_name = output.get("OutputValue")
        elif output.get("OutputKey") == "ExpressUser":
            express_user_name = output.get("OutputValue")
    if not regular_user_name or not express_user_name:
        error_string = f"""
Failed to retrieve required outputs from CloudFormation stack.
'regular_user_name'={regular_user_name},
'express_user_name'={express_user_name}
        """
        logger.error(error_string)
        raise ValueError(error_string)
    return express_user_name, regular_user_name

def setup_clients_and_buckets(
    self, express_user_name: str, regular_user_name: str
) -> None:
    """
    Set up two S3 clients, one regular and one express, and two buckets, one
    regular and one express.

```

```

        :param express_user_name: The name of the user with S3 Express
permissions.
        :param regular_user_name: The name of the user with regular S3
permissions.
        """
        regular_credentials = self.create_access_key(regular_user_name)
        express_credentials = self.create_access_key(express_user_name)
        # 3. Create an additional client using the credentials with S3 Express
permissions.
        print(
            """
3. Create an additional client using the credentials with S3 Express permissions.
This client is created with the
credentials associated with the user account with the S3 Express policy attached,
so it can perform S3 Express operations.
"""
        )
        press_enter_to_continue()
        s3_regular_client = self.create_s3_client_with_access_key_credentials(
            regular_credentials
        )
        self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
        s3_express_client = self.create_s3_client_with_access_key_credentials(
            express_credentials
        )
        self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
        print(
            """
All the roles and policies were created and attached to the user. Then a new S3
Client were created using
that user's credentials. We can now use this client to make calls to S3 Express
operations. Keeping permissions in mind
(and adhering to least-privilege) is crucial to S3 Express.
"""
        )
        press_enter_to_continue()
        # 4. Create two buckets.
        print(
            """
3. Create two buckets.
Now we will create a Directory bucket which is the linchpin of the S3 Express One
Zone service. Directory buckets
behave in different ways from regular S3 buckets which we will explore here.
We'll also create a normal bucket, put

```



```

an object into the normal bucket, and copy it over to the Directory bucket.
"""

    )

    # Create a directory bucket. These are different from normal S3 buckets
    in subtle ways.
    bucket_prefix = q.ask(
        "Enter a bucket name prefix that will be used for both buckets: ",
        q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
    )

    # Some availability zones are not supported for Directory buckets. We'll
    choose one that is supported.
    print(
        "Now, let's choose an availability zone for the Directory bucket.
    We'll choose one that is supported."
    )
    while True:
        availability_zone = self.select_availability_zone_id(self.region)
        # Construct the parts of a directory bucket name that is made unique
        with a UUID string.
        directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
        max_uuid_length = 63 - len(bucket_prefix) -
len(directory_bucket_suffix) - 1
        bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
        directory_bucket_name = (
            f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
        )
        regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
        configuration = {
            "Bucket": {
                "Type": "Directory",
                "DataRedundancy": "SingleAvailabilityZone",
            },
            "Location": {
                "Name": availability_zone["ZoneId"],
                "Type": "AvailabilityZone",
            },
        }
        press_enter_to_continue()
        print(
            "Now, let's create the actual Directory bucket, as well as a
        regular bucket."
        )

```

```

        press_enter_to_continue()
    try:
        self.s3_express_wrapper.create_bucket(
            directory_bucket_name, configuration
        )
        break
    except ClientError as client_error:
        if client_error.response["Error"]["Code"] == "InvalidBucketName":
            print(
                f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
            )
            if q.ask(
                "Would you like to select a different availability zone?",
                q.is_yesno,
            ):
                continue
            else:
                raise
        else:
            raise
    print(f"Created directory bucket, '{directory_bucket_name}'")
    self.directory_bucket_name = directory_bucket_name

    self.s3_regular_wrapper.create_bucket(regular_bucket_name)
    print(f"Created regular bucket, '{regular_bucket_name}'")
    self.regular_bucket_name = regular_bucket_name
    print("Great! Both buckets were created.")
    press_enter_to_continue()

def create_session_and_add_objects(self) -> None:
    """
    Create a session for the express S3 client and add objects to the
    buckets.
    """
    print(
        """

```

5. Create an object and copy it over.

We'll create a basic object consisting of some text and upload it to the normal bucket. Next we'll copy the object into the Directory bucket using the regular client. This works fine because copy operations are not restricted for Directory buckets.

```

        """
    )
    press_enter_to_continue()
    bucket_object = "basic-text-object"
    self.s3_regular_wrapper.put_object(
        self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
    )
    self.s3_express_wrapper.create_session(self.directory_bucket_name)
    self.s3_express_wrapper.copy_object(
        self.regular_bucket_name,
        bucket_object,
        self.directory_bucket_name,
        bucket_object,
    )
    print(
        """

```

It worked! It's important to remember the user permissions when interacting with Directory buckets. Instead of validating permissions on every call as normal buckets do, Directory buckets utilize the user credentials and session token to validate.

This allows for much faster connection speeds on every call. For single calls, this is low, but for many concurrent calls this adds up to a lot of time saved.

```

"""
    )
    press_enter_to_continue()
    return bucket_object

def demonstrate_performance(self, bucket_object: str) -> None:
    """
    Demonstrate performance differences between regular and Directory
    buckets.
    :param bucket_object: The name of the object to download from each
    bucket.
    """
    print("")
    print("6. Demonstrate performance difference.")
    print(
        """

```

Now, let's do a performance test. We'll download the same object from each bucket 'downloads' times and compare the total time needed. Note: the performance difference will be much more pronounced if this example is run in an EC2 instance in the same Availability Zone as the bucket.

```

"""
    )
    downloads = 1000
    print(
        f"The number of downloads of the same object for this example is set
at {downloads}."
    )
    if q.ask("Would you like to download a different number? (y/n) ",
q.is_yesno):
        max_downloads = 1000000
        downloads = q.ask(
            f"Enter a number between 1 and {max_downloads} for the number of
downloads: ",
            q.is_int,
            q.in_range(1, max_downloads),
        )
        # Download the object 'downloads' times from each bucket and time it to
demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")

                self.s3_express_wrapper.get_object(
                    self.directory_bucket_name, bucket_object
                )

        directory_time_difference = time.time_ns() - directory_time_start
        print("Downloading from the normal bucket.")
        normal_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")
                self.s3_regular_wrapper.get_object(self.regular_bucket_name,
bucket_object)

        normal_time_difference = time.time_ns() - normal_time_start
        print(
            f"The directory bucket took {directory_time_difference} nanoseconds,
while the normal bucket took {normal_time_difference}."
        )

```

```

        difference = normal_time_difference - directory_time_difference
        print(f"That's a difference of {difference} nanoseconds, or")
        print(f"{(difference) / 1000000000} seconds.")
        if difference < 0:
            print(
                "The directory buckets were slower. This can happen if you are
not running on the cloud within a vpc."
            )
        press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
        buckets.
        This is done by creating a few objects in each bucket and listing them to
        show the difference.
        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
7. Populate the buckets to show the lexicographical difference.
Now let's explore how Directory buckets store objects in a different manner to
regular buckets. The key is in the name
"Directory". Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual
directories/folders. This allows for more rapid indexing, traversing, and
therefore retrieval times! The more segmented
your bucket is, with lots of directories, sub-directories, and objects, the more
efficient it becomes. This structural
difference also causes ListObjects to behave differently, which can cause
unexpected results. Let's add a few more
objects with layered directories to see how the output of ListObjects changes.
"""
        )
        press_enter_to_continue()
        # Populate a few more files in each bucket so that we can use ListObjects
        and show the difference.
        other_object = f"other/{bucket_object}"
        alt_object = f"alt/{bucket_object}"
        other_alt_object = f"other/alt/{bucket_object}"
        self.s3_regular_wrapper.put_object(self.regular_bucket_name,
other_object, "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
other_object, "")

```

```

        self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
        "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
alt_object, "")
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, other_alt_object, ""
        )
        self.s3_express_wrapper.put_object(
            self.directory_bucket_name, other_alt_object, ""
        )
        directory_bucket_objects = self.s3_express_wrapper.list_objects(
            self.directory_bucket_name
        )

        regular_bucket_objects = self.s3_regular_wrapper.list_objects(
            self.regular_bucket_name
        )

        print("Directory bucket content")
        for bucket_object in directory_bucket_objects:
            print(f"    {bucket_object['Key']}")
        print("Normal bucket content")
        for bucket_object in regular_bucket_objects:
            print(f"    {bucket_object['Key']}")
        print(
            """
Notice how the normal bucket lists objects in lexicographical order, while the
directory bucket does not. This is
because the normal bucket considers the whole "key" to be the object identifier,
while the directory bucket actually
creates directories and uses the object "key" as a path to the object.
            """
        )
        press_enter_to_continue()

def cleanup(self) -> None:
    """
    Delete resources created by this scenario.
    """
    if self.directory_bucket_name is not None:
        self.s3_express_wrapper.delete_bucket_and_objects(
            self.directory_bucket_name
        )
        print(f"Deleted directory bucket, '{self.directory_bucket_name}'")

```

```

        self.directory_bucket_name = None

    if self.regular_bucket_name is not None:

self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
        print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
        self.regular_bucket_name = None

    if self.stack is not None:
        self.destroy_cloudformation_stack(self.stack)
        self.stack = None

    self.tear_done_vpc()

def create_access_key(self, user_name: str) -> dict[str, any]:
    """
    Creates an access key for the user.
    :param user_name: The name of the user.
    :return: The access key for the user.
    """
    try:
        access_key = self.iam_client.create_access_key(UserName=user_name)
        return access_key["AccessKey"]
    except ClientError as client_error:
        logging.error(
            "Couldn't create the access key. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def create_s3_client_with_access_key_credentials(
    self, access_key: dict[str, any]
) -> client:
    """
    Creates an S3 client with access key credentials.
    :param access_key: The access key for the user.
    :return: The S3 Express One Zone client.
    """
    try:
        s3_express_client = boto3.client(
            "s3",
            aws_access_key_id=access_key["AccessKeyId"],
            aws_secret_access_key=access_key["SecretAccessKey"],
            region_name=self.region,

```

```

        )
        return s3_express_client
    except ClientError as client_error:
        logging.error(
            "Couldn't create the S3 Express One Zone client. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def select_availability_zone_id(self, region: str) -> dict[str, any]:
    """
    Selects an availability zone.
    :param region: The region to select the availability zone from.
    :return: The availability zone dictionary.
    """
    try:
        response = self.ec2_client.describe_availability_zones(
            Filters=[{"Name": "region-name", "Values": [region]}]
        )
        availability_zones = response["AvailabilityZones"]
        zone_names = [zone["ZoneName"] for zone in availability_zones]
        index = q.choose("Select an availability zone: ", zone_names)
        return availability_zones[index]
    except ClientError as client_error:
        logging.error(
            "Couldn't describe availability zones. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def deploy_cloudformation_stack(
    self, stack_name: str, cfn_template: str
) -> ServiceResource:
    """
    Deploys prerequisite resources used by the scenario. The resources are
    defined in the associated `cfn_template.yaml` AWS CloudFormation script
    and are deployed
    as a CloudFormation stack, so they can be easily managed and destroyed.

    :param stack_name: The name of the CloudFormation stack.
    :param cfn_template: The CloudFormation template as a string.
    :return: The CloudFormation stack resource.
    """
    print(f"Deploying CloudFormation stack: {stack_name}.")

```



```

        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

    return stack

def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
    """
    Destroys the resources managed by the CloudFormation stack, and the
    CloudFormation
    stack itself.

    :param stack: The CloudFormation stack that manages the example
    resources.
    """
    try:
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may
            take a few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")
    except ClientError as client_error:
        logging.error(
            "Couldn't delete the CloudFormation stack. Here's why: %s",
            client_error.response["Error"]["Message"],
        )

    @staticmethod
    def get_template_as_string() -> str:

```

```

    """
    Returns a string containing this scenario's CloudFormation template.
    """
    script_directory = os.path.dirname(os.path.abspath(__file__))
    template_file_path = os.path.join(script_directory,
"s3_express_template.yaml")
    file = open(template_file_path, "r")
    return file.read()

def setup_vpc(self):
    cidr = "10.0.0.0/16"
    try:
        response = self.ec2_client.create_vpc(CidrBlock=cidr)
        self.vpc_id = response["Vpc"]["VpcId"]

        waiter = self.ec2_client.get_waiter("vpc_available")
        waiter.wait(VpcIds=[self.vpc_id])
        print(f"Created vpc {self.vpc_id}")

    except ClientError as client_error:
        logging.error(
            "Couldn't create the vpc. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise
    try:
        response = self.ec2_client.describe_route_tables(
            Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
        )
        route_table_id = response["RouteTables"][0]["RouteTableId"]
        service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

        response = self.ec2_client.create_vpc_endpoint(
            VpcId=self.vpc_id,
            RouteTableIds=[route_table_id],
            ServiceName=service_name,
        )
        self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
        print(f"Created vpc endpoint {self.vpc_endpoint_id}")

    except ClientError as client_error:
        logging.error(
            "Couldn't create the vpc endpoint. Here's why: %s",

```

```

        client_error.response["Error"]["Message"],
    )
    raise

def tear_down_vpc(self) -> None:
    if self.vpc_endpoint_id is not None:
        try:
            self.ec2_client.delete_vpc_endpoints(
                VpcEndpointIds=[self.vpc_endpoint_id]
            )
            print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
            self.vpc_endpoint_id = None
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the vpc endpoint %s. Here's why: %s",
                self.vpc_endpoint_id,
                client_error.response["Error"]["Message"],
            )
    if self.vpc_id is not None:
        try:
            self.ec2_client.delete_vpc(VpcId=self.vpc_id)
            print(f"Deleted vpc {self.vpc_id}")
            self.vpc_id = None
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the vpc %s. Here's why: %s",
                self.vpc_id,
                client_error.response["Error"]["Message"],
            )

```

Amazon S3 Express SDK 函数的包装器类。

```

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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

适用于使用 AWS SDK 的 S3 目录存储桶的操作

以下代码示例演示了如何使用 AWS 开发工具包来执行各个 S3 Directory Buckets 操作。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 S3 Directory Buckets API，是必须在上下文中运行的大型程序的代码节选。您可以在[使用 AWS SDK 的 S3 目录存储桶场景](#)中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅《[Amazon S3 Directory Buckets API Reference](#)》。

示例

- [将 AbortMultipartUpload 和 AWS SDK 搭配使用](#)
- [将 CompleteMultipartUpload 和 AWS SDK 搭配使用](#)
- [将 CopyObject 和 AWS SDK 搭配使用](#)
- [将 CreateBucket 和 AWS SDK 搭配使用](#)
- [将 CreateMultipartUpload 和 AWS SDK 搭配使用](#)
- [将 CreateSession 和 AWS SDK 搭配使用](#)

- [将 DeleteBucket 和 AWS SDK 搭配使用](#)
- [将 DeleteBucketEncryption 和 AWS SDK 搭配使用](#)
- [将 DeleteBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 DeleteObject 和 AWS SDK 搭配使用](#)
- [将 DeleteObjects 和 AWS SDK 搭配使用](#)
- [将 GetBucketEncryption 和 AWS SDK 搭配使用](#)
- [将 GetBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 GetObject 和 AWS SDK 搭配使用](#)
- [将 GetObjectAttributes 和 AWS SDK 搭配使用](#)
- [将 HeadBucket 和 AWS SDK 搭配使用](#)
- [将 HeadObject 和 AWS SDK 搭配使用](#)
- [将 ListDirectoryBuckets 和 AWS SDK 搭配使用](#)
- [将 ListMultipartUploads 和 AWS SDK 搭配使用](#)
- [将 ListObjectsV2 和 AWS SDK 搭配使用](#)
- [将 ListParts 和 AWS SDK 搭配使用](#)
- [将 PutBucketEncryption 和 AWS SDK 搭配使用](#)
- [将 PutBucketPolicy 和 AWS SDK 搭配使用](#)
- [将 PutObject 和 AWS SDK 搭配使用](#)
- [将 UploadPart 和 AWS SDK 搭配使用](#)
- [将 UploadPartCopy 和 AWS SDK 搭配使用](#)

将 **AbortMultipartUpload** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 AbortMultipartUpload。

Java

适用于 Java 的 SDK 2.x

 Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

中止目录存储桶中的分段上传。

```
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;  
    }  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [AbortMultipartUpload](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CompleteMultipartUpload** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 CompleteMultipartUpload。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

完成目录存储桶中的分段上传。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [CompleteMultipartUpload](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CopyObject** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 CopyObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

将对象从目录存储桶复制到目录存储桶。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [CopyObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateBucket** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 CreateBucket。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建 S3 目录存储桶。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [CreateBucket](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateMultipartUpload** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 CreateMultipartUpload。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

在目录存储桶中创建分段上传。

```
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;
    }
}
```

```
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [CreateMultipartUpload](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateSession** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 `CreateSession`。

Python

适用于 Python 的 SDK (Boto3)

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 [CreateSession](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucket** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 DeleteBucket。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除 S3 目录存储桶。

```
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;  
    }  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteBucket](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketEncryption** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 DeleteBucketEncryption。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除目录存储桶的加密配置。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteBucketEncryption](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketPolicy** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 DeleteBucketPolicy。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除目录存储桶的存储桶策略。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteBucketPolicy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObject** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 DeleteObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除目录存储桶中的一个对象。

```
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;
    }
}
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [DeleteObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObjects** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 DeleteObjects。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

删除目录存储桶中的多个对象。

```
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;
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 [DeleteObjects](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketEncryption** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 GetBucketEncryption。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取目录存储桶的加密配置。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetBucketEncryption](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketPolicy** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 GetBucketPolicy。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取目录存储桶的策略。

```
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;
    }
}
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetBucketPolicy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObject** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 GetObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [创建预签名 URL 以获取对象](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取目录存储桶中的一个对象。

```
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;
        }
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `GetObjectAttributes` 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 `GetObjectAttributes`。

Java

适用于 Java 的 SDK 2.x

Note

查看 [GitHub](#)，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取目录存储桶中的对象属性。

```
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;
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetObjectAttributes](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **HeadBucket** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 HeadBucket。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

检查指定的 S3 目录存储桶是否存在并且可供访问。

```
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;
}
}
```

- 有关 API 详细，请参阅《AWS SDK for Java 2.x API Reference》中的 [HeadBucket](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **HeadObject** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 HeadObject。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取目录存储桶中对象的元数据。

```
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;
    }
}
```

- 有关 API 详细，请参阅《AWS SDK for Java 2.x API Reference》中的 [HeadObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListDirectoryBuckets** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 ListDirectoryBuckets。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出所有目录存储桶。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListDirectoryBuckets](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListMultipartUploads** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 ListMultipartUploads。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出目录存储桶中的分段上传。

```
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
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListMultipartUploads](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListObjectsV2** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 ListObjectsV2。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出目录存储桶中的对象。

```
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;
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListObjectsV2](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListParts** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 ListParts。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出目录存储桶中分段上传的分段。

```
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
    }
}

```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [ListParts](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketEncryption** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 PutBucketEncryption。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

为目录存储桶设置存储桶加密。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketEncryption](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketPolicy** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 PutBucketPolicy。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

将存储桶策略应用于目录存储桶。

```
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;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutBucketPolicy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObject** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 PutObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [了解基本功能](#)

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

将对象放入目录存储桶。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [PutObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UploadPart** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 UploadPart。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

上传目录存储桶的分段上传的分段。

```
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;
}
```



```
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [UploadPart](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UploadPartCopy** 和 AWS SDK 搭配使用

以下代码示例演示了如何使用 UploadPartCopy。

Java

适用于 Java 的 SDK 2.x

 Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

根据源对象大小创建复制分段，然后将各个分段复制到目录存储桶。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [UploadPartCopy](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 的 S3 目录存储桶场景

以下代码示例显示如何通过 AWS SDK 实施 S3 Directory Buckets 中的常见场景。这些场景向您展示了如何通过调用 S3 Directory Buckets 中的多个函数或与其它 AWS 服务 结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- [使用 AWS SDK 为 Amazon S3 目录存储桶创建获取对象的预签名 URL](#)

使用 AWS SDK 为 Amazon S3 目录存储桶创建获取对象的预签名 URL

以下代码示例演示如何为 S3 目录存储桶创建预签名 URL 以及获取对象。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

生成用于访问 S3 目录存储桶中的对象的预签名 GET URL。

```
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;
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 [GetObject](#)。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 [使用 AWS SDK 通过 Amazon S3 进行开发](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

对请求进行身份验证 (AWS 签名版本 4)

主题

- [身份验证方法](#)
- [对请求进行签名简介](#)
- [对请求进行身份验证：使用 Authorization 标头 \(AWS 签名版本 4 \)](#)
- [对请求进行身份验证：使用查询参数 \(AWS 签名版本 4 \)](#)
- [示例：AWS 签名版本 4 中的签名计算](#)
- [对请求进行身份验证：使用 POST 的基于浏览器的上传 \(AWS 签名版本 4 \)](#)
- [特定于 Amazon S3 签名版本 4 身份验证的策略键](#)

与 Amazon S3 的每一次交互都是经身份验证的或匿名的。本节介绍使用 AWS 签名版本 4 算法进行请求身份验证。

Note

如果您使用 AWS SDK (请参阅[示例代码和库](#)) 发送请求，则不需要阅读本节，因为 SDK 客户端会使用您提供的访问密钥对请求进行身份验证。您应始终使用 AWS SDK，除非您有充分的理由不这样做。在支持这两个签名版本的区域中，您可以请求 AWS SDK 使用特定的签名版本。有关更多信息，请参阅[使用 AWS 开发工具包发送经身份验证的请求](#)。只有在自定义客户端中实施 AWS 签名版本 4 算法时，才需要阅读本节。

使用 AWS 签名版本 4 进行身份验证可提供以下部分或全部功能，具体取决于您选择如何对请求进行签名：

- 验证请求者的身份：经过身份验证的请求需要您使用访问密钥 (访问密钥 ID、秘密访问密钥) 创建的签名。有关获取访问密钥的信息，请参阅《AWS 一般参考》中的[了解和获取安全凭证](#)。如果您使用的是临时安全凭证，则签名计算还需要安全令牌。有关更多信息，请参阅《IAM 用户指南》中的[请求临时安全凭证](#)。
- 传输中数据保护：为了防止在传输请求的过程中篡改请求，您可以使用一些请求元素来计算请求签名。收到请求后，Amazon S3 会使用相同的请求元素计算签名。如果 Amazon S3 收到的任何请求组件与用于计算签名的组件不匹配，Amazon S3 将拒绝该请求。

- 防止重复使用请求的已签名部分：请求的已签名部分（使用 AWS 签名）在请求中时间戳的 15 分钟内有效。有权访问已签名请求的未经授权方可以在 15 分钟时段内修改请求中未签名的部分，而不会影响请求的有效性。因此，我们建议您通过对请求标头和正文进行签名、向 Amazon S3 发出 HTTPS 请求以及使用 AWS 策略中的 `s3:x-amz-content-sha256` 条件键（请参阅[特定于 Amazon S3 签名版本 4 身份验证的策略键](#)），来要求用户对 Amazon S3 请求正文进行签名，从而最大限度地提供保护。

Note

Amazon S3 在所有 AWS 区域中都支持签名版本 4，这是一种用于对 AWS 服务的入站 API 请求进行身份验证的协议。目前，于 2014 年 1 月 30 日前创建的 AWS 区域 将继续支持之前的协议：Signature Version 2。在 2014 年 1 月 30 日后创建的所有新区域将只支持签名版本 4，因此，发往这些区域的所有请求都必须采用签名版本 4。有关 AWS 签名版本 2 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[对 REST 请求进行签名和身份验证](#)。

身份验证方法

您可以使用以下方法之一表达身份验证信息：

- HTTP Authorization 标头：使用 HTTP Authorization 标头是对 Amazon S3 请求进行身份验证的最常用方法。所有 Amazon S3 REST 操作（使用 POST 请求的基于浏览器的上传除外）都需要此标头。有关 Authorization 标头值以及如何计算签名和相关选项的更多信息，请参阅[对请求进行身份验证：使用 Authorization 标头 \(AWS 签名版本 4\)](#)。
- 查询字符串参数：您可以使用查询字符串在 URL 中完整表达请求。在这种情况下，您可以使用查询参数来提供请求信息，包括身份验证信息。由于请求签名是 URL 的一部分，因此这类 URL 通常称为预签名 URL。您可以使用预签名 URL 在 HTML 中嵌入可单击链接，该链接的有效期最长为七天。有关更多信息，请参阅[对请求进行身份验证：使用查询参数 \(AWS 签名版本 4\)](#)。

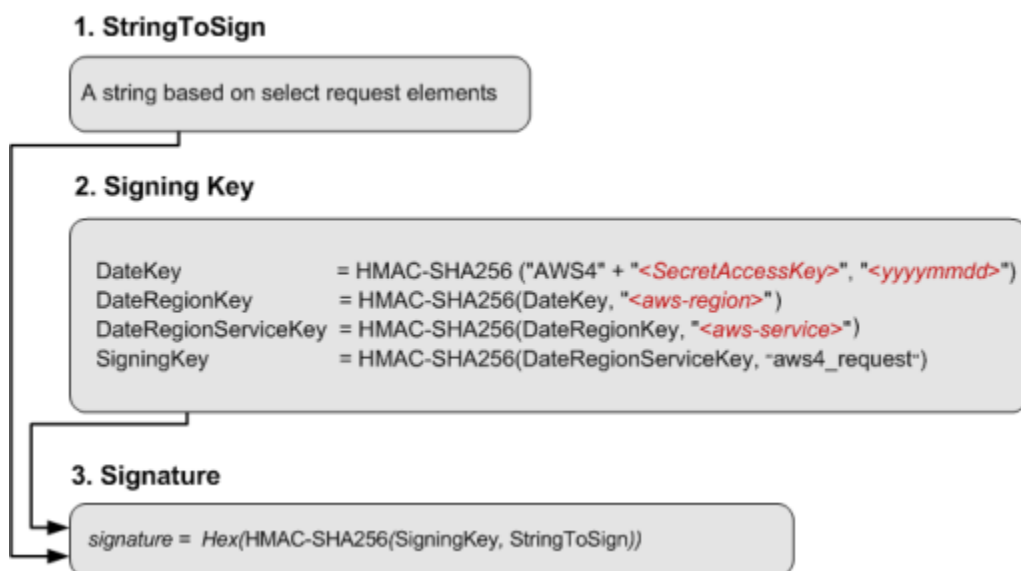
Amazon S3 还支持使用 HTTP POST 请求的基于浏览器的上传。通过 HTTP POST 请求，您可以直接从浏览器将内容上传到 Amazon S3。有关对 POST 请求进行身份验证的信息，请参阅[使用 POST 的基于浏览器的上传 \(AWS 签名版本 4\)](#)。

对请求进行签名简介

您在请求中发送的身份验证信息必须包含签名。要计算签名，首先要将 select 请求元素连接形成一个字符串，称为待签字符串。然后，您可以使用签名密钥来计算待签字符串的 HMAC 散列消息认证码。

在 AWS 签名版本 4 中，秘密访问密钥不用于对请求进行签名。相反，您可以使用秘密访问密钥派生签名密钥。派生的签名密钥特定于日期、服务和区域。有关如何在不同的编程语言中派生签名密钥的更多信息，请参阅[如何为签名版本 4 派生签名密钥的示例](#)。

下图说明了计算签名的一般过程。



待签字符串取决于请求类型。例如，使用 HTTP 授权标头或查询参数进行身份验证时，可以使用不同的请求元素组合来创建待签字符串。对于 HTTP POST 请求，请求中的 POST 策略是您签名的字符串。有关计算待签字符串的更多信息，请访问本节末尾提供的链接。

对于签名密钥，此图显示了一系列计算，计算过程中每个步骤的结果都会输入到下一步。最后一步是签名密钥。

收到经身份验证的请求后，Amazon S3 服务器使用请求中包含的身份验证信息来重新创建签名。如果签名相匹配，Amazon S3 会处理您的请求；否则将拒绝请求。

有关对请求进行身份验证的更多信息，请参阅以下主题：

- [对请求进行身份验证：使用 Authorization 标头 \(AWS 签名版本 4\)](#)
- [对请求进行身份验证：使用查询参数 \(AWS 签名版本 4\)](#)
- [使用 POST 的基于浏览器的上传 \(AWS 签名版本 4\)](#)

对请求进行身份验证：使用 Authorization 标头 (AWS 签名版本 4)

主题

- [概述](#)
- [Authorization 标头的签名计算：在单个分块中传输有效载荷 \(AWS 签名版本 4 \)](#)
- [Authorization 标头的签名计算：在多个分块中传输有效载荷 \(分块上传 \) \(AWS 签名版本 4 \)](#)
- [尾随标头的签名计算 \(分块上传 \) \(AWS 签名版本 4 \)](#)

概述

使用 HTTP Authorization 标头是提供身份验证信息的最常用方法。除了 [POST 请求](#) 和使用查询参数签名的请求外，所有 Amazon S3 操作都使用 Authorization 请求标头来提供身份验证信息。

以下是 Authorization 标头值的示例：为便于阅读，此示例中添加了换行符：

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

下表介绍了上述示例中 Authorization 标头值的各个组成部分：

组件	说明
AWS4-HMAC-SHA256	用于计算签名的算法。使用 AWS 签名版本 4 进行身份验证时必须提供此值。 该字符串指定 AWS 签名版本 4 (AWS4) 和签名算法 (HMAC-SHA256)。
Credential	您的访问密钥 ID 和范围信息，包括用于计算签名的日期、区域和服务。 该字符串具有以下形式：

组件	说明
	<your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request 其中： • <date> 值使用 YYYYMMDD 格式指定。 • 向 Amazon S3 发送请求时，<aws-service> 值为 s3。
SignedHeaders	用于计算 Signature 的请求标头的分号分隔列表。该列表仅包含标头名称，并且标头名称必须为小写。例如： host;range;x-amz-date
Signature	256 位签名以 64 个小写十六进制字符表示。例如： fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024 请注意，签名计算因所选择的传输有效负载的选项而异。

签名计算因您选择用于传输请求有效载荷的方法而异。S3 支持以下选项：

- 在单个分块中传输有效载荷：在这种情况下，您可以使用以下签名计算选项：
- 签名的有效载荷选项：您可以选择计算整个有效载荷校验和并将其包含在签名计算中。这增加了安全性，但您需要读取两次有效载荷或将其缓冲到内存中。

例如，为了上传文件，您需要先读取文件以计算签名计算的有效载荷哈希值，然后在创建请求时再次读取文件以进行传输。对于较小的有效载荷，这种方法可能更可取。但是，对于大型文件，读取文件两次可能效率低下，因此您可能需要改为分块上传数据。

- 为了提高安全性，我们建议您包含有效载荷校验和。
- 未签名的有效载荷选项：不在签名计算中包含有效载荷校验和。
- 有关计算签名和构造 Authorization 标头值的分步说明，请参阅 [Authorization 标头的签名计算：在单个分块中传输有效载荷 \(AWS 签名版本 4 \)](#)。
- 在多个分块中传输有效载荷 (分块上传)：在这种情况下，您可以分块传输有效载荷。无论有效载荷大小如何，您都可以分块传输有效载荷。

您可以将有效载荷分成多个分块。这些分块可以是固定大小，也可以是可变大小。通过分块上传数据，可以避免读取整个有效载荷来计算签名。相反，对于第一个分块，您可以计算仅使用请求标头的种子签名。第二个分块包含第一个分块的签名，随后的每个分块都包含其前一个分块的签名。在上传结束时，您将发送一个包含 0 字节数据的最后一个分块，其中包含有效载荷的最后一个分块的签名。有关更多信息，请参阅 [Authorization 标头的签名计算：在多个分块中传输有效载荷 \(分块上传 \) \(AWS 签名版本 4 \)](#)。

签署请求时，您可以使用 AWS 签名版本 4 或 AWS 签名版本 4A。两者之间的关键区别取决于签名的计算方式。对于 AWS 签名版本 4A，签名不包含区域特定的信息，并且使用 AWS4-ECDSA-P256-SHA256 算法计算得出。

除了这些选项外，您还可以选择在请求中加入尾部。为了在请求中包含尾部，您需要通过将 x-amz-content-sha256 设置为适当的值来在标头中指定该尾部。如果您使用的是尾随标头，则必须在标头中包含 x-amz-trailer，并将尾随标头名称指定为逗号分隔列表中的字符串。所有尾随标头都在最后一个分块之后写入。如果您要将数据分成多个分块进行上传，则在发送尾随标头之前，必须发送包含 0 字节数据的最后一个分块。

发送请求时，您必须通过添加带有以下值之一的 x-amz-content-sha256 标头，来告诉 Amazon S3 您在签名计算中选择了上述哪个选项：

标头值	说明
实际有效载荷校验和值	此值是对对象的实际校验和，仅当您在单个分块中上传数据时才适用。
UNSIGNED-PAYLOAD	当您将对象作为单个未签名的分块上传时，使用此值。
STREAMING-UNSIGNED-PAYLOAD-TRAILER	在通过多个分块发送未签名的有效载荷时，使用此值。在这种情况下，在上传分块后，您还有一个尾随标头。

标头值	说明
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	在通过多个分块发送有效载荷并使用 AWS4-HMAC-SHA256 对分块进行签名时，使用此值。这会生成一个 SigV4 签名。
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	在通过多个分块发送有效载荷并使用 AWS4-HMAC-SHA256 对分块进行签名时，使用此值。这会生成一个 SigV4 签名。此外，分块的摘要作为尾随标头包含在内。
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	在通过多个分块发送有效载荷并使用 AWS4-ECDSA-P256-SHA256 对分块进行签名时，使用此值。这会生成一个 SigV4A 签名。
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	在通过多个分块发送有效载荷并使用 AWS4-ECDSA-P256-SHA256 对分块进行签名时，使用此值。这会生成一个 SigV4A 签名。此外，分块的摘要作为尾随标头包含在内。

收到请求后，Amazon S3 使用 Authorization 标头和 date 标头中的信息重新创建待签字符串。然后，它使用身份验证服务来验证签名是否匹配。可以使用 HTTP Date 或 x-amz-date 标头来指定请求日期。如果两个标头都存在，则 x-amz-date 优先。

如果签名匹配，Amazon S3 会处理您的请求；否则，您的请求将失败。

有关更多信息，请参阅以下主题：

[Authorization 标头的签名计算：在单个分块中传输有效载荷 \(AWS 签名版本 4 \)](#)

[Authorization 标头的签名计算：在多个分块中传输有效载荷 \(分块上传 \) \(AWS 签名版本 4 \)](#)

[尾随标头的签名计算 \(分块上传 \) \(AWS 签名版本 4 \)](#)

Authorization 标头的签名计算：在单个分块中传输有效载荷 (AWS 签名版本 4)

使用 Authorization 标头对请求进行身份验证时，标头值包括签名等其它内容。签名计算因您就传输有效载荷所进行的选择而异 ([概述](#))。本节介绍在选择以单个分块传输有效载荷时的签名计算。示例部分 (请参阅 [示例：签名计算](#)) 显示了签名计算和生成的 Authorization 标头，您可以将其用作测试套件来验证代码。

Important

在单个分块中传输有效载荷时，您可以选择在签名计算中包含有效载荷哈希值，称为已签名的有效载荷（如果不包括该哈希值，则该有效载荷被视为未签名）。下一节中讨论的签名过程适用于这两者，但请注意以下区别：

- 签名的有效载荷选项：在构造规范请求时包含有效载荷哈希值（然后，该哈希值将成为 StringToSign 的一部分，如签名计算部分所述）。在向 S3 发送请求时，您还可以指定与 `x-amz-content-sha256` 标头值相同的值。
- 未签名的有效载荷选项：您在构造规范请求时包含文本字符串 `UNSIGNED-PAYLOAD`，并在向 Amazon S3 发送请求时设置与 `x-amz-content-sha256` 标头值相同的值。

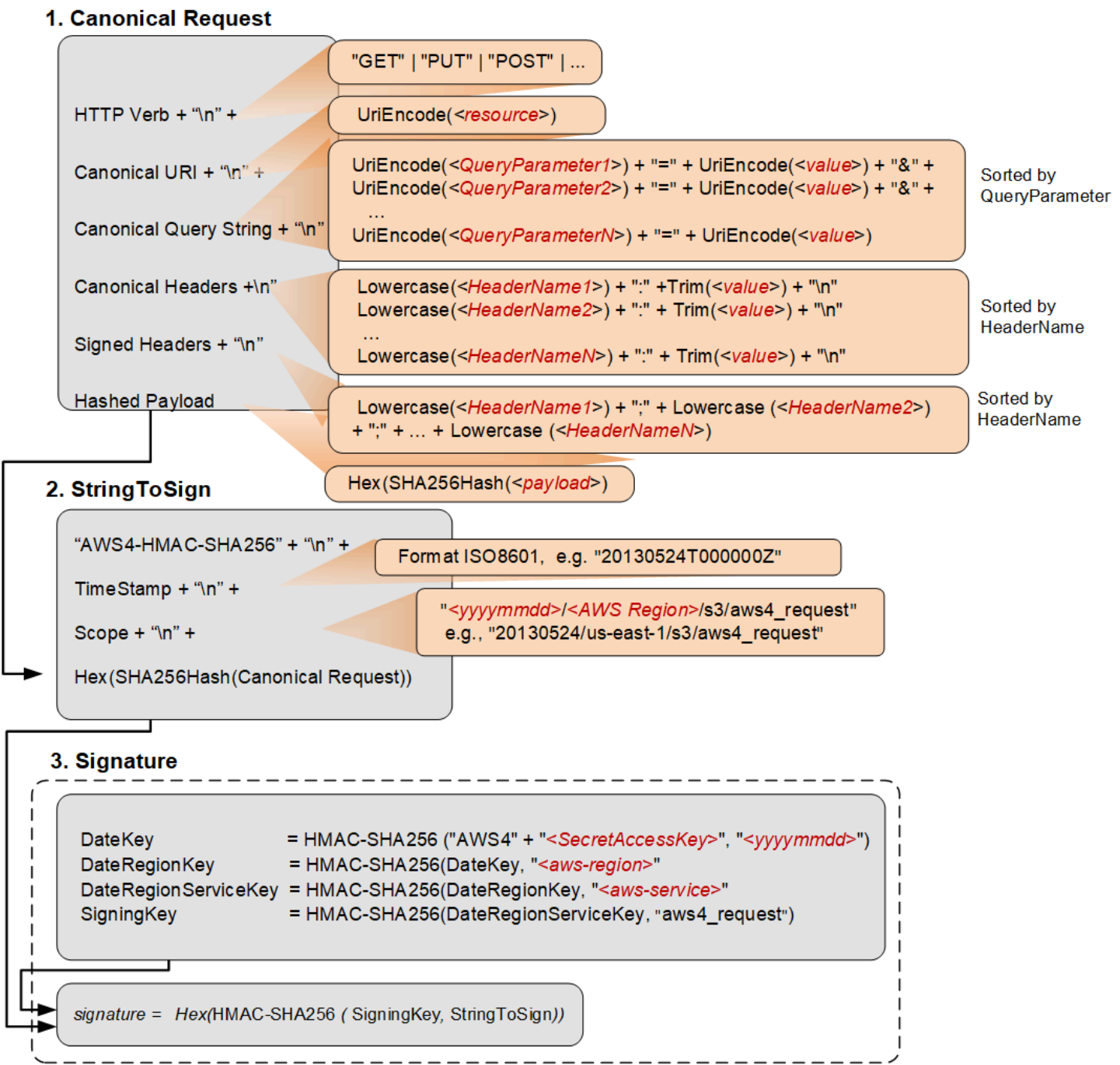
当您将请求发送到 Amazon S3 时，`x-amz-content-sha256` 标头值会通知 Amazon S3 有效载荷是否已签名。然后，Amazon S3 可以相应地创建签名以进行验证。

在这两种情况下，由于 `x-amz-content-sha256` 标头值已经是 HashedPayload 的一部分，因此您无需将 `x-amz-content-sha256` 标头作为规范标头包含在内。

计算签名

要计算签名，首先需要待签字符串。然后，使用签名密钥计算待签字符串的 HMAC-SHA256 哈希值。下图演示了该过程，包括您为签名而创建的字符串的各个组成部分

Amazon S3 收到经身份验证的请求时，它将计算签名，然后将其与您在请求中提供的签名进行比较。出于此原因，您必须采用 Amazon S3 使用的相同方法来计算签名。根据适用于签名的统一格式放置请求的过程称为标准化。



下表介绍了图中显示的函数。您需要为这些函数实现代码。

函数	说明
Lowercase()	将字符串转换为小写。
Hex()	base-16 编码的小写形式。

函数	说明
SHA256Hash()	安全哈希算法 (SHA) 加密哈希函数。
HMAC-SHA256()	使用 SHA256 算法和提供的签名密钥计算 HMAC。这是最终的签名。
Trim()	删除所有前导空格或尾随空格。

函数	说明
UriEncode()	URI 对每个字节进行编码。UriEncode() 必须强制执行以下规则： • URI 对除非预留字符之外的所有字节进行编码：“A”-“Z”、“a”-“z”、“0”-“9”、“-”、“.”、“_”和“~”。 • 空格字符是预留字符，必须编码为“%20”（而不是“+”）。 • 每个 URI 编码字节由“%”和该字节的两位十六进制值组成。 • 十六进制值中的字母必须为大写，例如“%1A”。 • 对除对象键名称之外的所有位置的正斜杠字符“/”进行编码。例如，如果对象键名称是 photos/Jan/sample.jpg，则不会对键名称中的正斜杠进行编码。
	 Important 由于底层 RFC 中的实现差异和相关歧义，您开发平台提供的标准 UrienCode 函数可能无法正常工作。建议您编写自己的自定义 UrienCode 函数，以确保编码能够正常工作。
	以下是 Java 中的 UriEncode() 函数示例。 <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else {</pre>

函数	说明
	<pre> result.append(toHexUTF8(ch)); } } return result.toString(); }</pre>

任务 1：创建规范请求

本节提供创建规范请求的概述。

以下是 Amazon S3 用来计算签名的规范请求格式。要使签名匹配，您必须使用以下格式创建规范请求：

```
<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>
```

其中：

- *HTTPMethod* 是 HTTP 方法之一，例如 GET、PUT、HEAD 和 DELETE。
- *CanonicalURI* 是 URI 的绝对路径组件的 URI 编码版本，即从域名后面的“/”开始，直至字符串结尾处的任何内容；或者如果包含查询字符串参数，则为直至问号字符（“?”）的任何内容。以下示例中的 URI /examplebucket/myphoto.jpg 是绝对路径，并且您无需在绝对路径中对“/”进行编码：

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

 Note

向 Amazon S3 发出请求时，您没有对 URI 路径进行规范化。例如，您可能有一个存储桶，其中包含一个名为“my-object//example//photo.user”的对象。对路径进行规范化会将请求中的对象名称更改为“my-object/example/photo.user”。这是该对象的不正确路径。

- **CanonicalQueryString** 指定 URI 编码的查询字符串参数。您可以单独对名称和值进行 URI 编码。您还必须按键名称的字母顺序对规范查询字符串中的参数进行排序。编码后进行排序。以下 URI 示例中的查询字符串是：prefix=somePrefix&marker=someMarker&max-keys=20

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

规范查询字符串如下所示（为便于阅读，此示例中添加了换行符：）：

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

当请求针对子资源时，相应的查询参数值将为空字符串（""）。例如，以下 URI 标识了 examplebucket 存储桶上的 ACL 子资源：

```
http://s3.amazonaws.com/examplebucket?acl
```

本例中的 CanonicalQueryString 如下：

```
UriEncode("acl") + "=" + ""
```

如果 URI 不包含“?”，则请求中没有查询字符串，并且您需要将规范查询字符串设置为空字符串（""）。此外，仍需包含“\n”。

- **CanonicalHeaders** 是请求标头及其值的列表。各个标头名称和值对用换行符（“\n”）分隔。标头名称必须为小写。您必须按字母顺序对标头名称进行排序才能构造字符串，如以下示例所示：

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

上一部分介绍了本示例中使用的 Lowercase() 和 Trim() 函数。

CanonicalHeaders 列表必须包含以下内容：

- HTTP host 标头。
- 如果请求中存在 Content-MD5 标头，则必须将其添加到 *CanonicalHeaders* 列表中。
- 此外，还必须添加计划在请求中包含的所有 x-amz-* 标头。例如，如果您使用临时安全凭证，则请求中必须包含 x-amz-security-token。您必须将此标头添加到 *CanonicalHeaders* 列表中。

 Note

- 所有 AWS 签名版本 4 请求都需要 x-amz-content-sha256 标头。它将提供请求负载的哈希。但是，您无需将 x-amz-content-sha256 作为规范标头包含在内，因为 S3 在计算请求中发送的有效载荷哈希值时会自动使用其值。
- 如果不包含有效负载，则必须提供空字符串的哈希值。
- 如果您不希望 S3 检查请求的哈希值，则可以改用文本字符串 "UNSIGNED-PAYLOAD"。

以下是示例 CanonicalHeaders 字符串。标头名称为小写且已排序。

Example 示例 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Example 示例 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Example 示例 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

 Note

为了计算授权签名，需要主机标头 and 所有 `x-amz-*` 标头（不包括 `x-amz-content-sha256`）。但是，为了防止篡改数据，您应该考虑在签名计算中包含所有标头。

对 `x-amz-content-sha256` 标头进行签名是可选的，因为 S3 将在计算收到的请求有效载荷哈希值时使用其值。

- *SignedHeaders* 是按字母顺序排序、以分号分隔的小写请求标头名称的列表。列表中的请求标头与您在 CanonicalHeaders 字符串中包含的标头相同。在前面的示例中，*SignedHeaders* 的值如下：

Example 示例 1 和 2

```
host;x-amz-content-sha256;x-amz-date
```

Example 示例 3

```
host;x-amz-date
```

- *HashedPayload* 是请求有效载荷的 SHA256 哈希值的十六进制值。

```
Hex(SHA256Hash(<payload>))
```

如果请求中不包含有效负载，则按如下方式计算空字符串的哈希值：

```
Hex(SHA256Hash(""))
```

哈希值将返回以下值：

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

例如，当您使用 PUT 请求上传对象时，您需要在正文中提供对象数据。当您使用 GET 请求检索对象时，您将计算空字符串哈希值。

任务 2：创建待签字符串

本节提供创建待签字符串的概述。有关分步说明，请参阅《AWS 一般参考》中的[任务 2：创建待签字符串](#)。

待签字符串是以下字符串的结合：

```
"AWS4-HMAC-SHA256" + "\n" +  
timeStampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

常量字符串 AWS4-HMAC-SHA256 指定您正在使用的哈希算法 HMAC-SHA256。timeStamp 是采用 ISO 8601 格式的当前 UTC 时间（例如 20130524T000000Z）。

Scope 将生成的签名绑定到特定的日期、AWS 区域和服务。这样，生成的签名将仅在特定区域和特定服务中有效。签名在指定日期后的七天内有效。

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

对于 Amazon S3，服务字符串为 s3。有关 *region* 字符串的列表，请参阅《AWS 一般参考》中的[区域和端点](#)。此表中的“区域”列提供了有效区域字符串的列表。

以下范围将生成的签名限制为 us-east-1 区域和 Amazon S3。

```
20130606/us-east-1/s3/aws4_request
```

Note

Scope 必须使用与用于计算签名密钥的相同日期，如下一节所述。

任务 3：计算签名

在 AWS 签名版本 4 中，您首先创建一个范围仅限于特定区域和服务的签名密钥，而不是使用 AWS 访问密钥对请求进行签名。有关签名密钥的更多信息，请参阅[对请求进行签名简介](#)。

```
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**

一些使用案例可以处理签名密钥长达 7 天。有关更多信息，请参阅[与他人共享对象](#)。

有关区域字符串的列表，请参阅《AWS 一般参考》中的[区域和端点](#)。

使用签名密钥，您可以将 AWS 凭证保存在一个安全的地方。例如，如果您有多个服务器与 Amazon S3 通信，则可以与这些服务器共享签名密钥；您不必在每个服务器上保留秘密访问密钥的副本。签名密钥的有效期最长为七天。因此，每次计算签名密钥时，都需要与服务器共享签名密钥。有关更多信息，请参阅[对请求进行身份验证 \(AWS 签名版本 4\)](#)。

最终签名是待签字符串的 HMAC-SHA256 哈希值，使用签名密钥作为密钥。

```
HMAC-SHA256(SigningKey, StringToSign)
```

有关创建签名的分步说明，请参阅《AWS 一般参考》中的[任务 3：创建签名](#)。

示例：签名计算

您可以将本节中的示例用作参考，以检查代码中的签名计算。示例中显示的计算使用以下数据：

- 访问密钥示例。

参数	值
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- 请求时间戳：20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT)。

- 存储桶名称 examplebucket。
- 假设存储桶位于美国东部（弗吉尼亚州北部）区域。凭证 Scope 和 Signing Key 计算使用 us-east-1 作为区域说明符。有关其它区域的信息，请参阅《AWS 一般参考》中的[区域和端点](#)。
- 您可以使用路径类型请求或虚拟托管类型请求。以下示例演示如何对虚拟托管类型请求进行签名，例如：

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[存储桶的虚拟托管](#)。

示例：GET 对象

以下示例从 examplebucket 获取对象（test.txt）的前 10 个字节。有关 API 操作的更多信息，请参阅[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
```

由于此 GET 请求不提供任何正文内容，因此 x-amz-content-sha256 值可以是空请求正文的哈希值，也可以是文本字符串 "UNSIGNED-PAYLOAD"。以下步骤显示使用空字符串的哈希值进行签名计算以及构造 Authorization 标头的过程。

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
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

在规范请求字符串中，最后一行是空请求正文的哈希值。因为请求中没有查询参数，所以第三行是空的。

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
7344ae5b7ee6c3e7e6b0fe0640412a37625d1fbfff95c48bbb2dc43964946972
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

4. Authorization 标头

生成的 Authorization 标头如下：

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

示例：PUT 对象

此示例 PUT 请求在 `examplebucket` 中创建一个对象 (`test$file.text`)。示例假定以下各项：

签名计算：在单个分块中传输有效载荷

API 版本 2006-08-01 1471

- 您正在通过添加 `x-amz-storage-class` 请求标头来请求 `REDUCED_REDUNDANCY` 作为存储类别。有关存储类别的信息，请参阅《Amazon Simple Storage Service 用户指南》中的[存储类别](#)。
- 已上传文件的内容是一个字符串 "Welcome to Amazon S3."，请求中 `x-amz-content-sha256` 的值基于这个字符串。

有关 API 操作的信息，请参阅 [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>

以下步骤显示了签名计算。

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
```

在规范请求中，第三行是空的，因为请求中没有查询参数。可以选择对 `x-amz-content-sha256` 规范标头进行签名，因为其有效载荷哈希值已在请求的底部提供。最后一行是正文

的哈希值，它应该与在 HTTP 请求中发送给 S3 的 `x-amz-content-sha256` header 值相同。

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9e0e90d9c76de8fa5b200d8c849cd5b8dc7a3be3951ddb7f6a76b4158342019d
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

4. Authorization 标头

生成的 Authorization 标头如下：

```
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
```

示例：GET 存储桶生命周期

以下 GET 请求检索 examplebucket 的生命周期配置。有关 API 操作的信息，请参阅 [GetBucketLifecycleConfiguration](#)。

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
```

```
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

由于请求未提供任何正文内容，因此 `x-amz-content-sha256` 标头值是空请求正文的哈希值。以下步骤显示了签名计算。

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
```

在规范请求中，最后一行是空请求正文的哈希值。

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9766c798316ff2757b517bc739a67f6213b4ab36dd5da2f94eaebf79c77395ca
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

4. Authorization 标头

生成的 Authorization 标头如下：

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

示例：Get 存储桶（列出对象）

以下示例从 examplebucket 存储桶中检索对象的列表。有关 API 操作的信息，请参阅 [ListObjects](#)。

```
GET ?max-keys=2&prefix=J HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

由于请求未提供正文，因此 x-amz-content-sha256 的值是空请求正文的哈希值。以下步骤显示了签名计算。

1. StringToSign

a. CanonicalRequest

```
GET
/
max-keys=2&prefix=J
host:examplebucket.s3.amazonaws.com
x-amz-content-
sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

在规范请求中，最后一行是空请求正文的哈希值。

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
df57d21db20da04d7fa30298dd4488ba3a2b47ca3a489c74750e0f1e7df1b9b7
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. Authorization 标头

生成的 Authorization 标头如下：

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

Authorization 标头的签名计算：在多个分块中传输有效载荷（分块上传） （AWS 签名版本 4）

如[概述](#)中所述，当使用 Authorization 标头对请求进行身份验证时，您可以选择分块上传有效载荷。您可以按固定大小或可变大小的分块发送数据。本节介绍分块上传中的签名计算过程、如何创建分块正文，以及延迟签名的工作原理，即您首先上传分块，然后在后续分块中发送其签名。示例部分（请参阅[示例：PUT 对象](#)）显示了签名计算和生成的 Authorization 标头，您可以将其用作测试套件来验证代码。

Note

在一系列分块中传输数据时，必须执行以下操作之一：

- 使用 Content-Length HTTP 标头显式指定内容的总长度（以字节为单位的对象长度加上每个分块中的元数据）。为此，在开始请求之前，您必须预先计算有效载荷的总长度，包括您在每个分块中发送的元数据。
- 指定 Transfer-Encoding HTTP 标头。如果您包含 Transfer-Encoding 标头并指定除 identity 之外的任何值，则必须忽略 Content-Length 标头。

对于所有请求，您都必须包含 x-amz-decoded-content-length 标头，同时指定对象的大小（以字节为单位）。

每个分块签名计算都包括前一个分块的签名。首先，您仅使用标头来创建种子签名。在第一个分块的签名计算中使用种子签名。对于每个后续分块，您都要创建一个分块签名，其中包含前一个分块的签名。这样，分块签名链接在一起；也就是说，分块 n 的签名是函数 $F(\text{chunk } n, \text{signature}(\text{chunk } n-1))$ 。链接可确保您按正确的顺序发送分块。

要执行分块上传，请执行以下操作：

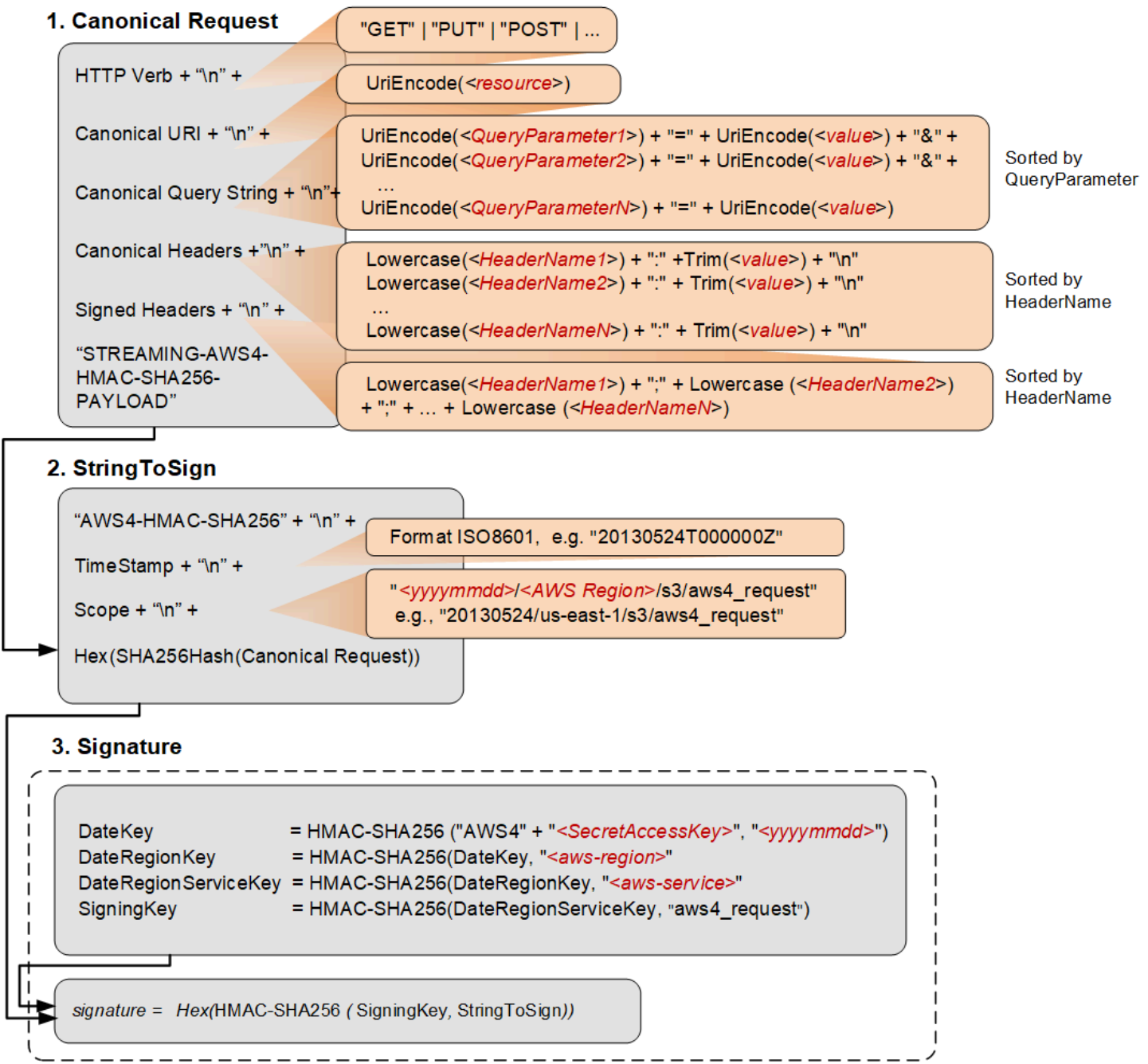
1. 确定有效载荷分块大小。编写代码时需要此信息。

分块大小必须至少为 8 KB。为了获得更好的性能，我们建议分块大小至少为 64 KB。此分块大小适用于除最后一个分块之外的所有分块。您发送的最后一个分块可以小于 8 KB。如果有效载荷较小而适合放入一个分块中，则它可以小于 8 KB。

2. 创建要包含在第一个分块中的种子签名。有关更多信息，请参阅 [计算种子签名](#)。
3. 创建第一个分块并对其进行流式传输。有关更多信息，请参阅 [定义分块正文](#)。
4. 对于每个后续分块，计算在您签名的字符串中包含先前签名的分块签名，构造该分块并发送它。有关更多信息，请参阅 [定义分块正文](#)。
5. 发送最后一个额外的分块，它与构造中的其它分块相同，但其数据字节为零。有关更多信息，请参阅 [定义分块正文](#)。

计算种子签名

下图说明了计算种子签名的过程。



下表介绍了图中显示的函数。您需要为这些函数实现代码。

函数	说明
Lowercase()	将字符串转换为小写。
Hex()	base-16 编码的小写形式。

函数	说明
SHA256Hash()	安全哈希算法 (SHA) 加密哈希函数。
HMAC-SHA256()	使用 SHA256 算法和提供的签名密钥计算 HMAC。这是最终的签名。
Trim()	删除所有前导空格或尾随空格。

函数	说明
UriEncode()	URI 对每个字节进行编码。UriEncode() 必须强制执行以下规则： • URI 对除非预留字符之外的所有字节进行编码：“A”-“Z”、“a”-“z”、“0”-“9”、“-”、“.”、“_”和“~”。 • 空格字符是预留字符，必须编码为“%20”（而不是“+”）。 • 每个 URI 编码字节由“%”和该字节的两位十六进制值组成。 • 十六进制值中的字母必须为大写，例如“%1A”。 • 对除对象键名称之外的所有位置的正斜杠字符“/”进行编码。例如，如果对象键名称是 photos/Jan/sample.jpg，则不会对键名称中的正斜杠进行编码。
	 Important 由于底层 RFC 中的实现差异和相关歧义，您开发平台提供的标准 UriEncode 函数可能无法正常工作。建议您编写自己的自定义 UriEncode 函数，以确保编码能够正常工作。
	以下是 Java 中的 UriEncode() 函数示例。 <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else {</pre>

函数	说明
	<pre> result.append(toHexUTF8(ch)); } } return result.toString(); }</pre>

有关签名过程的信息，请参阅 [Authorization 标头的签名计算：在单个分块中传输有效载荷 \(AWS 签名版本 4 \)](#)。过程是相同的，只是创建 CanonicalRequest 的过程有不同之处，如下所示：

- 除了您计划添加的请求标头外，还必须包括以下标头：

标题	说明
x-amz-content-sha256	所有 AWS 签名版本 4 请求都需要此标头。将值设置为 STREAMING-AWS4-HMAC-SHA256-PAYLOAD ，以表示签名仅涵盖标头且没有有效载荷。
Content-Encoding	将值设置为 aws-chunked 。 Amazon S3 支持多个内容编码值。当使用签名版本 4 流式传输 API 时，您可以指定自定义内容编码。 例如： Content-Encoding : aws-chunked,gzip Amazon S3 存储生成的对象，content-encoding 标头中不带 aws-chunked 值。如果 aws-chunked 是您在 content-encoding 标头中传递的唯一值，则 S3 会认为 content-encoding 标头为空，在您检索对象时不会返回此标头。

标题	说明
x-amz-decoded-content-length	将值设置为要分块的数据的长度（以字节为单位），而不计入任何元数据。例如，如果您要上传一个 4 GB 的文件，请将该值设置为 4294967296。这是要上传的对象的原始大小（您要存储在 Amazon S3 中的数据）。
Content-Length	将该值设置为所传输的 HTTP 正文的实际大小，其中包括数据的长度（为 x-amz-decoded-content-length 设置的值）以及分块元数据。每个分块都有元数据，例如前一个分块的签名。下一节将讨论分块计算。如果您包含 Transfer-Encoding 标头并指定除 identity 之外的任何值，则不得包含 Content-Length 标头。

您发送带有种子签名的第一个分块。您必须按下一节所述构造该分块。

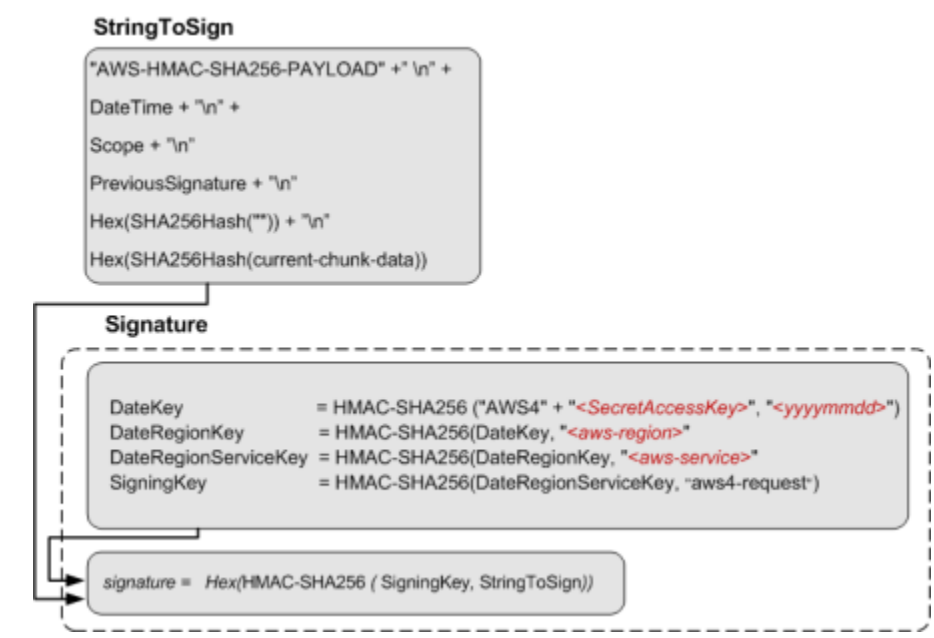
定义分块正文

所有分块都包含一些元数据。每个分块都必须符合以下结构：

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

其中：

- IntHexBase() 是您编写的用于将整数分块大小转换为十六进制的函数。例如，如果分块大小为 65536，则十六进制字符串为“10000”。
- chunk-size* 是分块数据的大小（以字节为单位），不含元数据。例如，如果您要上传一个 65 KB 的对象，并且使用的分块大小为 64 KB，则将数据分成三个分块上传：第一个为 64 KB，第二个为 1 KB，最后一个分块为 0 字节。
- signature* 对于每个分块，您可以使用以下待签字符串计算签名。对于第一个分块，使用种子签名作为前一个签名。



您发送的最后一个分块数据的大小为 0，尽管分块正文仍包含元数据（包括前一个分块的签名）。

示例：PUT 对象

您可以将本节中的示例用作参考，以检查代码中的签名计算。在查看示例之前，请注意以下事项：

- 这些示例中的签名计算使用以下示例安全凭证。

参数	值
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- 所有示例都使用请求时间戳 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT)。
- 所有示例都使用 examplebucket 作为存储桶名称。
- 假设存储桶位于美国东部（弗吉尼亚州北部）区域，凭证 Scope 和 Signing Key 计算使用 us-east-1 作为区域说明符。有关更多信息，请参阅 [Amazon Web Services 一般参考](#) 中的区域和终端节点。
- 您可以使用路径类型请求或虚拟托管类型请求。以下示例使用虚拟托管类型请求，例如：

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[存储桶的虚拟托管](#)。

以下示例发送 PUT 请求以上传对象。签名计算假设如下：

- 您正在上传一个 65 KB 的文本文件，文件内容是由字母“a”组成的单字符的字符串。
- 分块大小为 64 KB。因此，有效载荷分三个分块上传，分别为 64 KB、1 KB，以及包含 0 字节分块数据的最后一个分块。
- 生成的对象具有键名称 chunkObject.txt。
- 您正在通过添加 x-amz-storage-class 请求标头来请求 REDUCED_REDUNDANCY 作为存储类别。

有关 API 操作的信息，请参阅 [PutObject](#)。常用请求语法如下：

```
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>
```

以下步骤显示了签名计算。

1. 种子签名：创建待签字符串

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
```

在规范请求中，第三行是空的，因为请求中没有查询参数。最后一行是作为经过哈希处理的有效载荷值提供的常量字符串，应与 x-amz-content-sha256 header 的值相同。

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

有关待签字符串中每行的信息，请参阅解释种子签名计算的图表。

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 种子签名

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```


4. Authorization 标头

生成的 Authorization 标头如下：

```
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. 分块 1：(65536 字节，字母“a”的值为 97)

a. 分块待签字符串：

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

有关待签字符串中每行的信息，请参阅前面的图表，其中显示了待签字符串的各个组成部分（例如，最后三行是 previous-signature、hash("") 和 hash(current-chunk-data)）。

b. 分块签名：

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. 发送的分块数据：

```
10000;chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

6. 分块 2：(1024 字节，字母“a”的值为 97)

a. 分块待签字符串：

签名计算：在多个分块中传输有效载荷

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. 分块签名：

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. 发送的分块数据：

```
400; chunk-
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
<1024 bytes>
```

7. 分块 3：(0 字节数据)

a. 分块待签字符串：

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. 分块签名：

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. 发送的分块数据：

```
0; chunk-
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

尾随标头的签名计算（分块上传）（AWS 签名版本 4）

使用 Authorization 标头对请求进行身份验证时，您也可以分块上传有效载荷。当您以分块发送对象的数据时，您也可以选择包括尾随标头。（有关更多信息，请参阅 [Authorization 标头的签名计算：在多个分块中传输有效载荷（分块上传）（AWS 签名版本 4）](#)。）本节介绍当您想在多分块上传的末尾包含尾随标头时需要采取的步骤。

 Important

当包含尾随标头时，必须在初始标头中发送以下内容：

- 您必须将 x-amz-content-sha256 设置为适当的值，以表示将包含尾部。要查看 x-amz-content-sha256 的可接受值，请参阅[对请求进行身份验证：使用 Authorization 标头（AWS 签名版本 4）](#)。
- 您必须设置 x-amz-trailer，才能在尾随标头中指明要包含的内容。

只有在上传分块后才会发送尾随标头。之前的分块按常规发送，并按照前几节所述进行签名，包括发送有效载荷为 0 字节的最后一个分块。尾随标头作为其自己的分块包含在内，并在有效载荷为 0 字节的最后一个分块之后发送。例如，如果您的数据以 100 KB 的分块结尾，则应发送以下内容：

- 之前的数据分块
- 对象的 100 KB 的最后一个分块
- 0 字节的分块，表示对象的结尾
- 尾随标头分块

示例：检查签名计算

您可以将本节中的示例用作参考，以检查代码中的签名计算。在查看示例之前，请注意以下事项：

- 这些示例中的签名计算使用以下示例安全凭证。

参数	值
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSCredentialsExpiration	2014-01-01T12:00:00Z
AWSSessionToken	AKIAIOSFODNN7EXAMPLE
AWSSessionTokenExpiration	2014-01-01T12:00:00Z
AWSSessionTokenSecretKey	AKIAIOSFODNN7EXAMPLE

参数	值
AWSecretAccessKey	wJa1rXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- 所有示例都使用请求时间戳 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT)。
- 所有示例都使用 examplebucket 作为存储桶名称。
- 假设存储桶位于美国东部 (弗吉尼亚州北部) 区域，凭证 Scope 和 Signing Key 计算使用 us-east-1 作为区域说明符。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 [区域和终端节点](#)。
- 您可以使用路径类型请求或虚拟托管类型请求。以下示例使用虚拟托管类型请求，例如：

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [存储桶的虚拟托管](#)。

以下示例发送 PUT 请求以上传对象。签名计算假设如下：

- 您正在上传一个 65 KB 的文本文件，文件内容是由字母“a”组成的单字符的字符串。
- 分块大小为 64 KB。因此，有效载荷分三个分块上传，分别为 64 KB、1 KB，以及包含 0 字节分块数据的最后一个分块。
- 生成的对象具有键名称 chunkObject.txt。
- 您正在通过添加 x-amz-storage-class 请求标头来请求 REDUCED_REDUNDANCY 作为存储类别。
- 传输包含 CRC32C 校验和值作为尾随标头。

有关 API 操作的信息，请参阅 [PutObject](#)。常用请求语法如下：

```
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>
```

以下步骤显示了签名计算。

1. 种子签名：创建待签字符串

a. CanonicalRequest

```
PUT
/examplebucket/chunk0bject.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
```

在规范请求中，第三行是空的，因为请求中没有查询参数。最后一行是作为经过哈希处理的有效载荷值提供的常量字符串，应与 x-amz-content-sha256 header 的值相同。

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

 Note

有关待签字符串中每行的信息，请参阅解释种子签名计算的图表。

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +  
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 种子签名

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Authorization 标头

生成的 Authorization 标头如下：

```
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. 分块 1：(65536 字节，字母“a”的值为 97)

a. 分块待签字符串：

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

 Note

有关待签字符串中每行的信息，请参阅前一主题 ([计算种子签名](#)) 中的图表，其中显示了待签字符串的各个组成部分。例如，最后三行包括以下内容：

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. 分块签名：

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. 发送的分块数据：

```
10000;chunk-  
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2  
<65536-bytes>
```

6. 分块 2：(1024 字节，字母“a”的值为 97)

a. 分块待签字符串：

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. 分块签名：

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. 发送的分块数据：

```
400;chunk-  
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7  
<1024-bytes>
```

7. 分块 3：(0 字节数据)

a. 分块待签字符串：

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. 分块签名：

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. 发送的分块数据：

```
0;chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. 分块 4：尾随标头

a. 尾部分块待签字符串：

```
AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774
```

Note

最后两行是 previous-signature (0 字节的数据分块签名)、hash (*trailing-checksum-header-name:base64-encoded-trailing-checksum-value*)。

计算哈希值的方式如下，不带空格：

- 尾随校验和标头名称
- 冒号 (:)
- Base64 编码的尾随校验和值

- 换行符 (\n)。

在此示例中，我们使用 crc32c 哈希算法，base64 编码的校验和值为 s008/Q==，可以按如下方式表示计算：hash('x-amz-checksum-crc32c:s008/Q==\n')。

- b. 分块签名：

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

- c. 发送的分块数据：

```
x-amz-checksum-crc32c:s008/Q==  
x-amz-trailer-  
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

对请求进行身份验证：使用查询参数（AWS 签名版本 4）

如身份验证概述（请参阅[身份验证方法](#)）中所述，您可以使用查询字符串参数提供身份验证信息。当您想要在 URL 中完整表达请求时，使用查询参数对请求进行身份验证很有用。此方法也称为对 URL 进行预签名。

预签名 URL 的一个使用案例场景是，您可以授予对 Amazon S3 资源的临时访问权限。例如，您可以在网站上嵌入预签名 URL，或者在命令行客户端（例如 Curl）中使用它来下载对象。

Note

也可以使用 AWS CLI 来创建预签名 URL。有关更多信息，请参阅《AWS CLI Command Reference》中的 [presign](#)。

以下是示例预签名 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>
```

在示例 URL 中，注意以下各项：

- 为了便于阅读，添加了换行符。
- URL 中的 X-Amz-Credential 值显示“/”字符只是为了方便阅读。实际上，它应该编码为 %2F。例如：

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

下表介绍了 URL 中提供身份验证信息的查询参数。

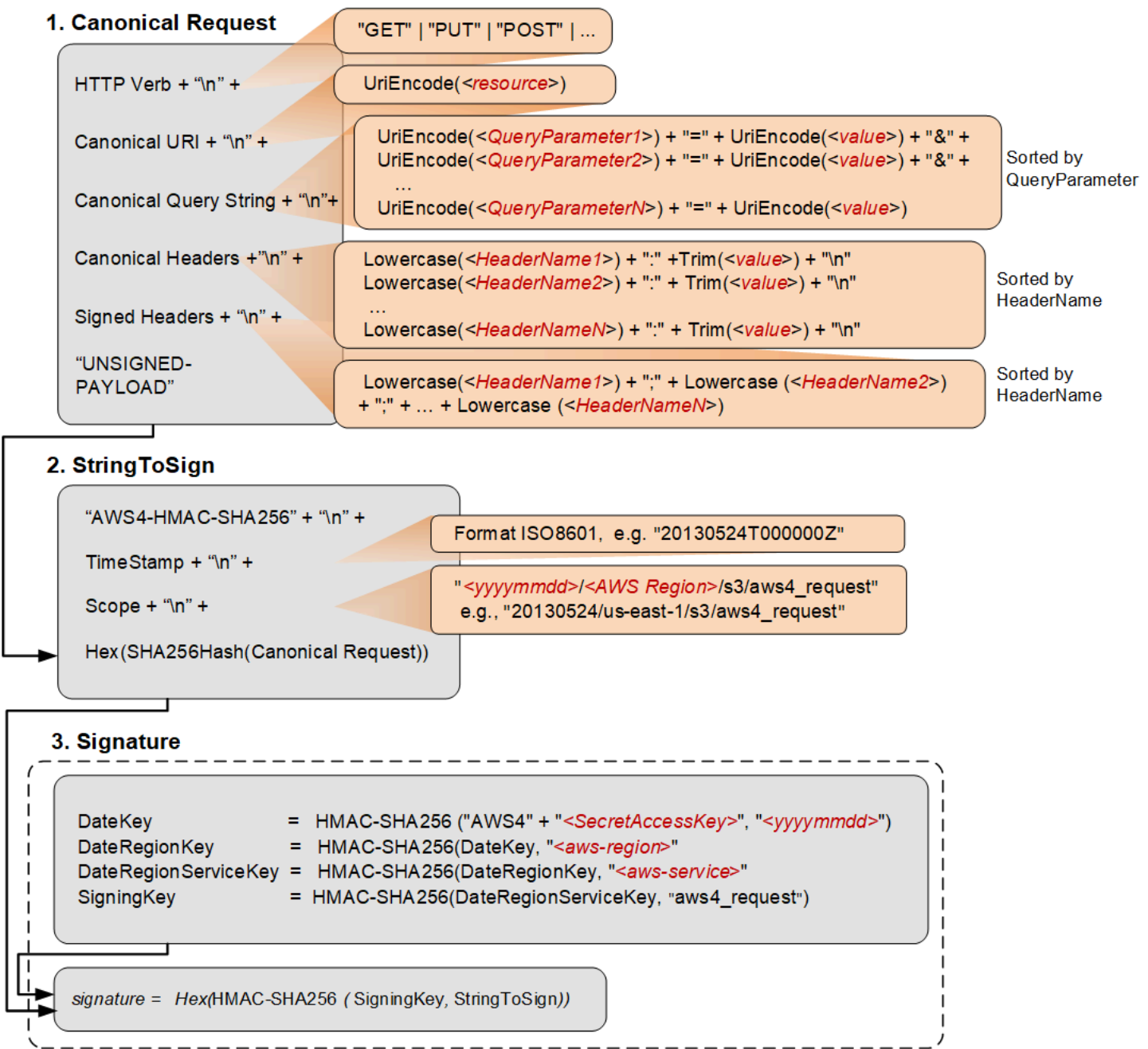
查询字符串参数名称	示例值
X-Amz-Algorithm	标识 AWS 签名版本以及用于计算签名的算法。 对于 AWS 签名版本 4，您可以将此参数值设置为 AWS4-HMAC-SHA256。该字符串标识 AWS 签名版本 4 (AWS4) 和 HMAC-SHA256 算法 (HMAC-SHA256) 。
X-Amz-Credential	除了访问密钥 ID 外，该参数还提供签名有效的范围 (AWS 区域和服务)。该值必须与您在签名计算中使用的范围匹配 (后续部分将对此进行说明)。该参数值的一般形式如下： <your-access-key-id> /<date>/<AWS ##>/<AWS-service> /aws4_request 例如： AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request

查询字符串参数名称	示例值
	对于 Amazon S3， <i>AWS-service</i> 字符串为 s3。有关 S3 <i>AWS-region</i> 字符串的列表，请参阅《AWS 一般参考》中的 区域和端点 。
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须按照“ <i>yyyyMMddTHHmmsZ</i> ”格式进行格式化。例如，如果日期和时间是“08/01/2016 15:32:41.982-700”，则必须先将其转换为 UTC（协调世界时），然后以“20160801T223241Z”形式提交。
X-Amz-Expires	提供生成的预签名 URL 的有效时间段（以秒为单位）。例如，86400（24 小时）。该值是一个整数。您可以设置的最小值为 1，最大值为 604800（七天）。 预签名 URL 的有效期最长为七天，因为您在签名计算中使用的签名密钥的有效期最长为七天。
X-Amz-SignedHeaders	列出用于计算签名的标头。签名计算中需要以下标头： • HTTP host 标头。 • 您计划添加到请求的任何 x-amz-* 标头。  Note 为了提高安全性，您应该签署计划在请求中包含的所有请求标头。
X-Amz-Signature	提供签名以验证您的请求。该签名必须与 Amazon S3 计算出的签名相匹配；否则 Amazon S3 会拒绝该请求。例如， 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 以下部分将介绍签名计算：

查询字符串参数名称	示例值
X-Amz-Security-Token	如果使用来自 STS 服务的凭证，则为可选凭证参数。

计算签名

下图说明了签名计算过程。



下表介绍了图中显示的函数。您需要为这些函数实现代码。

函数	说明
Lowercase()	将字符串转换为小写。
Hex()	base-16 编码的小写形式。
SHA256Hash()	安全哈希算法 (SHA) 加密哈希函数。
HMAC-SHA256()	使用 SHA256 算法和提供的签名密钥计算 HMAC。这是最终的签名。
Trim()	删除所有前导空格或尾随空格。
UriEncode()	URI 对每个字节进行编码。UriEncode() 必须强制执行以下规则： • URI 对除非预留字符之外的所有字节进行编码：“A”-“Z”、“a”-“z”、“0”-“9”、“-”、“.”、“_”和“~”。 • 空格字符是预留字符，必须编码为“%20”（而不是“+”）。 • 每个 URI 编码字节由“%”和该字节的两位十六进制值组成。 • 十六进制值中的字母必须为大写，例如“%1A”。 • 对除对象键名称之外的所有位置的正斜杠字符“/”进行编码。例如，如果对象键名称是 photos/Jan/sample.jpg ，则不会对键名称中的正斜杠进行编码。  Important 由于底层 RFC 中的实现差异和相关歧义，您开发平台提供的标准 UrienCode 函数可能无法正常工作。建议您编写自己的自定义 UrienCode 函数，以确保编码能够正常工作。 以下是 Java 中的 UriEncode() 函数示例。

函数	说明
	<pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBuil lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); }</pre>

有关签名过程 (创建规范请求、待签字符串和签名计算的详细信息) 的更多信息，请参阅 [Authorization 标头的签名计算：在单个分块中传输有效载荷 \(AWS 签名版本 4 \)](#)。除了在预签名 URL 中创建 CanonicalRequest 之外，过程大致相同，如下所示：

- 您未在规范请求中包含有效载荷哈希值，原因是当您创建预签名 URL 时，您并不知道有效载荷内容，因为该 URL 用于上传任意有效载荷。而应使用常量字符串 UNSIGNED-PAYLOAD。
- 规范查询字符串必须包含上表中除 X-Amz-Signature 之外的所有查询参数。
- 对于 S3，如果使用源自 STS 服务的凭证，您必须在 URL 中包含 X-Amz-Security-Token 查询参数。
- 规范标头必须包含 HTTP host 标头。如果您计划包含任何 x-amz-* 标头，则还必须添加这些标头以进行签名计算。您可以选择添加您计划在请求中包含的所有其它标头。为了提高安全性，您应该对尽可能多的标头进行签名。如果您添加的已签名标头也是已签名的查询参数，而它们的值不同，则由于输入存在冲突，您将收到 InvalidRequest 错误消息。

一个示例

假设您的 `examplebucket` 存储桶中有一个对象 `test.txt`。您希望通过创建预签名 URL 与他人共享此对象达 24 小时 (86400 秒)。

```
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>
```

以下步骤首先说明签名计算，然后说明预签名 URL 的构造。该示例作出以下附加假设：

- 请求时间戳为 `Fri, 24 May 2013 00:00:00 GMT`。
- 存储桶位于美国东部 (弗吉尼亚州北部) 区域，凭证 Scope 和 Signing Key 计算使用 `us-east-1` 作为区域说明符。有关更多信息，请参阅 https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region 中的 AWS 一般参考区域和终端节点。

您可以将此示例用作测试用例来验证您的代码计算的签名；但是，您必须使用相同的存储桶名称、对象键、时间戳和以下示例凭证：

参数	值
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

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
3bfa292879f6447bbcd7001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

现在，您已经掌握了构造预签名 URL 的所有信息。此示例生成的 URL 如下所示（您可以使用它来比较您的预签名 URL）：

```
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
```

示例 2

以下是一个显示带有 X-Amz-Security-Token 参数的预签名 URL 的示例（与前一个示例无关）。

使用 Java 的签名计算示例 (AWS 签名版本 4)

显示签名计算的 Java 示例可以从 <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip> 下载。请注意，这些示例使用适用于 Java 的 AWS SDK v1，我们建议对新的应用程序使用[适用于 Java 的 AWS SDK v2](#)。在 RunAllSamples.java 中，main() 函数执行示例请求，以创建对象、检索对象以及为该对象创建预签名 URL。该示例根据代码中提供的文本字符串创建一个对象：

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
awsSecretKey);
```

在基于 Linux 的计算机上测试示例

以下说明适用于 Linux 操作系统。

1. 在终端中，导航到包含 AWSS3SigV4JavaSamples.zip 的目录。
2. 将 .zip 文件解压缩。
3. 在文本编辑器中，打开 ./com/amazonaws/services/s3/samples/RunAllSamples.java 文件。使用以下信息更新代码：
 - 可以在其中创建新对象的存储桶的名称。

Note

这些示例使用虚拟托管类型的请求来访问存储桶。为避免潜在的错误，请确保存储桶名称符合在《Amazon Simple Storage Service 用户指南》的[存储桶限制和局限性](#)中所述的存储桶命名规则。

- 存储桶所在的 AWS 区域。

如果存储桶位于美国东部（弗吉尼亚州北部）区域，请使用 us-east-1 指定区域。有关其它 AWS 区域的列表，请访问《AWS 一般参考》中的 [Amazon Simple Storage Service \(S3 \)](#)。

4. 编译源代码并将编译后的类存储到 bin/ 目录中。

```
javac -d bin -source 6 -verbose com
```

5. 将目录更改为 bin/，然后运行 RunAllSamples。

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

该代码运行 main() 中的所有方法。对于每个请求，输出都将显示规范请求、待签字符串和签名。

使用 C# 的签名计算示例 (AWS 签名版本 4)

显示签名计算的 C# 示例可从 https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip 下载。请注意，这些示例使用适用于 .NET 的 AWS SDK v1，我们建议对新的应用程序使用[适用于 .NET 的 AWS SDK v3](#)。在 Program.cs 中，main() 函数执行示例请求，以创建对象、检索对象以及为该对象创建预签名 URL。用于签名计算的代码位于 \Signers 文件夹中。

```
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");
```

使用 Microsoft Visual Studio 2010 或更高版本测试示例

1. 将 .zip 文件解压缩。
2. 启动 Visual Studio，然后打开 .sln 文件。
3. 使用有效的安全凭证更新 App.config 文件。
4. 按如下所示更新代码：
 - 在 Program.cs 中，提供存储桶名称和存储桶所在的 AWS 区域。此示例在此存储桶中创建对象。

5. 运行该代码。
6. 要验证对象是否已创建，请复制程序创建的预签名 URL，然后将其粘贴到浏览器窗口中。

对请求进行身份验证：使用 POST 的基于浏览器的上传 (AWS 签名版本 4)

Amazon S3 支持 HTTP POST 请求，以便用户可以直接将内容上传到 Amazon S3。使用 HTTP POST 上传内容可简化上传并减少上传延迟，适用于用户将数据上传至 Amazon S3 中进行存储的情况。本节说明如何对 HTTP POST 请求进行身份验证。有关 HTTP POST 请求、如何创建表单、创建 POST 策略以及示例的更多信息，请参阅[使用 POST 的基于浏览器的上传 \(AWS 签名版本 4 \)](#)。

要对 HTTP POST 请求进行身份验证，请执行以下操作：

1. 表单必须包含以下字段，以提供签名和相关信息，供 Amazon S3 用于在收到请求时重新计算签名：

元素名称	说明
policy	以 Base64 编码的安全策略，用于描述请求中允许的内容。对于签名计算，此策略是您签名的字符串。Amazon S3 必须获取此策略，这样它才能重新计算签名。
x-amz-algorithm	使用的签名算法。对于 AWS 签名版本 4，该值为 AWS4-HMAC-SHA256 。
x-amz-credential	除了访问密钥 ID 外，此元素还提供用于计算签名密钥的范围信息，以便进行签名计算。 它是采用以下形式的字符串： <your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request 例如：

元素名称	说明
	AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . . 对于 Amazon S3，aws-service 字符串为 s3。有关 Amazon S3 aws-region 字符串的列表，请参阅《AWS 一般参考》中的区域和端点。
x-amz-date	它是采用 ISO8601 格式的日期值。例如 20130728T000000Z 。 它与您创建签名密钥时使用的日期相同。该值也必须与您在签名的策略 (x-amz-date) 中提供的值相同。
x-amz-signature	(AWS 签名版本 4) 安全策略的 HMAC-SHA256 哈希值。 有关签名的选项的更多信息，请参阅《AWS 一般参考》中的将签名添加到 HTTP 请求。

2. POST 策略必须包含以下元素：

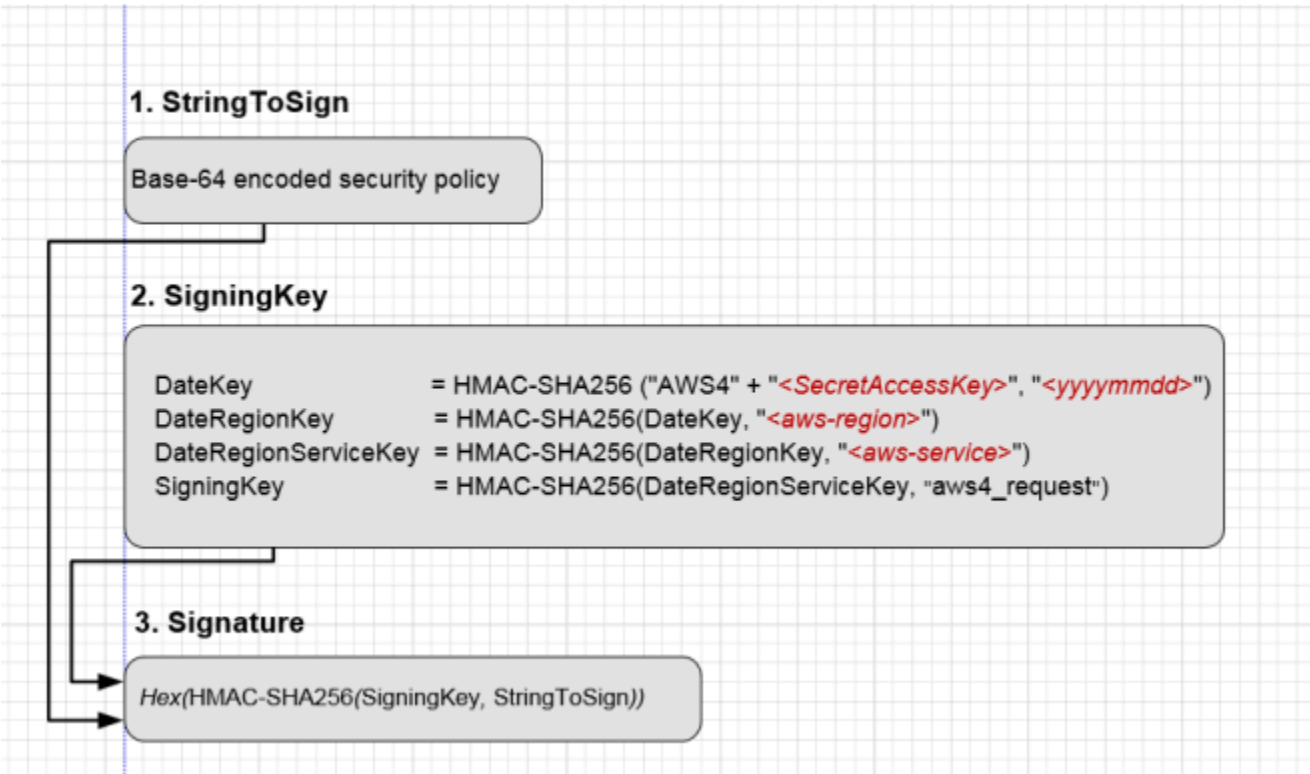
元素名称	说明
x-amz-algorithm	用于计算签名的签名算法。对于 AWS 签名版本 4，该值为 AWS4-HMAC-SHA256 。
x-amz-credential	除了访问密钥 ID 外，此元素还提供用于计算签名密钥的范围信息，以便进行签名计算。 它是采用以下形式的字符串： <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> 例如，

元素名称	说明
	AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .
x-amz-date	以 ISO8601 格式字符串指定的日期值。例如，“20130728T000000Z”。该日期必须与您在创建用于计算签名的签名密钥时使用的日期相同。

3. 对于签名计算，POST 策略是待签字符串。

计算签名

下图说明了签名计算过程。



计算签名

1. 使用 UTF-8 编码创建策略。
2. 将 UTF-8 编码的策略转换为 Base64。结果是待签字符串。

3. 将签名创建为待签字符串的 HMAC-SHA256 哈希值。您将提供签名密钥作为哈希函数的密钥。
4. 使用十六进制编码对签名进行编码。

有关创建 HTML 表单、安全策略和示例的更多信息，请参阅以下子主题：

- [创建 HTML 表单 \(使用 AWS 签名版本 4\)](#)
- [POST 策略](#)
- [示例：使用 HTTP POST 的基于浏览器的上传 \(使用 AWS 签名版本 4\)](#)

特定于 Amazon S3 签名版本 4 身份验证的策略键

下表显示了与 Amazon S3 签名版本 4 身份验证相关的策略键，这些键可以包含在 Amazon S3 策略中。您可以在存储桶策略中添加这些条件，以便在使用签名版本 4 对请求进行身份认证时强制执行特定的行为。有关示例策略，请参阅 [使用与签名版本 4 有关的条件键的存储桶策略示例](#)。

适用的键	说明
s3:signatureversion	标识您希望支持的 AWS 签名的版本，以将其用于经过身份验证的请求。对于经身份验证的请求，Amazon S3 同时支持签名版本 4 和签名版本 2。您可以在存储桶策略中添加此条件以要求使用特定的签名版本。 有效值： "AWS" 标识签名版本 2 "AWS4-HMAC-SHA256" 标识签名版本 4
s3:authType	Amazon S3 支持多种身份认证方法 [请参阅对请求进行身份验证 (AWS 签名版本 4)]。您可以选择使用此条件键来限制传入的请求使用特定的身份认证方法。例如，您可以仅支持在请求身份认证中使用 HTTP Authorization 标头。 有效值：

适用的键	说明
	REST-HEADER
	REST-QUERY-STRING
	POST
s3:signatureAge	签名在已通过身份认证的请求中的有效期 (以毫秒为单位) 。 这种情况适用于： • 预签名 URL：最严格的条件胜出。有关更多信息，请参阅使用预签名 URL。 • 预签名 POST：使用预签名 POST 将文件直接上传到 S3。有关更多信息，请参阅 Amazon S3 POST 策略。 在签名版本 2 中，此值始终设置为 0。 在签名版本 4 中，签名键的有效期最长为 7 天。因此，签名的有效期最初也为 7 天。您可以使用此条件进一步限制签名有效期。有关更多信息，请参阅 对请求进行签名简介。 示例值：100

适用的键	说明
s3:x-amz-content-sha256	您可以使用此条件键禁止存储桶中未签名的内容。 使用签名版本 4 时，对于使用 Authorization 标头的请求，您需要在签名计算中添加 x-amz-content-sha256 标头，然后将它的值设置为哈希负载。请注意，此条件键不支持将 x-amz-content-sha256 标头作为查询字符串参数。 您可以在存储桶策略中使用此条件键来拒绝任何未对有效载荷进行签名的上传。例如，您可以拒绝使用 Authorization 标头来对请求进行身份认证但未对有效载荷进行签名的上传。有关更多信息，请参阅 Authorization 标头的签名计算：在单个分块中传输有效载荷 (AWS 签名版本 4)。 有效值：UNSIGNED-PAYLOAD

使用与签名版本 4 有关的条件键的存储桶策略示例

以下存储桶策略将在签名的存在时间超过十分钟时，拒绝对 examplebucket 中对象的任何 Amazon S3 预签名 URL 请求。

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
      }
    }
  }
]
}

```

以下存储桶策略仅允许使用 Authorization 标头进行请求身份认证的请求。任何 POST 或预签名 URL 请求都将被拒绝。

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"
        }
      }
    }
  ]
}

```

以下存储桶策略拒绝使用预签名 URL 进行请求身份验证的请求：

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"
      }
    }
  }
]
}

```

以下存储桶策略拒绝任何带有未签名有效载荷的上传：

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"
        }
      }
    }
  ]
}

```

使用 POST 的基于浏览器的上传 (AWS 签名版本 4)

本节讨论如何使用 HTTP POST 请求通过浏览器将文件直接上传到 Amazon S3。它还包含有关如何使用 AWS Amplify JavaScript 库通过基于浏览器的上传将文件上传到 Amazon S3 的信息。

主题

- [POST Object](#)
- [POST Object restore](#)
- [使用 HTTP POST 的基于浏览器的上传](#)
- [计算签名](#)
- [创建 HTML 表单 \(使用 AWS 签名版本 4 \)](#)
- [POST 策略](#)
- [示例：使用 HTTP POST 的基于浏览器的上传 \(使用 AWS 签名版本 4 \)](#)
- [使用 AWS Amplify 库通过基于浏览器的上传来上传到 Amazon S3](#)

POST Object

说明

POST 操作使用 HTML 表单将对象添加到指定的存储桶。POST 是 PUT 的另一种形式，它支持基于浏览器的上传，以作为将对象放入存储桶的一种方式。通过 HTTP 标头传递给 PUT 的参数将作为表单字段传递到以分段/表单数据编码的消息正文中的 POST。要向存储桶添加对象，您必须对存储桶拥有 WRITE 访问权限。Amazon S3 从不存储不完整的对象。如果您收到成功的响应，则可以确信已存储整个对象。

Amazon S3 是一个分布式系统。除非您为存储桶启用了版本控制，否则，如果 Amazon S3 同时收到针对同一对象的多个写入请求，则只存储写入的对象的最后一个版本。

要确保数据在穿过网络时不会损坏，请使用 Content-MD5 表单字段。当您使用此表单字段时，Amazon S3 会根据提供的 MD5 值检查对象。如果它们不匹配，Amazon S3 将返回错误。此外，您可以在将对象发布到 Amazon S3 时计算 MD5 值，并将返回的 ETag 与计算出的 MD5 值进行比较。ETag 仅反映对于对象内容的更改，而不反映对于其元数据的更改。

Note

要将您的应用程序配置为在发送请求正文之前发送请求标头，请使用 HTTP 状态代码 100（继续）。对于 POST 操作，使用此状态代码有助于您避免在消息根据标头遭拒绝时（例如，由于身份验证失败或重定向）发送消息正文。有关 HTTP 状态代码 100（继续）的更多信息，请参阅 <http://www.ietf.org/rfc/rfc2616.txt> 的第 8.2.3 节。

Amazon S3 会自动加密上传到 S3 存储桶的所有新对象。已上传对象的加密设置取决于目标存储桶的默认加密配置。默认情况下，所有存储桶都有默认加密配置，该配置使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

如果目标存储桶的加密配置使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）或采用客户提供的加密密钥（SSE-C），则 Amazon S3 使用相应的 KMS 密钥或客户提供的密钥来加密已上传的对象。上传对象时，如果您想更改已上传对象的加密设置，可以指定服务器端加密的类型。您可以配置 SSE-S3、SSE-KMS、DSSE-KMS 或 SSE-C。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[使用服务器端加密保护数据](#)。

Important

构造请求时，请确保 `file` 字段是表单中的最后一个字段。

版本控制

如果您对存储桶启用版本控制，POST 会自动为要添加的对象生成唯一版本 ID。Amazon S3 使用 `x-amz-version-id` 响应标头在响应中返回此 ID。

如果您对存储桶暂停版本控制，Amazon S3 始终使用 `null` 作为存储在存储桶中的对象的版本 ID。

有关返回存储桶的版本控制状态的更多信息，请参阅 [GetBucketVersioning](#)。

Amazon S3 是一个分布式系统。如果对存储桶启用版本控制，并且 Amazon S3 同时收到针对同一对象的多个写入请求，则会存储该对象的所有版本。

要查看使用版本控制的示例请求，请参阅[示例请求](#)。

请求

语法

```
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--
```

请求参数

此操作的实施不使用请求参数。

表单字段

此操作可以使用以下表单字段。

名称	描述	必需
AWSAccessKeyId	存储桶拥有者的 AWS 访问密钥 ID，该拥有者向匿名用户授予对满足策略中一组约束的请求的访问权限。 类型：字符串 默认值：无 约束：如果策略文档随请求一起提供，则为必需项。	有条件
acl	指定的 Amazon S3 访问控制列表 (ACL)。如果指定的 ACL 无效，将生成错误。有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的访问控制列表 (ACL) 概述。 类型：字符串 默认值：private 有效值：private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	否
Cache-Control , Content-Type , Content-Disposition , Content-Encoding , Expires	特定于 REST 的标头。有关更多信息，请参阅PutObject。 类型：字符串 默认值：无	否
file	文件或文本内容。	是

名称	描述	必需
	文件或文本内容必须是表单中的最后一个字段。 您一次仅能上传一个文件。 类型：文件或文本内容 默认值：无	
key	已上传的键的名称。 要使用由用户提供的文件名，请使用 <code>\${filename}</code> 变量。例如，如果名为 Mary 的用户上传了文件 <code>example.jpg</code>，而您指定了 <code>/user/mary/\${filename}</code>，则键名称为 <code>/user/mary/example.jpg</code>。 有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的对象键和元数据。 类型：字符串 默认值：无	是
policy	描述请求中允许的内容的安全策略。不带安全策略的请求被认为是匿名的，只在公共可写的存储桶上有效。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的HTML 表单和上传示例。 类型：字符串 默认值：无 约束：如果存储桶不是可公开写入的，则需要安全策略。	有条件

名称	描述	必需
success_action_redirect , redirect	上传成功后客户端重定向到的 URL。 如果未指定 success_action_redirect ， Amazon S3 将返回在 success_action_status 字段中指定的空文档类型。 如果 Amazon S3 无法解释该 URL ， 则其行为就像该字段不存在一样。 如果上传失败 ， Amazon S3 将显示错误并且不会将用户重定向到某个 URL。 类型：字符串 默认值：无  Note 已弃用 redirect 字段名称，并且以后将移除对 redirect 字段名称的支持。	否
success_action_status	如果您未指定 success_action_redirect ， 则上传成功后，状态代码将返回给客户端。 此字段接受值 200、201 或 204 (默认值) 。 如果值设置为 200 或 204 ， Amazon S3 将返回一个空文档和一个 200 或 204 状态代码。 如果值设置为 201 ， Amazon S3 将返回一个 XML 文档和一个 201 状态代码。 如果没有设置值或设置为无效的值 ， Amazon S3 将返回一个空文档和一个 204 状态代码。 类型：字符串 默认值：无	否

名称	描述	必需
tagging	要添加到对象的指定标签集。要添加标签，请使用以下编码方案。 <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre> 有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的为对象添加标签。 类型：字符串 默认值：无	否
x-amz-storage-class	要用于存储对象的存储类别。如果您未指定存储类别，Amazon S3 将使用默认存储类别 STANDARD。Amazon S3 支持其它存储类别。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的存储类别。 类型：字符串 默认：STANDARD 有效值：REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	否

名称	描述	必需
x-amz-meta-*	以前缀开头的标头是用户定义的元数据。每个元数据均以一组键-值对的形式存储和返回。Amazon S3 不验证或解释用户定义的元数据。有关更多信息，请参阅 PutObject。 类型：字符串 默认值：无	否
x-amz-security-token	Amazon DevPay 安全令牌。 使用 Amazon DevPay 的每个请求都需要两个 x-amz-security-token 表单字段：一个用于产品令牌，一个用于用户令牌。 类型：字符串 默认值：无	否
x-amz-signature	(AWS 签名版本 4) 安全策略的 HMAC-SHA256 哈希值。 类型：字符串 默认值：无	有条件

名称	描述	必需
x-amz-website-redirect-location	如果存储桶配置为网站，则此字段会将对此对象的请求重定向到同一存储桶中的另一个对象，或者重定向到外部 URL。Amazon S3 将此标头的值存储在对象元数据中。有关对象元数据的信息，请参阅《Amazon Simple Storage Service 用户指南》中的对象键和元数据。 在以下示例中，请求标头将重定向设置为同一存储桶中的对象（anotherPage.html）： <pre>x-amz-website-redirect-location: /anotherPage.html</pre> 在以下示例中，请求标头将对象重定向设置为另一个网站： <pre>x-amz-website-redirect-location: http://www.example.com/</pre> 有关 Amazon S3 中的网站托管的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的在 Amazon S3 上托管网站和如何配置网站页面重定向。 类型：字符串 默认值：无 约束：该值必须以 /、http:// 或 https:// 为前缀。值的长度限制为 2 KB。	否

其它校验和请求表单字段

上传对象时，您可以指定要用于验证数据完整性的各种校验和。您可以指定一个额外的校验和算法供 Amazon S3 使用。有关其它校验和值的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[检查对象完整性](#)。

名称	描述	必填
x-amz-checksum-algorithm	表示用于为对象创建校验和的算法。如果指定了值，则必须包括匹配的校验和标头。否则，您的请求将生成 400 错误。 可能的值包括 CRC32、CRC32C、SHA1 和 SHA256。	否
x-amz-checksum-crc32	指定对象的 base64 编码的 32 位 CRC32 校验和。 如果 x-amz-checksum-algorithm 的值为 CRC32，则此参数是必需的。	有条件
x-amz-checksum-crc32c	指定对象的 base64 编码的 32 位 CRC32C 校验和。 如果 x-amz-checksum-algorithm 的值为 CRC32C，则此参数是必需的。	有条件
x-amz-checksum-sha1	指定对象的 base64 编码的 160 位 SHA-1 摘要。 如果 x-amz-checksum-algorithm 的值为 SHA1，则此参数是必需的。	有条件
x-amz-checksum-sha256	指定对象的 base64 编码的 256 位 SHA-256 摘要。 如果 x-amz-checksum-algorithm 的值为 SHA256，则此参数是必需的。	有条件

特定于服务器端加密的请求表单字段

服务器端加密是静态数据加密。Amazon S3 在将数据写入 AWS 数据中心的磁盘时对数据进行加密，并在您访问数据时对其进行解密。上传对象时，您可以指定您希望 Amazon S3 用于加密对象的服务器端加密的类型。

服务器端加密有四种类型：

- 具有 Amazon S3 托管式密钥的服务器端加密 (SSE-S3)：从 2022 年 5 月开始，所有 Amazon S3 存储桶都默认配置了加密。默认的服务器端加密选项是使用 SSE-S3。每个对象都使用唯一的密钥来进行加密。作为额外的保护措施，SSE-S3 使用定期轮换的根密钥加密密钥本身。SSE-S3 使用可用的最强数据块密码之一 [即 256 位高级加密标准 (AES-256)] 来加密您的数据。

- 具有 AWS KMS 密钥的服务器端加密 (SSE-KMS) ：通过将 AWS KMS 服务与 Amazon S3 集成来提供 SSE-KMS。使用 AWS KMS ，您可以更好地控制您的密钥。例如，您可以查看单独的密钥、编辑控制策略以及遵循 AWS CloudTrail 中的密钥。此外，您还可以创建和管理客户自主管理型密钥，或者使用对于您、服务和区域为唯一的 AWS 托管式密钥。
- 具有 AWS KMS 密钥的双层服务器端加密 (DSSE-KMS) ：具有 AWS KMS keys 的双层服务器端加密 (DSSE-KMS) 与 SSE-KMS 类似，但前者应用的是两层单独的对象级加密，而不是一层。
- 具有客户提供的密钥的服务器端加密 (SSE-C) ：使用 SSE-C ，您管理加密密钥，而 Amazon S3 管理加密 (在它对磁盘进行写入时) 和解密 (在您访问您的对象时) 。

 Note

在对通用存储桶阻止了具有客户提供密钥的服务器端加密 (SSE-C) 的情况下，如果在向存储桶写入新数据时指定 SSE-C 请求标头，您将收到“HTTP 403 拒绝访问”错误。有关更多信息，请参阅[对通用存储桶阻止或取消阻止 SSE-C](#)。

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[使用服务器端加密保护数据](#)。

根据您要使用的服务器端加密类型，指定以下表单字段。

- 使用 SSE-S3、SSE-KMS 或 DSSE-KMS ：如果您要使用这些类型的服务器端加密，请在请求中指定以下表单字段。

名称	描述	必填
x-amz-server-side-encryption	指定 Amazon S3 创建对象时要使用的服务器端加密算法。要使用 SSE-S3，请指定 AES256。要使用 SSE-KMS，请指定 aws:kms。要使用 DSSE-KMS，请指定 aws:kms:dsse 。 类型：字符串 有效值：aws:kms、AES256、aws:kms:dsse	是
x-amz-server-side-encryption	如果 x-amz-server-side-encryption 标头的有效值为 aws:kms 或 aws:kms:dsse ，则此标头指定用于加密对象的 AWS KMS key 的 ID。	是，如果 x-amz-ser

名称	描述	必填
n-aws-kms-key-id	类型：字符串	ver-side-encryption 的值为 aws:kms 或 aws:kms:dsse
x-amz-server-side-encryption-context	如果 x-amz-server-side-encryption 的有效值为 aws:kms 或 aws:kms:dsse ，则此标头指定对象的加密上下文。此标头的值是 base64 编码的 UTF-8 字符串，其中包含加密上下文的 JSON 格式的键-值对。 类型：字符串	否
x-amz-server-side-encryption-bucket-key-enabled	如果 x-amz-server-side-encryption 的有效值为 aws:kms 或 aws:kms:dsse ，则此标头指定 Amazon S3 是否应将 S3 存储桶密钥与 SSE-KMS 或 DSSE-KMS 结合使用。将此标头设置为 true ，会导致 Amazon S3 将 S3 存储桶密钥与 SSE-KMS 或 DSSE-KMS 结合使用来进行对象加密。 类型：布尔值	否

 Note

如果您指定 x-amz-server-side-encryption:aws:kms 或 x-amz-server-side-encryption:aws:kms:dsse ，但未提供 x-amz-server-side-encryption-aws-kms-key-id ，Amazon S3 将使用 AWS 托管式密钥 (aws/S3) 来保护数据。

- 使用 SSE-C：如果您想管理自己的加密密钥，则必须在请求中提供以下所有表单字段。

 Note

如果您使用 SSE-C，Amazon S3 在响应中返回的 ETag 值不是对象的 MD5。

名称	描述	必填
x-amz-server-side-encryption-customer-algorithm	指定加密对象时要使用的算法。 类型：字符串 默认值：无 有效值：AES256 约束：必须附有有效的 x-amz-server-side-encryption-customer-key 和 x-amz-server-side-encryption-customer-key-MD5 字段。	是
x-amz-server-side-encryption-customer-key	指定 Amazon S3 加密数据时要使用的客户提供的 base64 编码的加密密钥。此值用于存储对象，然后将其丢弃。Amazon 不存储加密密钥。密钥必须适合用于 x-amz-server-side-encryption-customer-algorithm 标头中指定的算法。 类型：字符串 默认值：无 约束：必须附有有效的 x-amz-server-side-encryption-customer-algorithm 和 x-amz-server-side-encryption-customer-key-MD5 字段。	是

名称	描述	必填
x-amz-server-side-encryption-customer-key-MD5	根据 RFC 1321 指定加密密钥的 base64 编码的 128 位 MD5 摘要。Amazon S3 使用此标头进行消息完整性检查，以确保加密密钥传输无误。 类型：字符串 默认值：无 约束：必须附有有效的 x-amz-server-side-encryption-customer-algorithm 和 x-amz-server-side-encryption-customer-key 字段。	是

响应

响应标头

除了所有响应通用的响应标头之外，此操作的实现还可以包括以下响应标头。有关更多信息，请参阅[通用响应标头](#)。

名称	描述
x-amz-checksum-crc32	对象的 base64 编码的 32 位 CRC32 校验和。 类型：字符串
x-amz-checksum-crc32c	对象的 base64 编码的 32 位 CRC32C 校验和。 类型：字符串
x-amz-checksum-sha1	对象的 base64 编码的 160 位 SHA-1 摘要。 类型：字符串
x-amz-checksum-sha256	对象的 base64 编码的 256 位 SHA-256 摘要。 类型：字符串

名称	描述
x-amz-expiration	如果在存储桶的生命周期配置中为对象配置了 Expiration 操作，则 Amazon S3 会返回此标头。标头值包括一个 expiry-date 组件和一个 URL 编码的 rule-id 组件。对于启用版本控制的存储桶，此标头仅适用于当前版本。Amazon S3 不提供标头来指示非当前版本何时符合永久删除条件。有关更多信息，请参阅 PutBucketLifecycleConfiguration。 类型：字符串
success_action_redirect, redirect	上传成功后客户端重定向到的 URL。 类型：字符串 原级：PostResponse
x-amz-server-side-encryption	在 Amazon S3 中存储此对象时使用的服务器端加密算法（例如 AES256、aws:kms、aws:kms:dsse）。 类型：字符串
x-amz-server-side-encryption-aws-kms-key-id	如果 x-amz-server-side-encryption 标头的有效值为 aws:kms，则此标头指定用于加密对象的 KMS 密钥的 ID。 类型：字符串
x-amz-server-side-encryption-bucket-key-enabled	如果 x-amz-server-side-encryption 的有效值为 aws:kms，则此标头表示是否使用 S3 存储桶密钥通过 SSE-KMS 为对象加密。如果此标头设置为 true，则对象会将 S3 存储桶密钥与 SSE-KMS 结合使用。 类型：布尔值

名称	描述
x-amz-server-side-encryption-customer-algorithm	如果请求了 SSE-C，则响应将包含此标头，用于确认所使用的加密算法。 类型：字符串 有效值：AES256
x-amz-server-side-encryption-customer-key-MD5	如果请求了 SSE-C，则响应将包含此标头，用于验证客户提供的加密密钥的往返消息完整性。 类型：字符串
x-amz-version-id	对象的版本。 类型：字符串

响应元素

名称	描述
Bucket	存储对象的存储桶的名称。 类型：字符串 原级：PostResponse
ETag	实体标签 (ETag) 是对象的 MD5 哈希值，使用该值，您可以通过在 GET 请求操作中使用 If-Modified 请求标签来执行有条件 GET 操作。ETag 仅反映对于对象内容的更改，而不反映对于其元数据的更改。 类型：字符串 原级：PostResponse
Key	对象键名称。 类型：字符串

名称	描述
	原级：PostResponse
Location	对象的 URI。 类型：字符串 原级：PostResponse

特殊错误

此操作的实施不返回特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

示例

示例请求

```
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
```

暂停版本控制时的响应示例

以下是暂停存储桶版本控制时的响应示例：

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIx1+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
```

在此响应中，版本 ID 为 null。

启用版本控制时的响应示例

以下是启用存储桶版本控制时的响应示例。

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIx1+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
```

相关资源

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

POST Object restore

说明

此操作执行以下类型的请求：

- `select`：对归档的对象执行 `select` 查询
- `restore an archive`：还原归档的对象

要使用此操作，您必须拥有执行 `s3:RestoreObject` 和 `s3:GetObject` 操作的权限。存储桶所有者默认拥有此权限，并可以将此权限授予他人。有关权限的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[与存储桶子资源操作相关的权限](#)和[管理对 Amazon S3 资源的访问权限](#)。

使用 Select 请求查询归档

您可以使用 `select` 类型的请求对归档的对象执行 SQL 查询。要由 `select` 请求查询的归档对象必须格式化为未压缩的逗号分隔值 (CSV) 文件。您可以针对归档的数据运行查询和自定义分析，而无需将数据还原到较热的 Amazon S3 层。有关 `select` 请求的概述，请参阅《Amazon Simple Storage Service 用户指南》中的[查询归档对象](#)。

在发出 `select` 请求时，请执行以下操作：

- 为 `select` 查询的输出定义输出位置。此位置必须是一个 Amazon S3 存储桶，与包含要查询的归档对象的存储桶位于同一 AWS 区域。启动作业的 AWS 账户必须拥有写入 S3 存储桶的权限。您可以为存储在存储桶中的输出对象指定存储类别和加密。有关输出的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[查询归档对象](#)。

有关请求正文中 S3 结构的更多信息，请参阅以下内容：

- [PutObject](#)
- 《Amazon Simple Storage Service 用户指南》中的[使用 ACL 管理访问权限](#)
- 《Amazon Simple Storage Service 用户指南》中的[使用服务器端加密保护数据](#)
- 在请求正文的 `SelectParameters` 结构中为查询的 `SELECT` 还原类型定义 SQL 表达式。您可以使用类似以下示例的表达式。
 - 下面的表达式返回指定对象中的所有记录。

```
SELECT * FROM Object
```

- 假设您没有对存储在对象中的数据使用任何标头，则可以使用位置标头指定列。

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- 如果您具有标头并且在请求正文的 CSV 结构中将 fileHeaderInfo 设置为 USE，则可以在查询中指定标头。(如果将 fileHeaderInfo 字段设置为 IGNORE，则查询会跳过第一行。) 您无法将序号位置与标题列名混合。

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

有关将 SQL 与 Amazon Glacier Select 还原结合使用的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [Amazon S3 Select 和 Amazon Glacier Select 的 SQL 参考](#)。

在发出 select 请求时，您还可以执行以下操作：

- 要加快您的查询，请指定 Expedited 层。有关层的更多信息，请参阅本主题后面的“还原归档”。
- 指定有关要查询的输入对象和 CSV 编码查询结果序列化的数据序列化格式的详细信息。

以下是有关 select 功能的其它重要信息：

- 输出结果为新的 Amazon S3 对象。与归档检索不同，这些对象会一直存储，直至通过手动方式或生命周期策略进行显式删除。
- 您可以对同一 Amazon S3 对象发出多个 select 请求。Amazon S3 不会删除重复的请求，因此请避免发出重复的请求。
- 即使对象已经还原，Amazon S3 也会接受 select 请求。select 请求不会返回错误响应 409。

还原归档

GLACIER 和 DEEP_ARCHIVE 存储类别中的对象已归档。要访问归档的对象，您必须先发起还原请求。这会还原归档对象的临时副本。在还原请求中，指定您希望还原的副本存在的天数。指定期限过后，Amazon S3 会删除临时副本，但对象仍会以原先还原时所处的 GLACIER 或 DEEP_ARCHIVE 存储类别进行归档。

要还原特定的对象版本，您可以提供版本 ID。如果您未提供版本 ID，Amazon S3 将还原当前版本。

完成还原作业所需的时间取决于从哪个存储类别还原对象以及您指定的数据访问层。

还原归档对象（或使用 select 请求）时，您可以在请求正文的 Tier 元素中指定以下数据访问层选项之一：

- **Expedited** : 加速检索可让您在偶尔需要紧急请求归档的子集时, 快速访问存储在 GLACIER 存储类别中的数据。对于除最大型归档对象 (250 MB+) 之外的所有其它归档, 使用加速检索访问的数据通常在 1 到 5 分钟内提供。预配置容量确保在您需要时, 可以使用针对加速检索的检索容量。DEEP_ARCHIVE 存储类别不提供加速检索和预调配容量。
- **Standard** - 标准检索允许您在数小时内访问您的任何存档对象。这是未指定检索选项的 GLACIER 和 DEEP_ARCHIVE 检索请求的默认选项。从 GLACIER 存储类别完成标准检索通常需要 3-5 小时, 从 DEEP_ARCHIVE 存储类别进行标准检索通常会在 12 小时内完成。
- **Bulk** : 批量检索是 Amazon Glacier 的成本最低的检索选项, 使您能够在一天内以较低的成本检索大量 (甚至是 PB 级) 的数据。从 GLACIER 存储类别完成批量检索通常需要 5-12 小时, 从 DEEP_ARCHIVE 存储类别进行批量检索通常会在 48 小时内完成。

有关 Expedited 数据访问的归档检索选项和预调配容量的更多信息, 请参阅《Amazon Simple Storage Service 用户指南》中的[还原已归档的对象](#)。

您可以使用 Amazon S3 还原速度升级, 以便在还原进行期间将还原速度更改为更快的速度。通过向同一对象发出另一个还原请求并设置新的 Tier 请求元素, 可以提升正在进行的还原的速度。在发出升级还原层的请求时, 您必须选择比正在进行的还原所使用的层更快的层。您不得更改任何其他参数, 如 Days 请求元素。有关更多信息, 请参阅《Amazon Simple Storage Service 用户指南》中的[升级正在进行的还原的速度](#)。

要获取对象还原的状态, 您可以发送 HEAD 请求。操作会在响应中返回 x-amz-restore 标头, 该标头提供有关还原状态的信息。您可以使用 Amazon S3 事件通知在还原启动或完成时通知您。有关更多信息, 请参阅《Amazon Simple Storage Service 用户指南》中的[配置 Amazon S3 事件通知](#)。

还原归档对象后, 您可以通过重新发出具有新时段的请求来更新还原时段。Amazon S3 会相对于当前时间更新还原期限, 并且仅针对该请求收费, 而不收取数据传输费用。当 Amazon S3 正在处理您当前对该对象的还原请求时, 您无法更新还原期限。

如果存储桶的生命周期配置带有包含过期操作的规则, 则对象过期时间将覆盖您在还原请求中指定的生命期。例如, 如果您将对象副本的还原期限设为 10 天, 但该对象计划在 3 天后过期, 则 Amazon S3 会在 3 天后删除该对象。有关生命周期配置的更多信息, 请参阅[PutBucketLifecycleConfiguration](#) 和《Amazon Simple Storage Service 用户指南》中的[对象生命周期管理](#)。

请求

语法

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
```

```
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see #####AWS ##### 4#)
Content-MD5: MD5

request body
```

 **Note**

语法显示了一些请求标头。有关完整列表，请参阅本主题后面的“请求标头”。

请求参数

此操作的实施不使用请求参数。

请求标头

名称	描述	必填
Content-MD5	数据的 base64 编码的 128 位 MD5 摘要。您必须将此标头用作消息完整性检查，以验证请求正文在传输过程中是否未损坏。有关更多信息，请参阅 RFC 1864。 类型：字符串 默认值：无	是

请求元素

以下是用于还原归档的请求正文的 XML 示例。

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

下表说明了请求正文中用于还原归档的 XML。

名称	描述	必填
RestoreRequest	还原信息的容器。 类型：容器	是
Days	已还原（活动）副本的生命周期。您可以从 Amazon Glacier 还原对象的最少天数为 1。在对象副本达到指定的生命周期后，Amazon S3 会将其从存储桶中移除。如果要还原归档，则此元素是必需的。 请勿在 SELECT 类型的请求中使用此元素。 类型：正整数 原级：RestoreRequest	是，如果还原归档
GlacierJobParameters	S3 Glacier 存储类别作业参数的容器。 请勿在 SELECT 类型的请求中使用此元素。 类型：容器 原级：RestoreRequest	否
Tier	还原归档时要使用的数据访问层。Standard 是默认值。 类型：Enum 有效值：Expedited Standard Bulk 原级：GlacierJobParameters	否

以下 XML 是对归档的对象进行 select 查询的请求正文：

```
<RestoreRequest>
```

```
<Type>SELECT</Type>
<Tier>Expedited</Tier>
<Description>Job description</Description>
<SelectParameters>
  <Expression>Select * from Object</Expression>
  <ExpressionType>SQL</ExpressionType>
  <InputSerialization>
    <CSV>
      <FileHeaderInfo>IGNORE</FileHeaderInfo>
      <RecordDelimiter>\n</RecordDelimiter>
      <FieldDelimiter>,</FieldDelimiter>
      <QuoteCharacter>"</QuoteCharacter>
      <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
      <Comments>#</Comments>
    </CSV>
  </InputSerialization>
  <OutputSerialization>
    <CSV>
      <QuoteFields>ASNEEDED</QuoteFields>
      <RecordDelimiter>\n</RecordDelimiter>
      <FieldDelimiter>,</FieldDelimiter>
      <QuoteCharacter>"</QuoteCharacter>
      <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
    </CSV>
  </OutputSerialization>
</SelectParameters>
<OutputLocation>
  <S3>
    <BucketName>Name of bucket</BucketName>
    <Prefix>Key prefix</Prefix>
    <CannedACL>Canned ACL string</CannedACL>
    <AccessControlList>
      <Grantee>
        <Type>Grantee Type</Type>
      </Grantee>
    </AccessControlList>
    <Encryption>
      <EncryptionType>Encryption type</EncryptionType>
      <KMSKeyId>KMS Key ID</KMSKeyId>
    </Encryption>
  </S3>
</OutputLocation>
</Job>
```

```
<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>
```

下表说明了请求正文中用于 SELECT 还原类型的 XML。

名称	描述	必填
RestoreRequest	还原信息的容器。 类型：容器	是
Tier	还原归档时要使用的数据访问层。Standard 是默认值。 类型：Enum 有效值：Expedited Standard Bulk 原级：RestoreRequest	否
Description	请求的可选描述。 类型：字符串	否

名称	描述	必填
	原级：RestoreRequest	
SelectParameters	描述 select 作业请求的参数。 类型：容器 原级：RestoreRequest	是，如果请求类型是 SELECT
OutputLocation	描述用于接收 select 还原请求结果的位置。 类型：Amazon S3 的容器 原级：RestoreRequest	是，如果请求类型是 SELECT

SelectParameters 容器元素包含以下元素。

名称	描述	必填
Expression	SQL 表达式。例如： - 以下 SQL 表达式从以 CSV 格式存储的对象中检索第一列数据： SELECT s._1 FROM Object s - 以下 SQL 表达式返回对象中的所有内容： SELECT * FROM Object 类型：字符串 原级：SelectParameters	是
ExpressionType	标识表达式类型。	是

名称	描述	必填
	类型：字符串 有效值：SQL 原级：SelectParameters	
InputSerialization	描述对象的序列化格式。 类型：CSV 的容器 原级：SelectParameters	是
OutputSerialization	描述如何序列化 select 任务的结果。 类型：CSV 的容器 原级：SelectParameters	是

InputSerialization 元素中的 CSV 容器元素包含以下元素。

名称	描述	必填
RecordDelimiter	用于分隔输入中各个记录的单个字符。您可以指定任意分隔符，而不是默认值。 类型：字符串 默认：\n 原级：CSV	否
FieldDelimiter	用于分隔记录中各个字段的单个字符。您可以指定任意分隔符。 类型：字符串 默认：,	否

名称	描述	必填
	原级：CSV	
QuoteCharacter	当字段分隔符是值的一部分时，用于转义的单字符。 考虑 CSV 文件中的以下示例： "a, b" 用引号将值括起来，可使该值成为单个字段。如果不使用引号，则逗号是字段分隔符（这使其成为两个单独的字段值，a 和 b）。 类型：字符串 默认：" 原级：CSV	否
QuoteEscapeCharacter	用于对已转义值内的引号字符进行转义的单字符。例如，值"""" a , b """" 解析为 " a , b "。 类型：字符串 默认：" 原级：CSV	否

名称	描述	必填
FileHeaderInfo	描述输入数据中的第一行。它是 ENUM 值之一。 - NONE：第一行不是标题。 - IGNORE：第一行是标题，但不能使用标题值来指示表达式中的列。您可以使用列位置（例如 _1、_2、...）来指示列 (SELECT s._1 FROM OBJECT s)。 - Use：第一行是标题，您可以使用标题值来标识表达式 (SELECT "name" FROM OBJECT) 中的列。 类型：Enum 有效值：NONE USE IGNORE 原级：CSV	否
Comments	单个字符，用于指示在行的开头出现该字符时应忽略该行。您可以指定任何字符来表示注释行。 类型：字符串 原级：CSV	否

CSV 容器元素（在 **OutputSerialization** 元素中）包含以下元素。

名称	描述	必填
QuoteFields	指示是否在输出字段周围使用引号。 - ALWAYS：始终为输出字段使用引号。 - ASNEEDED：根据需要为输出字段使用引号。	否

名称	描述	必填
	类型：Enum 有效值：ALWAYS ASNEEDED 默认值：AsNeeded 原级：CSV	
RecordDelimiter	用于分隔输出中各个记录的单个字符。您可以指定任意分隔符，而不是默认值。 类型：字符串 默认：\n 原级：CSV	否
FieldDelimiter	用于分隔记录中各个字段的单个字符。您可以指定任意分隔符。 类型：字符串 默认：, 原级：CSV	否
QuoteCharacter	当字段分隔符是值的一部分时，用于转义的单个字符。例如，如果值为 a, b，则 Amazon S3 将此字段值用引号括起来，如下所示：" a , b "。 类型：字符串 默认：" 原级：CSV	否

名称	描述	必填
QuoteEscapeCharacter	用于对已转义值内的引号字符进行转义的单个字符。例如，如果值为 " a , b "，则 Amazon S3 将此值用引号括起来，如下所示：" " a , b " "。 类型：字符串 原级：CSV	否

S3 容器元素 (在 **OutputLocation** 元素中) 包含以下元素。

名称	描述	必填
AccessControlList	控制对暂存结果的访问权限的授权列表。 类型：授权的容器 原级：S3	否
BucketName	存储 select 还原结果的 S3 存储桶的名称。该存储桶必须与包含输入归档对象的存储桶位于同一个 AWS 区域中。 类型：字符串 原级：S3	是
CannedACL	要应用于 select 还原结果的标准访问控制列表 (ACL)。 类型：字符串 有效值： private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	否

名称	描述	必填
	原级：S3	
Encryption	包含所存储结果的加密信息。 类型：加密的容器 原级：S3	否
Prefix	附加到 select 还原结果前的前缀。前缀的最大长度是 512 字节。 类型：字符串 原级：S3	是
StorageClass	用于存储 select 请求结果的存储类别。 类型：字符串 有效值：STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA 原级：S3	否
Tagging	标签信息的容器。 类型：标签结构 原级：S3	否
UserMetadata	包含要与 select 还原结果一起存储的元数据的列表。 类型：MetadataEntry 结构 原级：S3	否

被授权者容器元素 (在 **AccessControlList** 元素中) 包含以下元素。

名称	描述	必填
DisplayName	被授权者的屏幕名称。 类型：字符串 原级：Grantee	否
EmailAddress	被授权者的电子邮件地址。 类型：字符串 原级：Grantee	否
ID	被授权者的规范用户 ID。 类型：字符串 原级：Grantee	否
Type	被授权者的类型。 类型：字符串 原级：Grantee	否
URI	被授权者组的 URI。 类型：字符串 原级：Grantee	否
Permission	授予的权限。 类型：字符串	否

名称	描述	必填
	原级：Grantee	

Encryption 容器元素 (在 S3 中) 包含以下元素。

名称	描述	必填
EncryptionType	存储作业结果时使用的服务器端加密算法。默认值为无加密。 类型：字符串 有效值 aws:kms AES256 原级：Encryption	否
KMSContext	可选。如果加密类型是 aws:kms，您可以使用此值为 select 还原结果指定加密上下文。 类型：字符串 原级：Encryption	否
KMSKeyId	用于对象加密的 AWS Key Management Service (AWS KMS) 密钥 ID。 类型：字符串 原级：Encryption	否

TagSet 容器元素 (在 Tagging 元素中) 包含以下元素。

名称	描述	必填
Tag	包含标签。 类型：容器	否

名称	描述	必填
	原级：TagSet	

标签容器元素 (在 **TagSet** 元素中) 包含以下元素。

名称	描述	必填
Key	标签的名称。 类型：字符串 原级：Tag	否
Value	标签的值。 类型：字符串 原级：Tag	否

MetadataEntry 容器元素 (在 **UserMetadata** 元素中) 包含以下要与对象一起存储的键-值对元素。

名称	描述	必填
MetadataKey	元数据键。 类型：字符串 原级：	否
MetadataEntry	元数据值。 类型：字符串 原级：	否

响应

成功的操作将返回 200 OK 或 202 Accepted 状态代码。

- 如果之前未还原对象副本，则 Amazon S3 将在响应中返回 202 Accepted。
- 如果之前还原了对象副本，则 Amazon S3 将在响应中返回 200 OK。

响应标头

此操作的实施仅使用大多数响应通用的响应标头。有关更多信息，请参阅[通用响应标头](#)。

响应元素

此操作不返回响应元素。

特殊错误

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
RestoreAlreadyInProgress	对象还原已在进行中。（此错误不适用于 SELECT 类型请求。）	409 冲突	客户端
GlacierExpeditedRetrievalNotAvailable	S3 Glacier 存储类别的加速检索目前不可用。请稍后重试。（如果处理 Expedited 请求的容量不足，则返回该错误。此错误仅适用于 Expedited 检索，而不适用于 Standard 或 Bulk 检索。）	503	不适用

示例

使用加速检索选项还原对象，为期两天

以下还原请求使用加速检索选项从 Amazon Glacier 还原 photo1.jpg 对象的副本，为期两天。

```
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>
```

如果 examplebucket 没有已还原的对象副本，Amazon S3 将返回以下 202 Accepted 响应。

```
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
```

如果对象的副本已还原，Amazon S3 会返回 200 OK 响应，并且仅更新已还原副本的过期时间。

使用 SELECT 请求查询归档

以下是 select 还原请求示例。

```
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 返回以下 202 Accepted 响应。

```
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-
Orl2W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jk1-lvZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

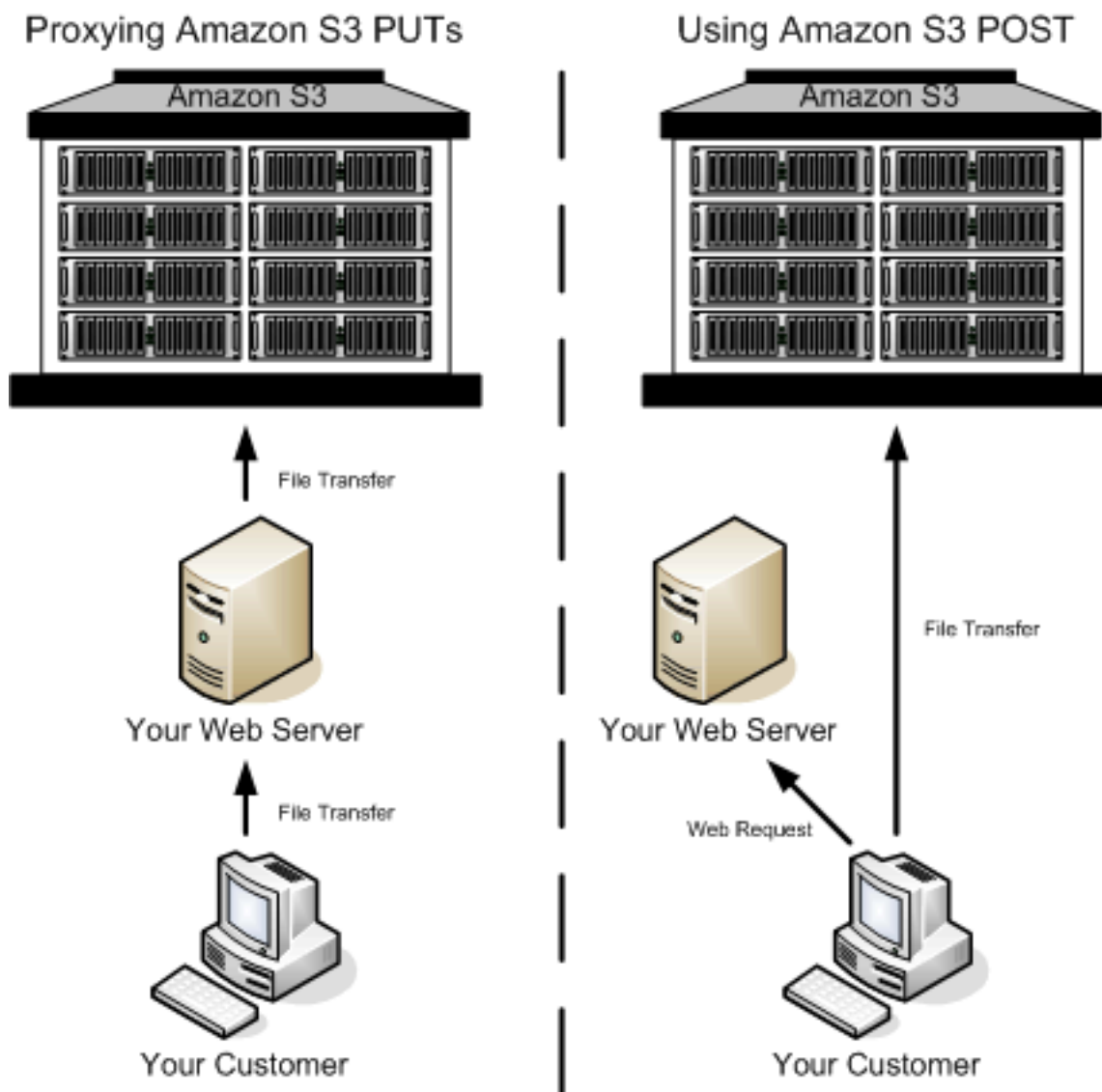
更多信息

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- 《Amazon Simple Storage Service 用户指南》中的 [Amazon S3 Select 和 Amazon Glacier Select 的 SQL 参考](#)

使用 HTTP POST 的基于浏览器的上传

Amazon S3 支持 HTTP POST 请求，以便用户可以直接将内容上传到 Amazon S3。通过使用 POST，最终用户可以对请求进行身份验证，而无需通过用于保护凭证的安全中间节点传递数据。因此，HTTP POST 有可能减少延迟。

下图演示了使用 POST 请求的 Amazon S3 上传。



1. 用户通过 Web 浏览器访问您的页面。
2. Web 页面包含一个 HTML 表单，其中包含用户将内容上传到 Amazon S3 所需的所有信息。
3. 用户通过 Web 浏览器将内容上传到 Amazon S3。

用于发送基于浏览器的 POST 请求的过程如下：

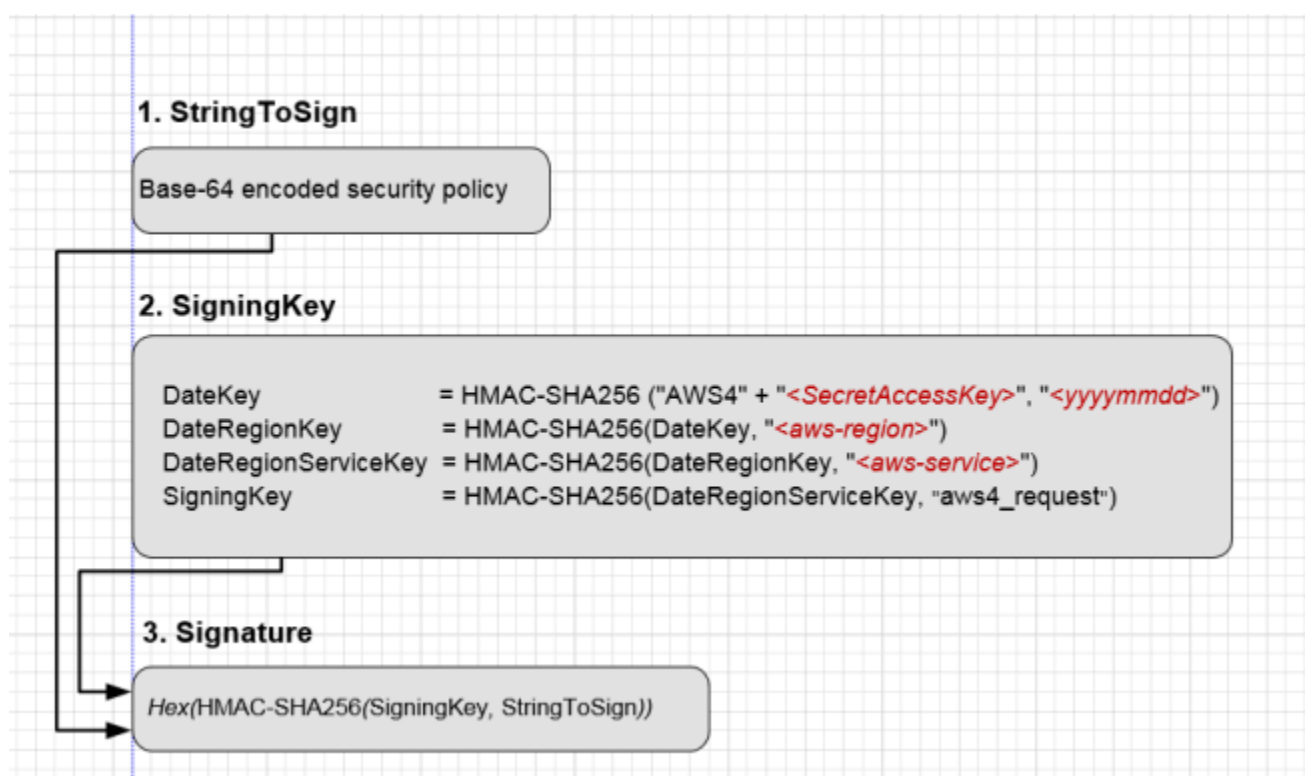
1. 创建安全策略，以指定条件来限制您希望请求中所支持的内容，例如，可以在其中上传对象的存储桶名称，以及对于正在创建的对象要支持的密钥名称前缀。
2. 创建基于该策略的签名。对于经过身份验证的请求，此表单必须包含有效的签名和策略。
3. 创建一个 HTML 表单，供用户访问以便将对象上传到您的 Amazon S3 存储桶。

下一节介绍如何创建签名来对请求进行身份验证。有关创建表单和安全策略的信息，请参阅[创建 HTML 表单 \(使用 AWS 签名版本 4\)](#)。

计算签名

对于经过身份验证的请求，HTML 表单必须包含用于安全策略和签名的字段。

- 安全策略 (请参阅 [POST 策略](#)) 控制请求中支持的内容。
- 安全策略是签名计算中的 StringToSign (请参阅[对请求进行签名简介](#))。



计算签名

1. 使用 UTF-8 编码创建策略。
2. 将 UTF-8 编码的策略字节转换为 base64。结果为 StringToSign。
3. 创建签名密钥。
4. 使用签名密钥，通过 HMAC-SHA256 签名算法对 StringToSign 进行签名。

有关创建 HTML 表单、安全策略和示例的更多信息，请参阅以下内容：

- [创建 HTML 表单（使用 AWS 签名版本 4）](#)
- [POST 策略](#)
- [示例：使用 HTTP POST 的基于浏览器的上传（使用 AWS 签名版本 4）](#)

创建 HTML 表单（使用 AWS 签名版本 4）

主题

- [HTML 表单声明](#)
- [HTML 表单字段](#)

要支持用户使用其浏览器（HTTP POST 请求）将内容上传到 Amazon S3，应使用 HTML 表单。HTML 表单由表单声明和表单字段组成。表单声明包含关于请求的高级别信息。表单字段包含详细的请求信息。

本节介绍如何创建 HTML 表单。有关使用 HTTP POST 和相关签名计算来进行请求身份验证的基于浏览器的上传的工作示例，请参阅[示例：使用 HTTP POST 的基于浏览器的上传（使用 AWS 签名版本 4）](#)。

必须采用 UTF-8 编码表单和策略。您可以通过在 content 属性中指定 charset=UTF-8 来将 UTF-8 编码应用于表单。下面是 HTML 标题中 UTF-8 编码的示例。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

下面是请求标头中 UTF-8 编码的示例。

```
Content-Type: text/html; charset=UTF-8
```

 Note

表单数据和边界 (不包括文件的内容) 不得超过 20 KB。

HTML 表单声明

HTML 表单声明具有以下三个属性：

- **action**：用于处理请求的 URL，它必须设置为存储桶的 URL。例如，如果存储桶的名称为 `examplebucket`，则 URL 为 `http://examplebucket.s3.amazonaws.com/`。

 Note

在表单字段中指定键名。

- **method**：方法必须是 POST。
- **enctype**：必须针对文件上传和文本区域上传，将附件类型 (enctype) 设置为“分段/表单数据”。有关 enctype 的更多信息，请参阅 [RFC 1867](#)。

这是存储桶 `examplebucket` 的表单声明。

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

HTML 表单字段

下表介绍可以在表单中使用的字段的列表。除其它字段外，还有一个可用于对请求进行身份验证的签名字段。提供了一些字段供您指定签名计算算法 (`x-amz-algorithm`)、用于生成签名密钥的凭证范围 (`x-amz-credential`) 以及用于计算签名的日期 (`x-amz-date`)。Amazon S3 使用这些信息来重新创建签名。如果签名匹配，Amazon S3 将处理该请求。

 Note

变量 `${filename}` 将自动替换为用户提供的文件的名称，并且可由所有表单字段识别。如果浏览器或客户端提供指向文件的完整或部分路径，则只会使用跟在最后一个斜杠 (/) 或

反斜杠 (\) 之后的文本 (例如 , C:\Program Files\directory1\file.txt 解释为 file.txt) 。如果没有提供文件或文件名 , 变量将替换为空字符串。

如果您未提供经过身份验证的请求所需的元素 (例如 policy 元素) , 则该请求将被视为匿名请求 , 只有在将存储桶配置为进行公开读取和写入时 , 请求才会成功。

元素名称	说明	必填
acl	Amazon S3 访问控制列表 (ACL)。如果指定的 ACL 无效 , Amazon S3 会拒绝该请求。有关 ACL 的更多信息 , 请参阅使用 Amazon S3 ACL。 类型 : 字符串 默认值 : private 有效值 : private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	否
Cache-Control	特定于 REST 的标头。有关更多信息 , 请参阅PutObject。	否
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		
key	已上传的对象的键名称。 要使用由用户提供的文件名 , 请使用 <code>\${filename}</code> 变量。例如 , 如果您上传文件 photo1.jpg 并将 <code>/user/user1/\${filename}</code> 指	是

元素名称	说明	必填
	定为键名称，则该文件将存储为 <code>/user/user1/photo1.jpg</code> 。 有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的对象键和元数据。	
policy	以 base64 编码的安全策略，用于描述请求中允许的内容。对于经过身份验证的请求，需要一个策略。 不带安全策略的请求被认为是匿名的，只会在公共可写的存储桶上取得成功。	经过身份验证的请求所必需
success_action_redirect	上传成功后客户端重定向到的 URL。 如果未指定 <code>success_action_redirect</code>，或者 Amazon S3 无法解释 URL，则 Amazon S3 将返回在 <code>success_action_status</code> 字段中指定的空文档类型。 如果上传失败，Amazon S3 将返回错误，而不会将用户重定向到另一个 URL。	否

元素名称	说明	必填
success_action_status	在未指定 success_action_redirect 的情况下，成功上传后返回到客户端的状态代码。 有效值为 200、201 或 204（默认值）。 如果值设置为 200 或 204，Amazon S3 将返回一个空文档以及指定的状态代码。 如果值设置为 201，Amazon S3 将返回一个 XML 文档和一个 201 状态代码。有关 XML 文档内容的信息，请参阅 POST Object。 如果该值未设置或无效，Amazon S3 将返回一个空文档和一个 204 状态代码。  Note 某些版本的 Adobe Flash player 无法正确处理使用空白正文的 HTTP 响应。要通过 Adobe Flash 支持上传，建议您将 success_action_status 设置为 201。	否
x-amz-algorithm	用于对请求进行身份验证的签名算法。对于 AWS 签名版本 4，该值为 AWS4-HMAC-SHA256。 如果策略文档随请求一起提供，则此字段为必填字段。	经过身份验证的请求所必需

元素名称	说明	必填
x-amz-credential	除了访问密钥 ID 外，此字段还提供范围信息，用于标识签名对其有效的区域和服务。这应该与您在计算签名密钥以进行签名计算时使用的范围相同。 它是采用以下形式的字符串： <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> 例如： <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> 对于 Amazon S3，aws-service 字符串为 s3。有关 Amazon S3 aws-region 字符串的列表，请参阅《AWS 一般参考》中的区域和端点。如果策略文档随请求一起提供，则此元素为必需项。	经过身份验证的请求所必需
x-amz-date	它是采用 ISO8601 格式的日期值。例如 20130728T000000Z 。 它与您创建签名密钥时使用的日期相同（例如，20130728）。该值也必须与您在签名的策略 (x-amz-date) 中提供的值相同。 如果策略文档随请求一起提供，则此元素为必需项。	经过身份验证的请求所必需

元素名称	说明	必填
x-amz-security-token	Amazon DevPay 和会话凭证使用的安全令牌 如果请求正在使用 Amazon DevPay，它需要两个 x-amz-security-token 表单字段：一个用于产品令牌，另一个用于用户令牌。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用 DevPay。 如果请求正在使用会话凭证，它需要一个 x-amz-security-token 表单。有关更多信息，请参阅《IAM 用户指南》中的请求临时安全凭证。	否
x-amz-signature	(AWS 签名版本 4) 安全策略的 HMAC-SHA256 哈希值。 如果策略文档随请求一起提供，则此字段为必填字段。	经过身份验证的请求所必需
x-amz-meta-*	以前缀开头的字段名称是用户定义的元数据。每个元数据均以一组键-值对的形式存储和返回。Amazon S3 不验证或解释用户定义的元数据。有关更多信息，请参阅 PutObject。	否
x-amz-*	有关其它 x-amz-* 标头，请参阅 POST 对象 (POST Object) 。	否
file	文件或文本内容。 文件或内容必须是表单中的最后一个字段。 您一次仅能上传一个文件。	是

对于经过身份验证的请求，条件项是必需的；对于匿名请求，条件项是可选的。

现在您已知道如何创建表单，接下来可以创建您可以签名的安全策略。有关更多信息，请参阅 [POST 策略](#)。

POST 策略

主题

- [过期](#)
- [条件匹配](#)
- [条件](#)
- [字符转义](#)

使用 HTTP POST 发出经过身份验证的请求所需的策略是使用 JavaScript 对象表示法 (JSON) 编写的以 UTF-8 和 base64 编码的文档，该文档指定了请求必须满足的条件。根据您的设计策略文档的方式，您可以对每次上传、每个用户、所有上传或根据其它可满足您需要的设计来控制访问权限粒度。

本节介绍 POST 策略。有关使用 POST 策略计算签名的示例，请参阅 [示例：使用 HTTP POST 的基于浏览器的上传 \(使用 AWS 签名版本 4\)](#)。

Note

尽管策略文档是可选的，但我们强烈建议您使用策略文档来控制请求中支持的内容。如果您将存储桶设置为可公开写入，您根本无法控制哪些用户可以向您的存储桶写入内容。

下面是 POST 策略文档的示例。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

POST 策略始终包含 expiration 和 conditions 元素。示例策略使用两种条件匹配类型 (精确匹配和开头为匹配)。以下各节介绍了这些元素。

过期

`expiration` 元素以 ISO8601 GMT 日期格式指定 POST 策略的过期日期和时间。例如，`2013-08-01T12:00:00.000Z` 指定 POST 策略在 GMT 时间 2013 年 8 月 1 日午夜之后无效。

条件匹配

下表描述了可用于指定 POST 策略条件的条件匹配类型（将在下一节中介绍）。尽管您必须为您在表单中指定的每个表单字段指定至少一个条件，但您也可以通过为某个表单字段指定多个条件来创建更复杂的匹配条件。

条件匹配类型	说明
精确匹配	表单字段值必须与指定的值匹配。此示例指示 ACL 必须设置为公共读取： <pre>{"acl": "public-read" }</pre> 此示例是指示 ACL 必须设置为公共读取的替代方法： <pre>["eq", "\$acl", "public-read"]</pre>
Starts With	值必须以指定的值开头。本示例指示对象键必须以 <code>user/user1</code> 开头： <pre>["starts-with", "\$key", "user/user1/"]</pre>
与逗号分隔列表中的 Content-Type 匹配	包含逗号的 <code>starts-with</code> 条件的 Content-Type 值解释为列表。列表中的每个值都必须满足该条件，才能满足整个条件。例如，给定以下条件： <pre>["starts-with", "\$Content-Type", "image/"]</pre> 以下值将满足条件： <pre>"image/jpg,image/png,image/gif"</pre> 以下值将无法满足条件：

条件匹配类型	说明
	["image/jpg,text/plain"] Note 不管是否存在逗号，Content-Type 以外的数据元素都被视为字符串。
匹配任何内容	要配置 POST 策略以支持表单字段中的任何内容，请使用 starts-with 以及一个空值 ("")。此示例对于 success_action_redirect 支持任意值： ["starts-with", "\$success_action_redirect", ""]
指定范围	对于接受范围的表单字段，请使用逗号来分隔上限和下限。本示例支持 1 到 10 MiB 的文件大小： ["content-length-range", 1048576, 10485760]

[条件](#)中介绍了 POST 策略中支持的特定条件。

条件

POST 策略中的 conditions 是一个对象数组，其中的每个对象均用于验证请求。您可以使用这些条件来限制请求中支持的内容。例如，前面的策略条件要求符合以下各项：

- 请求必须指定 johnsmith 存储桶名称。
- 对象键名称必须有 user/eric 前缀。
- 对象 ACL 必须设置为 public-read。

您在表单中指定的每个表单字段 (x-amz-signature、file、policy 以及带有 x-ignore- 前缀的字段名称除外) 都必须出现在条件列表中。

 Note

在验证 POST 策略之前，表单中的所有变量都会扩展。因此，所有条件匹配都应该针对扩展的表单字段。假设您想将对象键名称限制为特定的前缀 (user/user1)。在这种情况下，您可以将键表单字段设置为 user/user1/\${filename}。您的 POST 策略应为 ["starts-with", "\$key", "user/user1/"] (请勿输入 ["starts-with", "\$key", "user/user1/\${filename}"])。有关更多信息，请参阅 [条件匹配](#)。

下表中介绍了策略文档条件。

元素名称	说明
acl	指定提交表单时必须使用的 ACL 值。 此条件支持下一节中讨论的精确匹配和 starts-with 条件匹配类型。
bucket	指定可接受的存储桶名称。 此条件支持精确匹配条件匹配类型。
content-length-range	已上传的内容支持的最小和最大大小。 此条件支持 content-length-range 条件匹配类型。
Cache-Control	特定于 REST 的标头。有关更多信息，请参阅 POST Object。 此条件支持精确匹配和 starts-with 条件匹配类型。
Content-Type	
Content-Disposition	
Content-Encoding	
Expires	
key	

元素名称	说明
	此条件支持精确匹配和 starts-with 条件匹配类型。
success_action_redirect redirect	上传成功后客户端重定向到的 URL。 此条件支持精确匹配和 starts-with 条件匹配类型。
success_action_status	在未指定 success_action_redirect 的情况下，成功上传后返回到客户端的状态代码。 此条件支持精确匹配。
x-amz-algorithm	计算签名时必须使用的签名算法。对于 AWS 签名版本 4，该值为 AWS4-HMAC-SHA256 。 此条件支持精确匹配。
x-amz-credential	用于计算签名的凭证。它提供访问密钥 ID 和范围信息，用于标识签名对其有效的区域和服务。这应该与您在计算签名密钥以进行签名计算时使用的范围相同。 它是采用以下形式的字符串： <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> 例如： AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request 对于 Amazon S3，aws-service 字符串为 s3。有关 Amazon S3 aws-region 字符串的列表，请参阅《AWS 一般参考》中的 区域和端点。如果 POST 策略文档随请求一起提供，则此元素为必需项。 此条件支持精确匹配。

元素名称	说明
x-amz-date	以 ISO8601 格式字符串指定的日期值。例如 20130728T000000Z 。该日期必须与您在创建用于计算签名的签名密钥时使用的日期相同。 如果 POST 策略文档随请求一起提供，则此元素为必需项。 此条件支持精确匹配。
x-amz-security-token	Amazon DevPay 安全令牌。 使用 Amazon DevPay 的每个请求都需要两个 x-amz-security-token 表单字段：一个用于产品令牌，一个用于用户令牌。因此，必须使用逗号来分隔值。例如，如果用户令牌是 eW91dHVIZQ== ，产品令牌是 b0hnNVNKNWJIQTA= ，您可以将 POST 策略条目设置为：{ "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKNWJIQTA=" } 。 有关 Amazon DevPay 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用 DevPay。
x-amz-meta-*	特定于用户的元数据。 此条件支持精确匹配和 starts-with 条件匹配类型。
x-amz-*	有关其它 x-amz-* 标头，请参阅 POST 对象 (POST Object) 。 此条件支持精确匹配。

 Note

如果您的工具包添加了更多表单字段（例如，Flash 添加了 filename），您必须将它们添加到 POST 策略文档。如果您可以控制此功能，将 x-ignore- 添加为字段的前缀以使 Amazon S3 忽略此功能并使其不影响此功能的未来版本。

字符转义

下表介绍了 POST 策略文档中必须进行转义的字符。

转义序列	说明
\\	反斜杠
\\\$	美元符号
\\b	退格键
\\f	换页
\\n	新建行
\\r	回车
\\t	水平选项卡
\\v	垂直选项卡
\\uXXXX	所有 Unicode 字符

现在，您已经熟悉了表单和策略，并了解了签名的工作原理，可以尝试使用 POST 上传示例。您需要编写代码来计算签名。该示例提供了一个示例表单和一个可用于测试签名计算的 POST 策略。有关更多信息，请参阅 [示例：使用 HTTP POST 的基于浏览器的上传（使用 AWS 签名版本 4）](#)。

示例：使用 HTTP POST 的基于浏览器的上传（使用 AWS 签名版本 4）

本节展示了使用 HTTP POST 请求将内容直接上传到 Amazon S3 的示例。

有关签名版本 4 的更多信息，请参阅[签名版本 4 签名流程](#)。

使用 HTTP POST 将文件上传到 Amazon S3

此示例提供示例 POST 策略和可用于上传文件的表单。本主题使用示例策略和虚拟凭证来向您介绍工作流程以及生成的签名和策略哈希值。您可以将此数据用作测试套件来验证签名计算代码。

该示例使用以下示例凭证进行签名计算。您可以使用这些凭证来验证签名计算代码。但是，在向 AWS 发送请求时，您必须将这些凭证替换为自己的凭证。

参数	值
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

策略和表单示例

以下 POST 策略支持在特定条件下上传到 Amazon S3。

```
{ "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" }
]
```

此 POST 策略针对请求设置了以下条件：

- 上传必须在 UTC 时间 2015 年 12 月 30 日中午之前进行。
- 内容只能上传到 sigv4examplebucket。存储桶必须位于您在凭证范围 (x-amz-credential 表单参数) 中指定的区域，因为您提供的签名仅在此范围内有效。
- 您可以提供任何以 user/user1 开头的键名称。例如 user/user1/MyPhoto.jpg。
- ACL 必须设置为 public-read。
- 如果上传成功，则用户的浏览器将重定向到 `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html`。
- 对象必须是一个映像文件。
- x-amz-meta-uuid 标签必须设置为 14365123651274。
- x-amz-meta-tag 可以包含任意值。

下面是此 POST 策略的 Base64 编码版本。在签名计算中，您可以使用此值作为 StringToSign。

```
eyJhZiZxhwaXJhdGlvbiI6IClIyMDE1LTEyLTMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

当您复制/粘贴上述策略时，它应包含回车符和换行符，以便您计算的哈希值与此值匹配（即带有 CRLF 行终止符的 ASCII 文本）。

使用示例凭证创建签名，签名值如下所示 [在签名计算中，日期与策略中的 x-amz-date 日期相同 (20151229)]：

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

以下示例表单指定了上述 POST 策略，并支持向 sigv4examplebucket 发出 POST 请求。将内容复制/粘贴到文本编辑器中，然后将其另存为 exampleform.html。然后，您可以使用 exampleform.html 将映像文件上传到特定的存储桶。如果您提供的签名与 Amazon S3 计算出的签名相匹配，您的请求将会成功。

Note

您必须使用有效的值更新存储桶名称、日期、凭证、策略和签名，以便将其成功上传到 S3。

```
<html>
  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

  </head>
  <body>

    <form action="http://sigv4examplebucket.s3.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload:
      <input type="input" name="key" value="user/user1/${filename}" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="http://
sigv4examplebucket.s3.amazonaws.com/successful_upload.html" />
      Content-Type:
      <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      <input type="hidden" name="x-amz-server-side-encryption" value="AES256" />
      <input type="text" name="X-Amz-Credential" value="AKIAIOSFODNN7EXAMPLE/20151229/
us-east-1/s3/aws4_request" />
      <input type="text" name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
      <input type="text" name="X-Amz-Date" value="20151229T000000Z" />

      Tags for File:
      <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="Policy" value='<Base64-encoded policy string>' />
      <input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
      File:
      <input type="file" name="file" /> <br />
      <!-- The elements after this will be ignored -->
      <input type="submit" name="submit" value="Upload to Amazon S3" />
    </form>

  </html>
```

POST 参数不区分大小写。举例来说，可以指定 `x-amz-signature` 或 `X-Amz-Signature`。

使用 AWS Amplify 库通过基于浏览器的上传来上传到 Amazon S3

本节介绍如何使用 AWS Amplify JavaScript 库将文件上传到 Amazon S3。

有关设置 AWS Amplify 库的信息，请参阅 [AWS Amplify 安装和配置](#)。

使用 AWS Amplify JavaScript 库将文件上传到 Amazon S3

AWS Amplify 库 Storage 模块提供了一种基于浏览器的简单上传机制，用于管理公有或私有 Amazon S3 存储中的用户内容。

Example : AWS Amplify 手动设置

以下示例显示了有关使用 AWS Amplify 存储模块的手动设置。存储模块的默认实现使用 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 : 将数据放入 Amazon S3

以下示例显示了如何将公有数据 put 到 Amazon S3 中。

```
Storage.put('test.txt', 'Hello')
  .then(result => console.log(result))
  .catch(err => console.log(err));
```

以下示例显示了如何将私有数据 put 到 Amazon S3 中。

```
Storage.put('test.txt', 'Private Content', {
```



```
    level: 'private',  
    contentType: 'text/plain'  
  })  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

有关使用 AWS Amplify 存储模块的更多信息，请参阅 [AWS Amplify Storage](#)。

更多信息

[AWS Amplify 快速入门](#)

通用请求标头

下表描述了各种类型的 Amazon S3 REST 请求可以使用的标头。

标头名称	说明
Authorization	请求身份验证所需的信息。有关更多信息，请转至《Amazon Simple Storage Service 开发人员指南》中的 身份验证标头 。对于匿名请求，不需要此标头。
Access-Control-Request-Method	作为预检 CORS 请求发送的 HTTP 方法的列表。如果预检 CORS 评估取得成功，则支持在以下 CORS 请求中使用指定的方法。
Content-Length	符合 RFC 2616 的消息的长度（不带标头）。此标头是加载 XML 的 PUT 和操作（例如日志记录和 ACL）所必需的。
Content-Type	如果请求在正文中有内容，则为该资源的内容类型。示例：text/plain
Content-MD5	根据 RFC 1864，消息（不含标头）的 base64 编码的 128 位 MD5 摘要。此标头可用作消息完整性检查，以验证数据是否与最初发送的数据相同。尽管它是可选的，但我们建议使用 Content-MD5 机制作为端到端完整性检查。有关 REST 请求身份验证的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的 REST 身份验证 。
Date	可以用于创建 Authorization 标头中包含的签名的日期。如果要使用 Date 标头用于签名，则必须使用 ISO 8601 基本格式来指定它。在这种情况下，不需要 x-amz-date 标头。请注意，存在 x-amz-date 时，它始终会覆盖 Date 标头的值。 如果 Date 标头不用于签名，则可以为RFC 2616 第 3.3 部分指定的完整日期格式之一。例如，日期/时间 Wed, 01 Mar 2006 12:00:00 GMT 是用于 Amazon S3 的有效日期/时间标头。

标头名称	说明
	如果您要将 Date 标头用于签名，则它必须使用 ISO 8601 基本 YYYYMMDD'T'HHMMSS'Z' 格式。 如果指定了 Date，但它没有使用 ISO 8601 基本格式，则您还必须包括 x-amz-date 标头。如果使用 ISO 8601 基本格式指定了 Date，则它足够用于对请求签名，您无需 x-amz-date 标头。有关更多信息，请参阅《Amazon Web 服务词汇表》中的处理签名版本 4 中的日期。
Expect	当应用程序使用 100-Continue 时，它不会发送请求正文，直到收到确认。如果根据标头拒绝了消息，则不会发送消息的正文。只有在发送正文时才能使用此标头。 有效值：100-continue
Host	对于路径类型请求，该值为 s3.amazonaws.com 。对于虚拟类型请求，该值为 BucketName.s3.amazonaws.com 。有关更多信息，请转至《Amazon Simple Storage Service 用户指南》中的虚拟托管。 此标头对于 HTTP 1.1 是必需的（大多数工具包会自动添加此标头）；而对于 HTTP/1.0 请求则是可选的。
Origin	用于指定初始请求者的服务器名称的端点。
x-amz-content-sha256	使用签名版本 4 对请求进行身份验证时，此标头提供请求有效载荷的哈希值。有关更多信息，请参阅 Authorization 标头的签名计算：在单个分块中传输有效载荷（AWS 签名版本 4）。当分块上传对象时，您可以将值设置为 STREAMING-AWS4-HMAC-SHA256-PAYLOAD ，以表示签名仅涵盖标头且没有有效载荷。有关更多信息，请参阅 Authorization 标头的签名计算：在多个分块中传输有效载荷（分块上传）（AWS 签名版本 4）。

标头名称	说明
x-amz-date	用于在 Authorization 标头中创建签名的日期。格式必须为使用 YYYYMMDD'T'HHMMSS'Z' 格式的 ISO 8601 基本格式。例如，日期/时间 20170210T120000Z 是用于 Amazon S3 的有效 x-amz-date 。 x-amz-date 对所有请求都是可选的；它可以用于覆盖对请求进行签名所使用的日期。如果使用 ISO 8601 基本格式指定了 Date 标头，则无需 x-amz-date 。存在 x-amz-date 时，它始终会覆盖 Date 标头的值。有关更多信息，请参阅《Amazon Web 服务词汇表》中的处理签名版本 4 中的日期。
x-amz-security-token	此标头可以在以下场景中使用： - 为 Amazon DevPay 操作提供安全令牌：使用 Amazon DevPay 的每个请求都需要两个 x-amz-security-token 标头：一个用于产品令牌，一个用于用户令牌。当 Amazon S3 收到经身份验证的请求时，它会将计算出的签名与提供的签名进行比较。用于计算签名的多值标头格式不正确可能会导致身份验证问题。 - 在使用临时安全凭证时提供安全令牌：当使用从 IAM 获得的临时安全凭证发出请求时，必须使用此标头提供安全令牌。要了解有关临时安全凭证的更多信息，请参阅发出请求。 使用 Amazon DevPay 的请求和使用临时安全凭证进行签名的请求都需要此标头。

通用响应标头

下表描述了对于大多数 Amazon S3 响应通用的响应标头。

名称	描述
Access-Control-Allow-Credentials	一个布尔值，用于确定服务器是否支持 CORS 请求包含凭证。如果 Access-Control-Allow-Origin 请求标头设置为 '*'，则将忽略 Access-Control-Allow-Credentials 响应标头，否则在 CORS 评估取得成功时，它将设置为 true。 类型：布尔值 默认值：无
Access-Control-Allow-Headers	CORS 请求可以使用的 HTTP 标头列表。对于成功的 CORS 评估返回 Access-Control-Allow-Headers 响应标头，并显式指定所有支持的 Access-Control-Request-Headers。 类型：字符串 默认值：无
Access-Control-Allow-Methods	指定支持哪些 HTTP 方法的列表。只有在 CORS 评估取得成功后，Amazon S3 才会支持来自可让使用的 CORS 方法的 CORS 请求。 类型：字符串 默认值：无
Access-Control-Allow-Origin	支持的源的位置。只有在 CORS 评估取得成功后，Amazon S3 才会发送 Access-Control-Allow-Origin 响应标头。如果请求源与 CORS 配置支持的源中的 '*' 相匹配，则将在此响应标头而不是原始源中返回 '*'。 类型：字符串 默认值：无

名称	描述
Access-Control-Expose-Headers	一个列表，可让服务器识别响应标头，当 CORS 评估取得成功时，该标头会公开应用程序的访问权限。 类型：字符串 默认值：无
Access-Control-Max-Age	对于由资源、HTTP 方法和源识别的 CORS 预检请求，浏览器可以为该请求缓存响应的时间（以秒为单位）。仅当 CORS 评估取得成功时，才会返回 Access-Control-Max-Age 响应标头。 类型：整数 默认值：无
Vary	一个列表，用于指示 CORS 评估结果会因哪些请求标头而异。仅当 CORS 评估取得成功时，才会返回 Vary 响应标头。 类型：字符串 默认值：无
Content-Length	响应正文的长度（以字节为单位）。 类型：字符串 默认值：无
Content-Type	内容的 MIME 类型。例如 Content-Type: text/html; charset=utf-8。 类型：字符串 默认值：无

名称	描述
Connection	一个值，用于指定与服务器的连接是打开还是关闭。 类型：Enum 有效值：open close 默认值：无
Date	Amazon S3 响应的日期和时间；例如“Wed, 01 Mar 2006 12:00:00 GMT”。 类型：字符串 默认值：无
ETag	表示对象的特定版本的实体标签（ETag）。ETag 仅反映对对象的内容的更改，而不反映对对象的元数据的更改。ETag 可能是该对象数据的 MD5 摘要，也可能不是。是与不是取决于创建对象的方式及对象加密方式，具体如下： • 通过 AWS 管理控制台或由 PUT 对象、POST Object 或 Copy 操作创建的对象： • 对于属于纯文本或以具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）进行加密的对象，其 ETag 是其数据的 MD5 摘要。 • 对于以具有客户提供的密钥（SSE-C）或 AWS Key Management Service（AWS KMS）密钥（SSE-KMS）的服务器端加密进行加密的对象，其 ETag 不是其对象数据的 MD5 摘要。 • 对于通过分段上传或上传分段副本操作创建的对象，其 ETag 不是 MD5 摘要，而无论采用的是何种加密方法。 类型：字符串
Server	创建了响应的服务器的名称。 类型：字符串 默认：AmazonS3

名称	描述
x-amz-delete-marker	一个值，它指定返回的对象是 (true) 或不是 (false) 删除标记。 类型：布尔值 有效值：true false 默认：false
x-amz-id-2	与 x-amz-request-id 标头一起使用的特殊令牌，有助于 AWS 排查问题。有关使用这些请求 ID 的支持的信息，请参阅 Amazon S3 故障排除。 类型：字符串 默认值：无
x-amz-request-id	由 Amazon S3 创建的唯一标识请求的值。此值与 x-amz-id-2 标头一起使用，有助于 AWS 排查问题。有关使用这些请求 ID 的支持的信息，请参阅 Amazon S3 故障排除。 类型：字符串 默认值：无
x-amz-server-side-encryption	在 Amazon S3 中存储此对象时使用的服务器端加密算法（例如，AES256、aws:kms）。 有效值：AES256 aws:kms
x-amz-version-id	对象的版本。启用版本控制时，Amazon S3 会为添加到存储桶的对象生成一个随机数。该值采用 UTF-8 编码且支持 URL。当您在已暂停版本控制的存储桶中 PUT 对象时，版本 ID 始终为 null。 类型：字符串 有效值：null 任何支持 URL 的 UTF-8 编码字符串 默认值：null

错误响应

本节提供有关 Amazon S3 错误的参考信息。

Note

- 通常，S3 存储桶所有者需要为获得 HTTP 200 OK 成功响应和 HTTP 4XX 客户端错误响应的请求付费。存储桶所有者无需为 HTTP 5XX 服务器错误响应（例如 HTTP 503 Slow Down 错误）付费。有关不计费的 HTTP 3XX 和 4XX 状态代码下的 S3 错误代码的更多信息，请参阅《Amazon S3 用户指南》中的 [Amazon S3 错误响应的账单计费](#)。有关在存储桶配置为申请方付款存储桶时的账单费用的更多信息，请参阅《Amazon S3 用户指南》中的 [申请方付款的费用支付方式](#)。
- HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

主题

- [REST 错误响应](#)
- [错误代码列表](#)
- [SELECT 对象内容错误代码列表](#)
- [与复制相关的错误代码列表](#)
- [与标记相关的错误代码列表](#)
- [与 Amazon S3 注释相关的错误代码列表](#)
- [Amazon S3 on Outposts 错误代码列表](#)
- [Amazon S3 Storage Lens 存储统计管理工具错误代码列表](#)
- [Amazon S3 对象 Lambda 错误代码列表](#)
- [Amazon S3 异步错误代码列表](#)
- [Amazon S3 访问权限管控错误代码列表](#)
- [Amazon FSx 相关的错误代码列表](#)
- [Amazon S3 表类数据存储服务错误代码列表](#)
- [Amazon S3 排错最佳实践](#)

REST 错误响应

发生错误时，标头信息包含以下内容：

- Content-Type : application/xml
- 合适的 3xx、4xx 或 5xx HTTP 状态代码

响应的正文还包含有关错误的信息。下面的错误响应示例显示了所有 REST 错误响应常见的响应元素的结构。

```
<?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>
```

下表说明了 REST 错误响应元素。

名称	描述
Code	错误代码是用于唯一标识错误条件的字符串。这意味着按类型检测和处理错误的程序将读取和理解错误代码。有关更多信息，请参阅错误代码列表。 类型：字符串 原级：Error
Error	所有错误元素的容器。 类型：容器 原级：无
Message	错误消息包含错误条件的常规描述 (英语)。它主要面向用户受众。如果用户遇到不知如何处理或不愿处理的错误条件，简单的程序将直接向最终用户显示消息。支持更详尽的错误处理和适当国际化的复杂程序更容易忽略错误消息。 类型：字符串

名称	描述
	原级：Error
RequestId	与错误关联的请求的 ID。
	类型：字符串
	原级：Error
Resource	错误中涉及的存储桶或对象。
	类型：字符串
	原级：Error

许多错误响应包含额外的结构化数据，旨在供开发人员阅读和理解以诊断编程错误。例如，如果您使用与服务器上计算出的摘要不匹配的 REST PUT 请求发送 Content-MD5 标头，您将收到 BadDigest 错误。错误响应还包括服务器计算出的摘要作为详细元素，以及您指示服务器预期的摘要。在开发过程中，您可以使用此信息诊断错误。在生产过程中，运行良好的程序可能会将此信息包含在其错误日志中。

有关常规响应元素的信息，请转到[错误响应](#)。

错误代码列表

下表列出了 Amazon S3 错误代码。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
AccessControlListNotSupported	存储桶不支持 ACL。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
AccessDenied	拒绝访问	403 Forbidden	客户端
AccessPointAlreadyOwnedByYou	您的账户中已存在同名的接入点。	409 Conflict	客户端
AccountProblem	您的 AWS 账户存在问题，导致无法成功完成操作。如需更多帮助，请参阅 联系我们 。	403 Forbidden	客户端
AllAccessDisabled	对此 Amazon S3 资源的所有访问权限均已被禁用。如需更多帮助，请参阅 联系我们 。	403 Forbidden	客户端
AmbiguousGrantByEmailAddress	您提供的电子邮件地址与多个账户相关联。	400 Bad Request	客户端
AuthorizationHeaderMalformed	您提供的授权标头无效。	400 Bad Request	不适用
AuthorizationQueryParametersError	您提供的授权查询参数无效。	400 Bad Request	不适用
BadDigest	您指定的 Content-MD5 或校验和值与服务器收到的内容不匹配。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
BucketAlreadyExists	请求的存储桶名称不可用。存储桶命名空间由系统的所有用户共享。请指定其它名称，然后重试。	409 Conflict	客户端
BucketAlreadyOwnedByYou	您尝试创建的存储桶已经存在，并且您拥有该存储桶。Amazon S3 会在所有 AWS 区域中返回此错误，但美国东部（弗吉尼亚州北部）（us-east-1）区域除外。为了与旧版兼容，如果您重新创建您已在 us-east-1 中拥有的现有存储桶，Amazon S3 会返回 200 OK 并重置存储桶访问控制列表（ACL）。 对于 Amazon S3 on Outposts，您尝试创建的存储桶已存在于您的 Outpost 中，并且您拥有该存储桶。	409 Conflict 除 us-east-1 以外的所有区域中）	客户端
BucketHasAccessPointsAttached	您尝试删除的存储桶附加了接入点。在删除存储桶之前，请先删除接入点。	400 Bad Request	客户端
BucketNotEmpty	您尝试删除的存储桶不为空。	409 Conflict	客户端
ClientTokenConflict	您的多区域接入点幂等性令牌已用于不同的请求。	409 Conflict	客户端
ConnectionClosedByRequester	如果在读取 WriteGetObjectResponse 正文时遇到错误，则返回给原始调用方。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
ConditionalRequestConflict	发生了冲突的操作。如果使用 PutObject，则可以重试请求。如果使用分段上传，则应发起另一个 CreateMultipartUpload 请求并重新上传每个分段。	409 Conflict	客户端
CredentialsNotSupported	此请求不支持凭证。	400 Bad Request	客户端
CrossLocationLoggingProhibited	不支持跨区域日志记录。一个 AWS 区域中的存储桶无法将信息记录到另一个区域中的存储桶。	403 Forbidden	客户端
DeviceNotActiveError	设备当前处于不活动状态。	400 Bad Request	客户端
EndpointNotFound	将请求定向到正确的端点。	400 Bad Request	客户端
EntityTooSmall	您建议的上传小于支持的最小对象大小。	400 Bad Request	客户端
EntityTooLarge	您建议的上传超过了支持的最大对象大小。有关更多信息，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service 端点和配额 。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
	您建议的下载超过了支持的最大大小。	405 Method Not Allowed	客户端
ExpiredToken	提供的令牌过期。	400 Bad Request	客户端
IllegalLocationConstraintException	出现此错误可能的原因如下： - 您正在尝试从存储桶所在区域以外的区域访问存储桶。 - 您尝试创建具有位置约束的存储桶，该存储桶对应的区域与请求发送到的区域端点不同。	400 Bad Request	客户端
IllegalVersioningConfigurationException	在请求中指定的版本控制配置无效。	400 Bad Request	客户端
IncompleteBody	您没有提供由 Content-Length HTTP 标头指定的字节数。	400 Bad Request	客户端
IncorrectEndpoint	指定的存储桶存在于其它区域中。将请求定向到正确的端点。	400 Bad Request	客户端
IncorrectNumberOfFilesInPostRequest	POST 要求每个请求确切上传一个文件。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InlineDataTooLarge	内联数据超出支持的最大大小。	400 Bad Request	客户端
InternalError	出现内部错误。请重试。	500 Internal Server Error	服务器
InvalidAccessKeyId	您提供的 AWS 访问密钥 ID 在记录中不存在。	403 Forbidden	客户端
InvalidAccessPoint	指定的接入点名称或账户无效。	400 Bad Request	客户端
InvalidAccessPointAliasError	指定的接入点别名无效。	400 Bad Request	客户端
InvalidAddressingHeader	必须指定匿名角色。	不适用	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidArgument	出现此错误可能的原因如下： - 向与在 bucket-region 参数中指定的区域不同的区域端点发出 ListBuckets 请求。 - 指定的参数无效。 - 请求缺少必需的标头。 - 指定的参数不完整或格式有误。 - 指定的参数长度必须大于或等于 3。	400 Bad Request	客户端
InvalidBucketAclWithObjectOwnership	存储桶无法使用 ObjectOwnership 的 BucketOwnerEnforced 设置来设置 ACL。	400 Bad Request	客户端
InvalidBucketName	指定的存储桶无效。	400 Bad Request	客户端
InvalidBucketOwnerAWSAccountID	预期的存储桶所有者参数的值必须为 AWS 账户 ID。	400 Bad Request	客户端
InvalidBucketState	请求对于存储桶的当前状态无效。	409 Conflict	客户端
InvalidDigest	指定的 Content-MD5 或校验和值无效。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidEncryptionAlgorithmError	指定的加密请求无效。有效值为 AES256。	400 Bad Request	客户端
InvalidHostHeader	请求中提供的主机标头使用了不正确的样式寻址。	400 Bad Request	客户端
InvalidHttpMethod	请求是使用意外的 HTTP 方法发出的。	400 Bad Request	客户端
InvalidLocationConstraint	指定的位置（区域）约束无效。有关为存储桶选择区域的更多信息，请参阅 存储桶概述 。	400 Bad Request	客户端
InvalidObjectState	操作对于对象的当前状态无效。	403 Forbidden	客户端
InvalidPart	未找到一个或多个指定的分段。该分段可能尚未上传，或者指定的实体标签可能与该分段的实体标签不匹配。	400 Bad Request	客户端
InvalidPartOrder	分段列表未按升序排列。必须按分段编号依次指定分段列表。	400 Bad Request	客户端
InvalidPayer	已禁用针对此对象的所有访问权限。如需更多帮助，请参阅 联系我们 。	403 Forbidden	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidPolicyDocument	表单的内容不符合在策略文档中指定的条件。	400 Bad Request	客户端
InvalidRange	无法满足所请求的范围。	416 请求的范围无法满足	客户端
InvalidRegion	您已尝试在未选择加入的区域中创建多区域接入点。	403 禁止访问	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidRequest	出现此错误可能的原因如下： • 未分页的 ListBuckets 请求是从已批准的通用存储桶配额高于 10000 的账户发出的。您必须发出分页请求，才能列出拥有超过 10000 个存储桶的账户中的存储桶。 • 请求使用了错误的签名版本。请使用 AWS4-HMAC-SHA256 （签名版本 4）。 • 只能为现有存储桶创建接入点。 • 接入点未处于可以删除的状态。 • 只能列出现有存储桶的接入点。 • 下一个令牌无效。 • 生命周期规则中必须至少指定一个操作。 • 必须至少指定一个生命周期规则。 • 生命周期规则的数量不得超过所允许的 1000 条规则的限制。 • MaxResults 参数的范围无效。 •	400 Bad Request	客户 端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
	SOAP 请求必须通过 HTTPS 连接发出。 • 名称不符合 DNS 的存储桶不支持 Amazon S3 Transfer Acceleration。 • 名称中包含点 (.) 的存储桶不支持 Amazon S3 Transfer Acceleration。 • Amazon S3 Transfer Acceleration 端点仅支持虚拟样式的请求。 • 此存储桶上未配置 Amazon S3 Transfer Acceleration。 • 此存储桶上已禁用 Amazon S3 Transfer Acceleration。 • 该存储桶上不支持 Amazon S3 Transfer Acceleration。如需帮助，请联系 支持。 • 此存储桶上不能启用 Amazon S3 Transfer Acceleration。如需帮助，请联系 支持。 • HTTP 标头和查询参数中提供的值相互冲突。 •		

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
	HTTP 标头和 POST 表单字段中提供的值相互冲突。 在大于 5GB 的对象上发出 CopyObject 请求。		
InvalidSessionException	如果会话因超时或过期而不再存在，则返回。	400 Bad Request	客户端
InvalidSignature	服务器计算出的请求签名与您提供的签名不匹配。请检查您的 AWS 秘密访问密钥和签名方法。有关更多信息，请参阅 签署和对 REST 请求进行身份验证 。	400 Bad Request	客户端
InvalidSecurity	提供的安全凭证无效。	403 Forbidden	客户端
InvalidSOAPRequest	SOAP 请求正文无效。	400 Bad Request	客户端
InvalidStorageClass	指定的存储类无效。	400 Bad Request	客户端
InvalidTargetBucketForLogging	用于日志记录的目标存储桶不存在、不归您拥有，或者没有针对日志传送组的相应授权。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidToken	提供的令牌格式不正确或者无效。	400 Bad Request	客户端
InvalidURI	无法解析指定的 URI。	400 Bad Request	客户端
KeyTooLongError	键过长。	400 Bad Request	客户端
KMS.DisabledException	由于指定的 KMS 密钥未启用，请求被拒绝。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
KMS.InvalidKeyUsageException	由于以下原因之一，请求被拒绝： • KMS 密钥的 KeyUsage 值与 API 操作不兼容。 • 为操作指定的加密算法或签名算法与 KMS 密钥 (KeySpec) 中的密钥材料类型不兼容。 要加密、解密、重新加密和生成数据密钥，KeyUsage 必须为 ENCRYPT_DECRYPT。要对消息进行签名和验证，KeyUsage 必须为 SIGN_VERIFY。要生成和验证消息身份验证代码 (MAC)，KeyUsage 必须为 GENERATE_VERIFY_MAC。要获取密钥协议密钥，KeyUsage 必须为 KEY_AGREEMENT。要查找 KMS 密钥的 KeyUsage，请使用 DescribeKey 操作。 要查找特定 KMS 密钥支持的加密或签名算法，请使用 DescribeKey 操作。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
KMS.KMSInvalidStateException	由于指定资源的状态对此请求无效，请求被拒绝。此异常意味着以下情况之一： • KMS 密钥的密钥状态与操作不兼容。 要查找密钥状态，请使用 DescribeKey 操作。有关哪些密钥状态与每个 KMS 操作兼容的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 Key states of AWS KMS keys。 • 对于自定义密钥存储中 KMS 密钥的加密操作，此异常表示存在一般性故障，原因可能有很多。要确定原因，请参阅随异常显示的错误消息。	400 Bad Request	客户端
KMS.NotFoundException	由于找不到指定的实体或资源，请求被拒绝。	400 Bad Request	客户端
MalformedACLError	您提供的 ACL 格式有误，或者没有根据我们发布的架构进行验证。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
MalformedPOSTRequest	您的 POST 请求正文的分段/表单数据格式不正确。	400 Bad Request	客户端
MalformedXML	您提供的 XML 格式不正确，或者没有根据我们发布的架构进行验证。	400 Bad Request	客户端
MaxMessageLengthExceeded	您的请求过大。	400 Bad Request	客户端
MaxPostPreDataLengthExceededError	上传文件前的 POST 请求字段太大。	400 Bad Request	客户端
MetadataTooLarge	您的元数据标头超过了允许的最大元数据大小。	400 Bad Request	客户端
MethodNotAllowed	不允许对此资源使用指定的方法。	405 Method Not Allowed	客户端
MissingAttachment	预计会有 SOAP 附件，但未找到任何附件。	400 错误请求	客户端
MissingAuthenticationToken	未对请求进行签名。	403 禁止访问	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
MissingContentLength	您必须提供 Content-Length HTTP 标头。	411 Length Required	客户端
MissingRequestBodyError	您发送了一个空的 XML 文档作为请求。	400 Bad Request	客户端
MissingSecurityElement	SOAP 1.1 请求缺少安全元素。	400 Bad Request	客户端
MissingSecurityHeader	请求缺少必需的标头。	400 Bad Request	客户端
NoLoggingStatusForKey	不存在针对某个键的日志记录状态子资源之类的内容。	400 Bad Request	客户端
NoSuchAsyncRequest	找不到指定的请求。	404 Not Found	客户端
NoSuchBucket	指定的存储桶不存在。	404 Not Found	客户端
NoSuchBucketPolicy	指定的存储桶没有存储桶策略。	404 Not Found	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
NoSuchCORSConfiguration	指定的存储桶没有 CORS 配置。	404 Not Found	客户端
NoSuchKey	指定的键不存在。	404 Not Found	客户端
NoSuchLifecycleConfiguration	指定的生命周期配置不存在。	404 Not Found	客户端
NoSuchMultiRegionAccessPoint	指定的多区域接入点不存在。	404 Not Found	客户端
NoSuchObjectLockConfiguration	指定的对象没有 ObjectLock 配置。	404 Not Found	客户端
NoSuchWebsiteConfiguration	指定的存储桶没有网站配置。	404 Not Found	客户端
NoSuchTagSet	指定的标签不存在。	404 Not Found	客户端
NoSuchUpload	指定的分段上传不存在。上传 ID 可能无效，或者分段上传可能已中止或已完成。	404 Not Found	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
NoSuchVersion	在请求中指定的版本 ID 与现有版本不匹配。	404 Not Found	客户端
NotDeviceOwnerError	生成令牌的设备不归经过身份验证的用户拥有。	400 Bad Request	客户端
NotImplemented	您提供的标头意味着未实现的功能。	501 Not Implemented	服务器
NotModified	未更改资源。	304 Not Modified	服务器
NoTransformationDefined	找不到此对象 Lambda 接入点的转换。	404 Not Found	客户端
NotSignedUp	您的账户未注册 Amazon S3 服务。您必须先注册，然后才能使用 Amazon S3。您可以通过以下 URL 注册： https://aws.amazon.com/s3	403 Forbidden	客户端
ObjectLockConfigurationNotFoundError	此存储桶不存在对象锁定配置。	404 Not Found	客户端
OwnershipControlsNotFoundError	未找到存储桶所有权控制。	404 Not Found	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
OperationAborted	当前正在对此资源执行冲突的操作。请重试。	409 Conflict	客户端
PermanentRedirect	必须使用指定的端点来寻址您正在尝试访问的存储桶。请将未来的所有请求发送到此端点。	301 Moved Permanently	客户端
PermanentRedirectControlError	必须使用指定的端点来寻址您正在尝试访问的 API 操作。请将未来的所有请求发送到此端点。	301 Moved Permanently	客户端
PreconditionFailed	至少有一个您指定的先决条件不成立。	412 Precondition Failed	客户端
Redirect	临时重定向。域名系统 (DNS) 服务器更新时您会被重定向至存储桶。	307 Temporary Redirect	客户端
RequestHeaderSectionTooLarge	用于发出请求的请求标头和查询参数超出支持的最大大小。	400 Bad Request	客户端
RequestIsNotMultipartContent	存储桶 POST 请求必须是附件类型的分段/表单数据。	412 Precondition Failed	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
RequestTimeout	在超时时段内，未读取或写入您与服务器的套接字连接。	400 Bad Request	客户端
RequestTimeTooSkewed	请求时间和服务器时间之间的差异太大。	403 Forbidden	客户端
RequestTorrentOfBucketError	不允许请求存储桶的 torrent 文件。	400 Bad Request	客户端
ResponseInterrupted	如果在读取 WriteGetObjectResponse 正文时遇到错误，则返回给原始调用方。	400 Bad Request	客户端
RestoreAlreadyInProgress	对象还原已在进行中。	409 Conflict	客户端
ServerSideEncryptionConfigurationNotFound	未找到服务器端加密配置。	400 Bad Request	客户端
ServiceUnavailable	服务无法处理请求。	503 Service Unavailable	服务器
SignatureDoesNotMatch	服务器计算出的请求签名与您提供的签名不匹配。请检查您的 AWS 秘密访问密钥和签名方法。有关更多信息，请参阅 REST 身份验证 和 SOAP 身份验证 。	403 Forbidden	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
SlowDown	请降低请求速率。	503 Slow Down	服务器
503 SlowDown	降低	503 Slow Down	服务器
TagPolicyException	标签策略对于以下标签键不支持指定的值。	400 Bad Request	客户端
TemporaryRedirect	域名系统 (DNS) 服务器更新时您会被重定向至存储桶。	307 Temporary Redirect	客户端
TokenCodeInvalidError	您提供的序列号和/或令牌代码无效。	400 Bad Request	客户端
TokenRefreshRequired	必须刷新提供的令牌。	400 Bad Request	客户端
TooManyAccessPoints	您尝试创建的接入点数量超过了账户支持的数量。有关更多信息，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service 端点和配额 。	400 Bad Request	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
TooManyBuckets	您尝试创建的存储桶数量超过了账户支持的数量。有关更多信息，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service 端点和配额 。	400 Bad Request	客户端
TooManyMultiRegionAccessPointregionsError	您尝试创建的多区域接入点拥有的区域数超过了账户支持的区域数。有关更多信息，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service 端点和配额 。	400 Bad Request	客户端
TooManyMultiRegionAccessPoints	您尝试创建的多区域接入点数量超过了账户支持的数量。有关更多信息，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service 端点和配额 。	400 Bad Request	客户端
UnauthorizedAccessError	仅适用于中国区域。在向没有 ICP 许可证的存储桶发出请求时返回。有关更多信息，请参阅 ICP Recordal 。	403 Forbidden	客户端
UnexpectedContent	此请求包含不支持的内容。	400 Bad Request	客户端
UnexpectedIPError	仅适用于中国区域。由于 IP 不符合预期，请求被拒绝。	403 Forbidden	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
UnsupportedArgument	该请求包含不支持的参数。	400 Bad Request	客户端
UnsupportedSignature	所提供的请求使用不受支持的 STS 令牌版本进行签名，或者签名版本不受支持。	400 Bad Request	客户端
UnresolvableGrantByEmailAddress	您提供的电子邮件地址与记录中的任何账户都不匹配。	400 Bad Request	客户端
UserKeyMustBeSpecified	存储桶 POST 请求必须包含指定的字段名称。如果已指定，请检查字段的顺序。	400 Bad Request	客户端
NoSuchAccessPoint	指定的接入点不存在。	404 Not Found	客户端
InvalidTag	您的请求包含无效的标签输入。例如，您的请求可能包含重复的键、过长的键或值，或包含系统标签。	400 Bad Request	客户端
MalformedPolicy	您的策略包含无效的主体。	400 Bad Request	客户端

SELECT 对象内容错误代码列表

 Important

不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。[了解详情](#)

下表包含 SELECT Object Content 可能返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
AmbiguousFieldName	字段名称与文件中的多个字段匹配。检查 SQL 表达式和文件，然后重试。	400	客户端
Busy	该服务不可用。请稍后重试。	503	客户端
CastFailed	尝试在 SQL 表达式中使用 CAST 将一种数据类型转换为另一种数据类型失败。	400	客户端
ColumnTooLong	结果中某列的长度大于 1 MB 的 maxCharsPerColumn 。	400	客户端
CSVEscapingRecordDelimiter	在文件中找到了带引号的记录分隔符。要支持带引号的记录分隔符，请将 AllowQuotedRecordDelimiter 设置为 'TRUE'。	400	客户端
CSVParsingError	解析 CSV 文件时出错。请检查文件，然后重试。	400	客户端
CSVUnescapedQuote	解析 CSV 文件时发现未转义的引号。要支持带引号的记录分隔符，	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
	请将 AllowQuotedRecordDelimiter 设置为 'TRUE'。		
EmptyRequestBody	请求正文不能为空。	400	客户端
EvaluatorBindingDoesNotExist	SQL 表达式中不存在所提供的列名或路径。	400	客户端
EvaluatorInvalidArguments	SQL 表达式的函数调用中的参数数量不正确。	400	客户端
EvaluatorInvalidTimestampFormatPattern	SQL 表达式中的时间戳格式字符串无效。	400	客户端
EvaluatorInvalidTimestampFormatPatternSymbol	时间戳格式模式在 SQL 表达式中包含一个无效的符号。	400	客户端
EvaluatorInvalidTimestampFormatPatternSymbolForParsing	时间戳格式模式包含有效的格式符号，但该符号不能应用于 SQL 表达式中的时间戳解析。	400	客户端
EvaluatorInvalidTimestampFormatPatternToken	时间戳格式模式在 SQL 表达式中包含一个无效的令牌。	400	客户端
EvaluatorLikePatternInvalidEscapeSequence	向 LIKE 表达式提供的参数无效。	400	客户端
EvaluatorNegativeLimit	LIMIT 不得为负值。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
EvaluatorTimestampFormatPatternDuplicateFields	时间戳格式模式包含多个格式说明符来表示 SQL 表达式中的时间戳字段。	400	客户端
EvaluatorTimestampFormatPatternHourClockAmPmMismatch	时间戳格式模式包含一个 12 小时制格式符号，但不包含 AM/PM 字段，或者它包含一个 24 小时制格式说明符，并在 SQL 表达式中包含 AM/PM 字段。	400	客户端
EvaluatorUnterminatedTimestampFormatPatternToken	时间戳格式模式在 SQL 表达式中包含未终止的令牌。	400	客户端
ExpressionTooLong	SQL 表达式过长。SQL 表达式的最大字节长度为 256 KB。	400	客户端
ExternalEvalException	无法对查询进行评估。请检查文件，然后重试。	400	客户端
IllegalSqlFunctionArgument	SQL 函数中使用了非法参数。	400	客户端
IncorrectSqlFunctionArgumentType	在 SQL 表达式的函数调用中指定了不正确的参数类型。	400	客户端
IntegerOverflow	SQL 表达式中出现了整数溢出或下溢现象。	400	客户端
InternalError	出现内部错误。	500	客户端
InvalidCast	尝试在 SQL 表达式中使用 CAST 将一种数据类型转换为另一种数据类型失败。	400	客户端
InvalidColumnIndex	SQL 表达式中的列索引无效。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidCompressionFormat	该文件未采用支持的压缩格式。仅支持 GZIP 和 BZIP2。	400	客户端
InvalidDataSource	数据来源类型无效。仅支持 CSV、JSON 和 Parquet。	400	客户端
InvalidDataType	SQL 表达式包含无效的数据类型。	400	客户端
InvalidExpressionType	ExpressionType 值无效。仅支持 SQL 表达式。	400	客户端
InvalidFileHeaderInfo	FileHeaderInfo 值无效。仅支持 NONE、USE 和 IGNORE。	400	客户端
InvalidJsonType	JsonType 值无效。仅支持 DOCUMENT 和 LINES。	400	客户端
InvalidKeyPath	SQL 表达式中的键路径无效。	400	客户端
InvalidQuoteFields	QuoteFields 值无效。仅支持 ALWAYS 和 ASNEEDED。	400	客户端
InvalidRequestParameter	SelectRequest 元素中参数的值无效。请查看服务 API 文档，然后重试。	400	客户端
InvalidScanRange	提供的扫描范围无效。	400	客户端
InvalidTableAlias	SQL 表达式包含无效的表别名。	400	客户端
InvalidTextEncoding	编码类型无效。只支持 UTF-8 编码。	400	客户端
JSONParsingError	解析 JSON 文件时出错。请检查文件，然后重试。	400	客户端
LexerInvalidChar	SQL 表达式包含无效的字符。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
LexerInvalidIONLiteral	SQL 表达式包含无效的运算符。	400	客户端
LexerInvalidLiteral	SQL 表达式包含无效的运算符。	400	客户端
LexerInvalidOperator	SQL 表达式包含无效的文本。	400	客户端
LikeInvalidInputs	为 SQL 表达式中的 LIKE 子句提供的参数无效。	400	客户端
MalformedXML	提供的 XML 格式不正确，或者没有根据我们发布的架构进行验证。请查看服务文档，然后重试。	400	客户端
MaxOperatorsExceeded	无法解析 SQL 表达式，请尝试降低复杂性。例如，减少使用的运算符数量。	400	客户端
MethodNotAllowed	不允许对此资源使用指定的方法。	405 Method Not Allowed	客户端
MissingRequiredParameter	SelectRequest 实体缺少必需的参数。请查看服务文档，然后重试。	400	客户端
MultipleDataSourcesUnsupported	不支持多个数据来源。	400	客户端
NumberFormatError	解析数字时出错。此错误可能由整数下溢或溢出造成。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
ObjectSerializationConflict	InputSerialization 指定多种格式 (CSV、JSON 或 Parquet)，或者 OutputSerialization 指定多种格式 (CSV 或 JSON)。对于 InputSerialization 和 OutputSerialization，只能为每个分别指定一种格式。	400	客户端
OverMaxColumn	结果中的列数大于支持的最大列数。	400	客户端
OverMaxParquetBlockSize	Parquet 文件超过最大行组大小。	400	客户端
OverMaxRecordSize	输入或结果中记录的长度大于 1 MB 的 maxCharsPerRecord 限制。	400	客户端
ParquetParsingError	解析 Parquet 文件时出错。请检查文件，然后重试。	400	客户端
ParquetUnsupportedCompressionCodec	不支持指定的 Parquet 压缩编解码器。	400	客户端
ParseAsteriskIsNotAloneInSelectList	如果在 SQL 表达式中使用 * 而不带点符号，则 SELECT 列表中不支持其它表达式。	400	客户端
ParseCannotMixSqbandWildcardInSelectList	在 SQL 表达式的 SELECT 列表中，不能在同一个表达式中混用 [] 和 *。	400	客户端
ParseCastArity	SQL 表达式 CAST 的参数数量不正确。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
ParseEmptySelect	SQL 表达式包含一个空的 SELECT 子句。	400	客户端
ParseExpected2TokenTypes	找不到 SQL 表达式中预期的令牌。	400	客户端
ParseExpectedArgumentDelimiter	找不到 SQL 表达式中预期的参数分隔符。	400	客户端
ParseExpectedDatePart	找不到 SQL 表达式中预期的日期部分。	400	客户端
ParseExpectedExpression	找不到预期的 SQL 表达式。	400	客户端
ParseExpectedIdentifierForAlias	找不到 SQL 表达式中预期的别名标识符。	400	客户端
ParseExpectedIdentifierForAt	找不到 SQL 表达式中预期的 AT 名称标识符。	400	客户端
ParseExpectedIdentifierForGroupName	SQL 表达式中不支持 GROUP。	400	客户端
ParseExpectedKeyword	找不到 SQL 表达式中预期的关键字。	400	客户端
ParseExpectedLeftParenAfterCast	找不到 SQL 表达式中 CAST 后面的预期左括号。	400	客户端
ParseExpectedLeftParenBuiltinFunctionCall	找不到 SQL 表达式中预期的左括号。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
ParseExpectedLeftParenValueConstructor	找不到 SQL 表达式中预期的左括号。	400	客户端
ParseExpectedMember	SQL 表达式包含不支持的 MEMBER 用法。	400	客户端
ParseExpectedNumber	找不到 SQL 表达式中预期的数字。	400	客户端
ParseExpectedRightParenBuiltinFunctionCall	找不到 SQL 表达式中预期的右括号字符。	400	客户端
ParseExpectedTokenType	找不到 SQL 表达式中预期的令牌。	400	客户端
ParseExpectedTypeName	找不到 SQL 表达式中预期的类型名称。	400	客户端
ParseExpectedWhenClause	找不到 SQL 表达式中预期的 WHEN 子句。不支持 CASE。	400	客户端
ParseInvalidContextForWildcardInSelectList	在 SQL 表达式的 SELECT 列表中使用 * 无效。	400	客户端
ParseInvalidPathComponent	SQL 表达式包含无效的路径组件。	400	客户端
ParseInvalidTypeParameter	SQL 表达式包含无效的参数值。	400	客户端
ParseMalformedJoin	SQL 表达式中不支持 JOIN。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
ParseMissingIdentifierAt	找不到 SQL 表达式中 @ 符号后的预期标识符。	400	客户端
ParseNonUnaryAggregateFunctionCall	SQL 表达式中的聚合函数仅支持一个参数。	400	客户端
ParseSelectMissingFrom	SQL 表达式中的 SELECT 列表后缺少 FROM 子句。	400	客户端
ParseUnexpectedKeyword	SQL 表达式包含一个意外的关键字。	400	客户端
ParseUnexpectedOperator	SQL 表达式包含一个意外的运算符。	400	客户端
ParseUnexpectedTerm	SQL 表达式包含一个意外的术语。	400	客户端
ParseUnexpectedToken	SQL 表达式包含一个意外的令牌。	400	客户端
ParseUnknownOperator	SQL 表达式包含无效的运算符。	400	客户端
ParseUnsupportedAlias	SQL 表达式包含不支持的 ALIAS 用法。	400	客户端
ParseUnsupportedCallWithStar	SQL 表达式中仅支持将 (*) 作为参数的 COUNT。	400	客户端
ParseUnsupportedCase	SQL 表达式包含不支持的 CASE 用法。	400	客户端
ParseUnsupportedCaseClause	SQL 表达式包含不支持的 CASE 用法。	400	客户端
ParseUnsupportedLiteralsGroupBy	SQL 表达式包含不支持的 GROUP BY 用法。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
ParseUnsupportedSelect	SQL 表达式包含不支持的 SELECT 用法。	400	客户端
ParseUnsupportedSyntax	SQL 表达式包含不支持的语法。	400	客户端
ParseUnsupportedToken	SQL 表达式包含不支持的令牌。	400	客户端
TruncatedInput	对象解压缩失败。使用在请求中指定的格式检查对象是否已正确压缩。	400	客户端
UnauthorizedAccess	您没有权限执行此操作。	401	客户端
UnrecognizedFormatException	我们遇到了一种无效的记录类型。	400	客户端
UnsupportedFunction	我们遇到了一个不支持的 SQL 函数。	400	客户端
UnsupportedParquetType	不支持指定的 Parquet 类型。	400	客户端
UnsupportedRangeHeader	此操作不支持范围标头。	400	客户端
UnsupportedScanRangeInput	此类型的对象不支持扫描范围查询。	400	客户端
UnsupportedSqlOperation	我们遇到了不支持的 SQL 操作。	400	客户端
UnsupportedSqlStructure	我们遇到了不支持的 SQL 结构。请查看 SQL 参考。	400	客户端

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
UnsupportedStorageClass	我们遇到了不支持的存储类别。仅支持 STANDARD、STANDARD_IA 和 ONEZONE_IA 存储类别。	400	客户端
UnsupportedSyntax	我们遇到了无效的语法。	400	客户端
UnsupportedTypeForQuerying	您的查询包含不支持的比较类型（例如，验证 Parquet INT96 列类型是否大于 0）。	400	客户端
ValueParseFailure	SQL 表达式中出现时间戳解析失败。	400	客户端

与复制相关的错误代码列表

下表包含 Replication 操作可能返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidArgument	出现此错误可能的原因如下： • <Account> 元素为空。它必须包含有效的账户 ID。 • 在 <Account> 元素中指定的 AWS 账户必须与目标存储桶所有者匹配。 • ReplicationTime-Status 必须包含一个值。 •	400	客户端

错误代码	说明	HTTP 状态 代码	SOAP 故障 代码前缀
	ReplicationTime-ReplicationTimeValue 必须包含一个值。 • Replication-ReplicationTimeValue-Minutes 值必须为 15。 • ReplicationMetrics 必须包含 Status。 • ReplicationMetrics 必须包含 EventThreshold 。 • EventThreshold-ReplicationTimeValue-Minutes 值必须为 15。 • Rule ID 不得包含非 ASCII 字符。		

错误代码	说明	HTTP 状态 代码	SOAP 故障 代码前缀
InvalidRequest	出现此错误可能的原因如下： • <AccessControlTranslation> 中的 <Owner> 具有值，因此必须指定 <Account> 元素。 • <Account> 元素为空。它必须包含有效的账户 ID。 • 复制 destination 必须同时包含 ReplicationTime 和 Metrics，或者两者都不包含。 • ReplicationTime 和 ReplicationMetrics 必须具有相同的状态。 • 此 AWS 区域不支持 S3 Replication Time Control (S3 RTC)。	400	客户端
ReplicationConfigurationNotFoundErro r	此存储桶没有复制配置。	404 未找到	客户端

与标记相关的错误代码列表

下表包含 TagResource、UntagResource 和 ListTagsForResource 操作可能为 Storage Lens 组返回的特殊错误。有关常见 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidRequest	资源 ARN 中的 AWS 区域与在此请求中指定的区域不匹配。资源 ARN 中的 AWS 账户与在此请求中指定的账户 ID 不匹配。此 resourceArn 中的 AWS 分区无效。	400 Bad Request	不支持
InvalidTag	此请求包含无效的标签键或值。有效字符包括以下各项：[a-zA-Z+-. _:/] 。标签键最多可包含 128 个字符。标签值最多可包含 256 个字符。您的请求中有重复的标签键。用户定义的标签键不能以 aws: 开头。	400 Bad Request	不支持
NoSuchResource	指定的资源不存在。	404 Not Found	不支持
TagPolicyException	标签策略对于以下标签键不支持指定的值。	400 Bad Request	不支持
TooManyTags	标签数量超出 50 个标签的限制。	400 Bad Request	不支持

与 Amazon S3 注释相关的错误代码列表

下表包含注释操作

(PutObjectAnnotation、GetObjectAnnotation、ListObjectAnnotations、DeleteObjectAnnotation) 和注释相关的 CopyObject 行为可能返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
AccessDenied	拒绝访问。调用方无权执行所请求的注释操作，或者对象受合规模式下的对象锁定保护。	403 Forbidden	不支持
AnnotationLimitExceeded	对象注释计数超过限制 (1000)。	400 Bad Request	不支持
AnnotationNameTooLong	注释名称过长。注释名称的长度最多可达 512 字节。	400 Bad Request	不支持
AnnotationRequestConflict	当前正在对此注释执行冲突的操作。在跨区域复制期间删除源注释时，可能会发生这种情况。	409 Conflict	不支持
EntityTooSmall	您建议的上传小于支持的最小大小。注释有效载荷必须至少为 1 字节。	400 Bad Request	不支持
InvalidAnnotationName	注释名称包含无效字符，或者名称以保留前缀 (aws 或 s3，不区分大小写) 开头。注释名称只能包含 Unicode 字母、数字、下划线 (_)、句点 (.) 和连字符 (-)。	400 Bad Request	不支持
InvalidArgument	注释指令无效。有效值为 COPY 和 EXCLUDE。对于 ListObjectAnnotations，max-annotation-results 必须介于 1 和 1000 之间。	400 Bad Request	不支持
InvalidPrefix	您提供的注释前缀无效。	400 Bad Request	不支持
InvalidRequest	出现此错误的原因可能如下：该对象使用 SSE-C 进行加密，而 SSE-	400 Bad Request	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
	C 不支持注释。源对象具有无法复制到目标的注释，因为目标位于不受支持的区域。注释表配置请求无效 [例如，在指定加密时禁用表、启用已经启用的表、更新处于 CREATING (正在创建) 状态的表，或者提交空请求]。		
MethodNotAllowed	不允许对此资源使用指定的方法。例如，当请求以删除标记为目标时，就会发生此错误。	405 Method Not Allowed	不支持
NoSuchAnnotation	指定的注释不存在。	404 Not Found	不支持
NoSuchKey	指定的键不存在。	404 Not Found	不支持
NoSuchVersion	在请求中指定的版本 ID 与现有版本不匹配。	404 Not Found	不支持
NotImplemented	Amazon S3 目录存储桶不支持注释。	501 Not Implemented	不支持
OperationAborted	当前正在对此资源执行冲突的条件操作。这可能发生在并发修改期间，也可能发生在跨区域复制期间源注释计数或 ETag 发生变化时。请重试。	409 Conflict	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
PreconditionFailed	至少有一个您指定的先决条件不成立。x-amz-object-if-match ETag 与当前对象 ETag 不匹配。	412 Precondition Failed	不支持
UnsupportedMediaType	请求有效载荷包含不支持的媒体类型。注释有效载荷必须是有效的 UTF-8 编码文本。	415 Unsupported Media Type	不支持
UserAnnotationNameMustBeSpecified	必须指定用户注释名称。	400 Bad Request	不支持

Amazon S3 on Outposts 错误代码列表

下表包含 Amazon S3 on Outposts 操作可能返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
BadRequest	由于之前的删除尝试，该存储桶处于过渡状态。请稍后重试。	400 错误请求	不支持
InvalidRequest	出现此错误可能的原因如下： - 需要 Amazon VPC 配置。 - 不支持对 S3 on Outposts 接入点进行公开访问。	400 错误请求	客户端
InvalidOutpostState	请求对于 Outpost 的当前状态无效。	409 冲突	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidRequest	接入点未处于可以删除的状态。	400 错误请求	不支持
NoSuchOutpost	指定的 Outpost 不存在。	404 未找到	不支持
UnsupportedOperation	不支持指定的操作。	404 未找到	不支持
InsufficientCapacity	容量不足。	507 存储空间不足	不支持

Amazon S3 Storage Lens 存储统计管理工具错误代码列表

下表包含 Amazon S3 Storage Lens 存储统计管理工具操作可能返回的特殊错误。有关常见 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
AccessDenied	不支持将此区域作为 S3 Storage Lens 存储统计管理工具的主区域。	403 Forbidden	不支持
AccountNotAuthorized	未授权此账户使用 AWS Organizations。使用管理账户或委派管理员账户。	403 Forbidden	不支持
ActivityMetricsMustEnabled	必须启用活动指标。	400 错误请求	不支持
AWSOrganizationsNotInUseException	此账户不属于您组织的一部分。	403 Forbidden	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
DefaultConfigurationDeleteForbidden	无法删除默认配置。	403 Forbidden	不支持
DuplicateStorageLensGroupARN	此配置中有两个或更多条目属于同一个 Storage Lens 组 ARN。	400 Bad Request	不支持
EmptyExcludeContainer	出现此错误的原因如下： - 排除容器不能为空。 - 排除容器的存储桶数量不能为零。 - 排除容器的区域数量不能为零。	400 错误请求	不支持
EmptyExcludeElement	必须使用 Exclude 元素指定 Storage Lens 组。	400 Bad Request	不支持
EmptyIncludeContainer	出现此错误的原因如下： - 包含容器不能为空。 - 包含容器的存储桶数量不能为零。 - 包含容器的区域数量不能为零。	400 错误请求	不支持
InvalidAWSOrgArn	配置中存在格式错误的 AWS Organizations ARN。	400 错误请求	不支持
EmptyIncludeElement	必须使用 Include 元素指定 Storage Lens 组。	400 Bad Request	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidBucketFilter	组织级配置不支持存储桶筛选条件。	400 错误请求	不支持
InvalidConfigId	配置 ID 无效。	400 错误请求	不支持
InvalidDestination	S3 存储桶 ARN 格式错误。	400 错误请求	不支持
InvalidEncryptionMethod	只能指定一种加密方法。	400 错误请求	不支持
InvalidFilterForDefaultConfiguration	默认配置不得包含任何筛选条件。	400 错误请求	不支持
InvalidIncludeExcludeContainers	您可以在配置中指定 Include 容器或 Exclude 容器。您不能在配置中同时指定这两者。	400 错误请求	不支持
InvalidIncludeExcludeElements	只支持一个 Include 或 Exclude 元素。必须至少存在一个 Include 或 Exclude 元素。	400 Bad Request	不支持
InvalidKMSEncryptionKeyId	KMS 密钥 ID ARN 无效。	400 错误请求	不支持
InvalidMaximumPrefixDepth	MaxDepth 必须在范围 [1,10] 内。	400 错误请求	不支持
InvalidMinimumStorageBytesPercentage	MinStorageBytesPercentage 必须在范围 [1.00,100.00] 内。	400 错误请求	不支持
InvalidOrganizationARN	配置中的 AWS Organizations ARN 无效。	400 错误请求	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidOrganizationForDefaultConfiguration	默认配置不支持组织级指标。	400 错误请求	不支持
InvalidRegionForDefaultConfiguration	默认配置不支持指定的区域。	400 错误请求	不支持
InvalidRegionName	区域名称无效。	400 错误请求	不支持
InvalidStorageLensArn	输入中不需要 S3 Storage Lens 存储统计管理工具 ARN。	400 错误请求	不支持
InvalidStorageLensGroupARN	此 Storage Lens 组 ARN 无效，或您的账户中只支持 Storage Lens 组。此外，您必须遵循 Storage Lens 组 ARN 结构 <code>arn::s3:::storage-lens-group/</code> ，并遵守 64 个字符的限制。Storage Lens 组名称还可能仅包含以下字符：a-z、A-Z、0-9、连字符 (-) 和下划线 (_)。	400 Bad Request	不支持
MissingAccountLevelActivityMetrics	当在存储桶级启用活动指标时，必须在账户级启用活动指标。	400 错误请求	不支持
MissingBucketLevelActivityMetrics	当在账户级启用活动指标时，必须在存储桶级启用活动指标。	400 错误请求	不支持
MissingEncryptionMethod	加密方法不能为空。指定 SSE-KMS 或 SSE-S3。	400 错误请求	不支持
MissingPrefixLevelStorageMetrics	启用前缀级别后，前缀级的存储指标是必需的。	400 错误请求	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
OrganizationAccessDenied	未授权此账户添加 AWS Organizations。	403 禁止访问	不支持
OrgConfigurationNotSupported	指定的区域在配置中不支持 AWS Organizations。	403 禁止访问	不支持
ServiceNotEnabledForOrg	未为组织启用 S3 Storage Lens 存储统计管理工具服务相关角色。	403 禁止访问	不支持
StorageMetricsMustEnabled	必须启用前缀级存储指标。	400 错误请求	不支持
TooManyBuckets	存储桶容器的存储桶数量不能超过 50 个。	400 错误请求	不支持
TooManyRegions	区域容器的区域数量不能超过 50 个。	400 错误请求	不支持
TooManyStorageLensGroups	无法将超过 50 个 Storage Lens 组附加到 Storage Lens 存储统计管理工具控制面板。	400 Bad Request	不支持

下表包含 S3 Storage Lens 组操作可能返回的特殊错误。有关常见 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
AccessDenied	您无权执行 Storage Lens 组操作。不支持将此区域作为 S3 Storage Lens 组的主区域。	403 Forbidden	不支持
ConfigurationAlreadyExists	指定的配置已经存在。	409 Conflict	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
DuplicateElement	标签必须是唯一的。And 逻辑运算符包括重复的标签键。Or 逻辑运算符包含重复的标签。逻辑运算符包括重复的前缀或后缀。	400 Bad Request	不支持
InvalidAge	DaysLessThan 和 DaysGreaterThan 必须是正数。	400 Bad Request	不支持
InvalidFilter	筛选条件必须包含以下元素之一：And、Or、MatchAnyTag、MatchAnyPrefix、MatchAnySuffix、MatchObjectAge、MatchObjectSize。	400 Bad Request	不支持
InvalidLogicalOperator	逻辑运算符 And 或 Or 中必须至少存在两个子元素。	400 Bad Request	不支持
InvalidMatchAnyPrefix	MatchAnyPrefix 参数不能为空。	400 Bad Request	不支持
InvalidMatchAnySuffix	MatchAnySuffix 参数不能为空。	400 Bad Request	不支持
InvalidMatchAnyTag	MatchAnyTag 参数不能为空。	400 Bad Request	不支持
InvalidMatchObjectAge	MatchObjectAge 参数不能为空。	400 Bad Request	不支持
InvalidMatchObjectSize	MatchObjectSize 参数不能为空。	400 Bad Request	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
InvalidName	Storage Lens 组 Name 参数必须介于 1 到 64 个字符之间。Storage Lens 组 Name 参数必须使用 <code>^[a-zA-Z0-9\-_]+\$</code> 模式。	400 Bad Request	不支持
InvalidNumericCombination	此对象使用年限或对象大小组合无效。	400 Bad Request	不支持
InvalidPrefix	前缀的最大长度为 1024 个字符。前缀字符串不能为空。	400 Bad Request	不支持
InvalidSize	BytesLessThan 和 BytesGreaterThan 必须是正数。最大对象大小不能超过 50 TB。最小对象大小不能大于或等于 50 TB。	400 Bad Request	不支持
InvalidSuffix	后缀的最大长度为 1024 个字符。后缀字符串不能为空。	400 Bad Request	不支持
InvalidTag	对象标签键不能超过 128 个字符。对象标签键字符串不能为 null 或空。标签值的最大长度是 256 个字符。对象标签键包含无效的字符。对象标签键必须仅包含 a-z、A-Z、0-9、空格和以下字符： <code>^(_.:\/=+\-@]*)*\$</code> 。	400 Bad Request	不支持
MismatchedName	在请求中指定的名称与 Storage Lens 组名称不匹配。	400 Bad Request	不支持
TooManyConfigurations	您尝试创建的 Storage Lens 组配置超过了支持的 50 个。	400 Bad Request	不支持

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
TooManyElements	Element 超过了逻辑运算符中支持的最大元素数。只支持 10 个前缀、后缀或标签。	400 Bad Request	不支持

Amazon S3 对象 Lambda 错误代码列表

下表包含 S3 对象 Lambda 可能返回的特殊错误。有关一般 Amazon S3 错误的信息以及错误代码列表，请参阅[错误响应](#)。

在非 GetObject 请求期间从支持的接入点收到的错误响应将原封不动地发送给调用方。

错误代码	说明	HTTP 状态代码	
LambdaInvalidResponse	当 WriteGetObjectResponse 向 AWS Lambda 响应 ValidationError 时，返回给原始调用方。 有关更多详细信息，请参阅 ValidationError 消息。并非所有 ValidationError 的情况都会导致 LambdaInvalidResponse 错误。	400 Bad Request	
LambdaInvocationFailed	Lambda 函数调用失败。 当 S3 对象 Lambda 无法成功调用已配置的 Lambda 函数时，调用方可能会收到以下错误。 错误消息可能包含有关在调用函数时 AWS Lambda 服务返回的最终错误的详细信息（例如，状态代码、错误代码、错误消息和请求 ID）。	400 Bad Request	

错误代码	说明	HTTP 状态代码
LambdaNotFound	找不到 AWS Lambda 函数。 尝试调用配置的 Lambda 函数、版本或别名时找不到它。确保 S3 对象 Lambda 接入点配置指向正确的 Lambda 函数 ARN。 错误消息可能包含有关在调用函数时 AWS Lambda 服务返回的最终错误的详细信息（例如，状态代码、错误代码、错误消息和请求 ID）。	404 Not Found
LambdaPermissionError	调用方未获授权，无法调用 Lambda 函数。 调用方必须具有调用 Lambda 函数的权限。检查附加到调用方的策略，并确保这些策略已获准将 <code>lambda:Invoke</code> 用于所配置的函数。 错误消息可能包含有关在调用函数时 AWS Lambda 服务返回的最终错误的详细信息（例如，状态代码、错误代码、错误消息和请求 ID）。	403 Forbidden

错误代码	说明	HTTP 状态代码	
LambdaResponseNotReceived	Lambda 函数在未成功调用 WriteGetObjectResponse 的情况下退出。 GetObject 响应数据由 Lambda 函数通过调用 WriteGetObjectResponse API 操作来提供。通过该函数的 Amazon CloudWatch 日志，可以更深入地了解为什么该函数正常退出，却未能成功调用此 API 操作。	500 Internal Service Error	
LambdaRuntimeError	Lambda 函数在执行过程中失败。 从 Lambda 函数收到了显式错误。有关失败的详细信息，请查看 AWS CloudFormation 日志。	500 Internal Service Error	
LambdaTimeout	Lambda 函数未在支持的时间内进行响应。 Lambda 函数未能在 60 秒内完成其对 WriteGetObjectResponse 的调用。	500 Internal Service Error	

错误代码	说明	HTTP 状态代码	
SlowDown	降低涉及 AWS Lambda 的操作的请求速率。 函数调用受 AWS Lambda 限制，可能是因为它已达到已配置的并发限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的为 Lambda 函数管理并发。 错误消息可能包含有关在调用函数时 AWS Lambda 服务返回的最终错误的详细信息（例如，状态代码、错误代码、错误消息和请求 ID）。	503 Slow Down	
ValidationError	验证错误可能从 WriteGetObjectResponse API 操作中返回，并且可能由于多种原因而发生。有关更多详细信息，请参阅错误消息。	400 Bad Request	

Amazon S3 异步错误代码列表

下表包含异步请求可能返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

当您查询有关异步请求的状态（例如使用 DescribeMultiRegionAccessPointOperation）时，将返回这些错误。由于这些请求是异步的，因此所有这些错误的状态代码都为 200 OK。

错误代码	说明	HTTP 状态代码	
AccessDenied	访问遭拒绝。	200 OK	
InternalErrors	发生内部服务器错误。	200 OK	

错误代码	说明	HTTP 状态代码	
MalformedPolicy	指定的策略语法无效。	200 OK	
MultiRegionAccessPointAlreadyOwnedByYou	您已拥有一个同名的多区域接入点。	200 OK	
MultiRegionAccessPointModifiedByAnotherRequest	操作失败，因为另一个请求正在修改指定的资源。请尝试在前一个请求完成后重新提交您的请求。	200 OK	
MultiRegionAccessPointNotReady	指定的多区域接入点尚未准备好进行更新。	200 OK	
MultiRegionAccessPointSameBucketRegion	用于创建多区域接入点的存储桶不能位于同一区域。	200 OK	
MultiRegionAccessPointUnsupportedRegion	为创建多区域接入点而提供的其中一个存储桶位于不受支持的区域中。	200 OK	
NoSuchBucket	指定的存储桶不存在。	200 OK	
NoSuchMultiRegionAccessPoint	指定的多区域接入点不存在。	200 OK	

Amazon S3 访问权限管控错误代码列表

下表包含 S3 访问权限管控请求可能返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	
AccessGrantAlreadyExists	指定的访问权限管控已存在	409	

错误代码	说明	HTTP 状态代码	
AccessGrantsInstanceAlreadyExists	访问权限管控实例已存在	409	
AccessGrantsInstanceNotEmptyError	请在删除访问权限管控实例之前清理位置	400	
AccessGrantsInstanceNotExistsError	访问权限管控实例不存在	404	
AccessGrantsInstanceResourcePolicyNotExists	访问权限管控实例资源策略不存在	404	
AccessGrantsLocationAlreadyExistsError	指定的访问权限管控位置已存在	409	
AccessGrantsLocationNotEmptyError	请在删除访问权限管控位置之前清理访问权限管控	400	
AccessGrantsLocationsQuotaExceededError	已超过访问权限管控位置配额。访问权限管控位置配额： <i><value></i> 。如果需要增加，请联系 S3。	409	
AccessGrantsQuotaExceededError	已超过访问权限管控配额。访问权限管控配额： <i><value></i> 。如果需要增加，请联系 S3。	409	
InvalidTag	您的请求中有重复的标签键。删除重复的标签键，然后重试。	400	
InvalidAccessGrant	指定的访问权限管控无效	400	
InvalidAccessGrantLocation	指定的访问权限管控位置无效	400	
InvalidIamRole	指定的 IAM 角色无效	400	

错误代码	说明	HTTP 状态代码	
InvalidIdentityCenterInstance	指定的身份中心实例无效	400	
InvalidResourcePolicy	指定的资源策略无效	400	
InvalidResourcePolicy	指定的资源策略无效	400	
InvalidTag	此请求包含无效的标签键或值。有效字符包括以下各项：[a-zA-Z+-. _:/]。标签键最多可包含 128 个字符。标签值最多可包含 256 个字符。	400	
NoSuchAccessGrantError	指定的访问权限管控不存在	404	
NoSuchAccessGrantsLocationError	指定的访问权限管控位置不存在	404	
AccessDenied	您对所请求的 S3 前缀 <i><requested target></i> 没有 <i><requested permission></i> 权限	403 Forbidden	
StsNotAuthorizedError	调用 GetDataAccess 操作时发生错误 (StsNotAuthorizedError)：用户 access-grants.s3.amazonaws.com 无权对资源 <i><IAM Role ARN></i> 执行 sts:AssumeRole	403	
StsPackedPolicyTooLargeError	调用 GetDataAccess 操作时出错 (StsPackedPolicyTooLargeError)：序列化令牌过大，无法进行会话	400	
StsValidationError	错误消息因验证错误而异。	400	

错误代码	说明	HTTP 状态代码	
InvalidTags	标签键不能以系统标签的 AWS 保留前缀开头。	400	
TooManyTags	标签数量超出 50 个标签的限制。移除一些标签，然后重试。	400	

Amazon FSx 相关的错误代码列表

下表包含通过 Amazon S3 接入点向 Amazon FSx 发出的请求可能会返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码	
AccessDenied	出现此错误可能的原因如下： - 调用方无权执行指定的 S3 操作。 - 调用方无权访问指定密钥的 Amazon FSx 资源。	403 Forbidden	
InsufficientCapacity	已达到文件系统的最大存储容量。	507 Insufficient Capacity	
InvalidKey	指定的键无效。	400 Bad Request	
SlowDown	请降低请求速率。	503 Slow Down	

Amazon S3 表类数据存储服务错误代码列表

下表包含 Amazon S3 表类数据存储服务操作可能返回的特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

错误代码	说明	HTTP 状态代码
AccessDeniedException	由于您没有所需的权限，因此无法执行该操作。	403 Forbidden
BadRequestException	请求无效或格式错误。	400 Bad Request
ConflictException	请求失败，因为与之前的写入存在冲突。您可以重试请求。	409 Conflict
ForbiddenException	调用者无权发出该请求。	403 Forbidden
InternalServerErrorException	请求由于内部服务器错误而失败。	500 Internal Server Error
NotFoundException	由于找不到指定的资源，请求被拒绝。	404 Not Found
TooManyRequestsException	超出了有关每秒请求数的限制。	429 Too Many Requests

Amazon S3 排错最佳实践

许多错误响应包含额外的结构化数据，旨在供开发人员阅读和理解以诊断编程错误。例如，如果您使用与服务器上计算出的摘要不匹配的 REST PUT 请求发送 Content-MD5 标头，您将收到 BadDigest 错误。错误响应还包括我们计算出的摘要的详细元素，以及预期内容的摘要。在开发过程中，您可以使用此信息诊断错误。在生产过程中，运行良好的程序可能会将此信息包含在其错误日志中。

设计使用 Amazon S3 的应用程序时，正确处理 Amazon S3 错误非常重要。本节描述了设计应用程序时，您应该考虑的问题。

重试 InternalErrors

内部错误是指在 Amazon S3 环境中发生的错误。

可能还没有处理收到 InternalError 响应的请求。例如，如果 PUT 请求返回 InternalError，则后续的 GET 操作可能会检索旧的值或更新的值。

如果 Amazon S3 返回 InternalError 响应，请重新提交请求。

针对重复的 SlowDown 错误调整应用程序

与其他分布式系统一样，S3 的保护机制能够检测出有意或无意的资源过度消耗，并做出相应的反应。在较高的请求速率触发了其中某个保护机制后，将发生 SlowDown 错误。降低您的请求速率将减少或消除此类型的错误。一般而言，大多数用户不会经常遇到这些错误；但是，如果您希望了解更多信息，或遇到了严重或意外的 SlowDown 错误，请将该错误发布到我们的 [Amazon S3 开发人员论坛](https://aws.amazon.com/premiumsupport/)或在以下网址注册支持：[\(https://aws.amazon.com/premiumsupport/\)](https://aws.amazon.com/premiumsupport/)。

隔离错误

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止（EOL）。我们建议您使用 REST API 或 AWS 开发工具包。

Amazon S3 提供了一组由 SOAP 和 REST API 共同使用的错误代码。SOAP API 将返回标准的 Amazon S3 错误代码。REST API 的设计与标准的 HTTP 服务器相似，它与现有的 HTTP 客户端（例如，浏览器、HTTP 客户端库、代理、缓存等等）进行交互。为了确保 HTTP 客户端能够正确地处理错误，我们将每个 Amazon S3 错误都映射为一个 HTTP 状态代码。

HTTP 状态代码没有 Amazon S3 错误代码那样直观，并且包含的错误信息也较少。例如，NoSuchKey 和 NoSuchBucket Amazon S3 错误均映射为 HTTP 404 Not Found 状态代码。

尽管 HTTP 状态代码包含较少的错误信息，但能够理解 HTTP 而不是 Amazon S3 API 的客户端通常能够正确地处理错误。

因此，处理错误或向最终用户报告 Amazon S3 错误时，请使用 Amazon S3 错误代码而不是 HTTP 状态代码，这是因为前者包含最详细的错误信息。此外，调试应用程序时，您还应该参考 XML 错误响应的人类可读的 <Details> 元素。

AWS 术语表

有关最新的 AWS 术语，请参阅 AWS 词汇表 参考中的 [AWS 词汇表](#)。

Amazon S3 资源

下表列出了在您使用此服务时可为您提供帮助的相关资源。

资源	说明
Amazon Simple Storage Service 用户指南 。	该入门指南通过简单的使用案例，提供了相关服务的快速教程。
Amazon Simple Storage Service 开发人员指南 。	开发人员指南介绍了如何使用 Amazon S3 操作来完成任务。
Amazon S3 技术常见问题解答	“常见问题解答”涵盖了开发人员对此产品提出的最热门的 20 个问题。
Amazon S3 发行说明	发行说明概括介绍了当前发行版本的情况。特别说明了一些新功能、修复和已知问题。
用于 Amazon Web Services 的工具	查找相关文档、代码示例、发行说明和其它信息的统一入口，有助于您通过 AWS SDK 和工具构建创新的应用程序。
AWS 管理控制台	该控制台可让您执行 Amazon S3 的大多数功能而无需编程。
开发论坛	一个基于社区的论坛，供开发人员讨论与 Amazon Web Services 相关的技术问题。
支持 Center	AWS 技术支持的主页，从中可以访问我们的开发人员论坛、技术常见问题解答、服务状态页面和高级支持等。
支持	提供有关 支持 信息的主要 Web 页面，它是一种一对一、快速响应的支持渠道，有助于您在 AWS 基础设施服务上构建和运行应用程序。
Amazon S3 产品信息	提供了 Amazon SQS 相关信息的主要 Web 页面。
联系我们	用来查询 AWS 账单、账户、事件、滥用等信息的中心联络点。

资源	说明
使用条件	有关在 Amazon.com 和其他主题中的版权和商标使用的详细信息。

附录

主题

- [附录：SelectObjectContent 响应](#)
- [附录：OPTIONS 对象](#)
- [附录：SOAP API](#)
- [附录：对请求进行身份验证 \(AWS 版本签名 2 \)](#)
- [附录：生命周期配置 API \(已弃用 \)](#)

附录：SelectObjectContent 响应

说明

Amazon S3 Select 操作基于简单结构化查询语言 (SQL) 语句筛选 Amazon S3 对象的内容。鉴于此操作的响应大小是未知的，Amazon S3 Select 将响应作为一系列消息进行流式传输，并在响应中包含一个将 **chunked** 作为其值的 Transfer-Encoding 标头。

有关 Amazon S3 Select 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[从对象中选择内容](#)。

有关将 SQL 与 Amazon S3 Select 结合使用的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[Amazon S3 Select 和 Amazon Glacier Select 的 SQL 参考](#)。

响应

成功的 Amazon S3 Select 操作会返回 200 OK 状态代码。

响应标头

此操作的实施仅使用大多数响应通用的响应标头。有关更多信息，请参阅[通用响应标头](#)。

响应正文

鉴于 Amazon S3 Select 的响应大小是未知的，Amazon S3 将响应作为一系列消息进行流式传输，并在响应中包含一个将 **chunked** 作为其值的 Transfer-Encoding 标头。以下示例显示顶层的响应格式：

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

每个消息都包含两部分：前导信息和数据。前导信息部分包含：1) 消息的总字节长度，2) 所有标头的组合字节长度。数据部分包含：1) 标头，2) 有效载荷。

每个部分均以 4 字节大端整数校验和 (CRC) 结尾。Amazon S3 Select 使用 CRC32 (通常称为 GZIP CRC32) 来计算这两个 CRC。有关 CRC32 的更多信息，请参阅[GZIP 文件格式规范版本 4.3](#)。

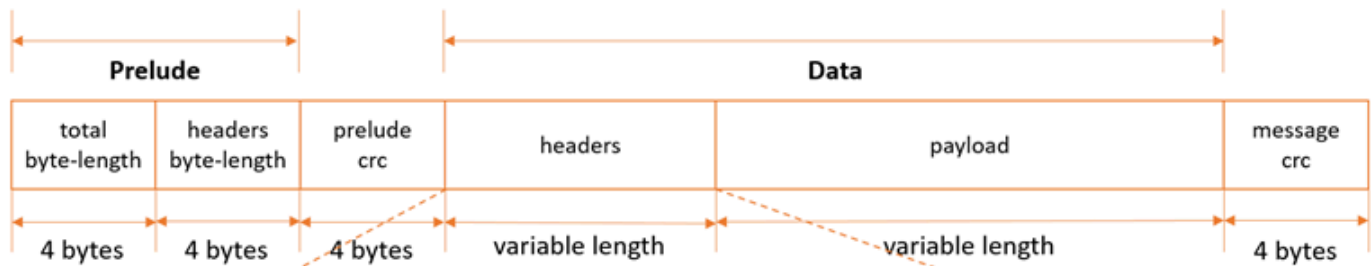
总消息开销 (包括前导信息和两个校验和) 为 16 字节。

Note

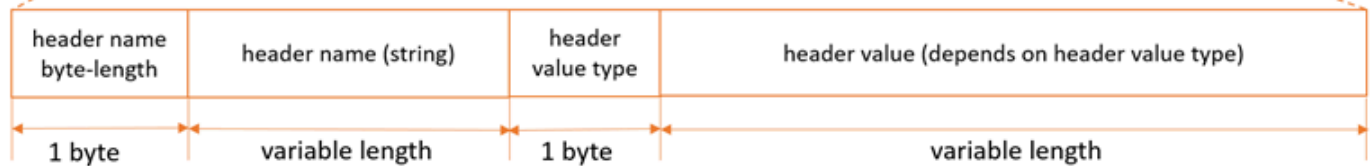
消息中的所有 integer 值均按网络字节顺序或大端序排列。

下图显示了构成消息和标头的组件。请注意，每个消息有多个标头。

Message:



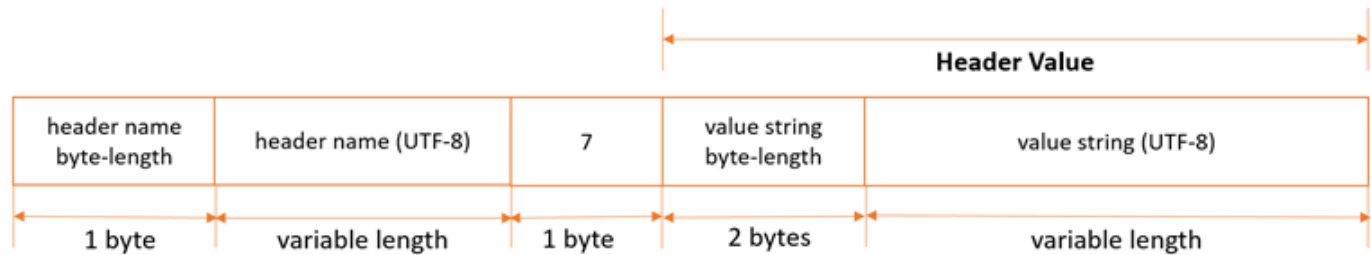
Headers (multiple headers per message):



Note

对于 Amazon S3 Select，标头值类型始终为 7（类型=字符串）。对于这种类型，标头值由两部分组成：一个 2 字节的大端整数长度，以及一个具有该字节长度的 UTF-8 字符串。下图显示了构成 Amazon S3 Select 标头的组件。

Amazon S3 Select Headers (type=String):



有效载荷字节长度计算（这两种计算是等效的）：

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{sizeof}(\text{total\_length}) - \text{sizeof}(\text{header\_length}) - \text{sizeof}(\text{prelude\_crc}) - \text{sizeof}(\text{message\_crc})$
- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - 16$

每个消息都包含以下组件：

- 前导信息：大小始终固定为 8 字节（两个各为 4 字节的字段）。
 - 前四个字节：总字节长度：整个消息的大端整数字节长度（包括 4 字节总长度字段本身）。
 - 第二个四字节：标头字节长度：消息标头部分的大端整数字节长度（不包括标头长度字段本身）。
- 前导信息 CRC：消息前导信息部分的 4 字节大端整数校验和（CRC）（不包括 CRC 本身）。前导信息具有不同于消息 CRC 的单独 CRC（请参阅下文），以确保可以立即检测到损坏的字节长度信息，而不会导致无法控制的缓冲行为。
- 标头：一组为消息添加注释的元数据，如消息类型、有效载荷格式等。消息可以有多个标头，因此消息的这一部分可能具有不同的字节长度，具体取决于消息类型。标头是键-值对，其中键和值都是 UTF-8 字符串。标头可按任何顺序出现在消息的标头部分内，并且任何给定的标头类型只能出现一次。

对于 Amazon S3 Select，以下是标头名称的列表和有效值的集合，具体取决于消息类型。

- MessageType 标头：
 - HeaderName => ":message-type"
 - 有效的 HeaderValue => "error"、"event"
- EventType 标头：
 - HeaderName => ":event-type"
 - 有效的 HeaderValue => "Records"、"Cont"、"Progress"、"Stats"、"End"
- ErrorCode 标头：
 - HeaderName => ":error-code"
 - 有效的 HeaderValue => [SELECT 对象内容错误代码列表](#) 部分的表中的错误代码。
- ErrorMessage 标头：
 - HeaderName => ":error-message"
 - 有效的 HeaderValue => 服务返回的错误消息，有助于诊断请求级错误。
- 有效载荷：可以是任何内容。

- 消息 CRC：从消息开头到校验和开头的 4 字节大端整数校验和 (CRC) (即消息中除消息 CRC 本身之外的所有内容)。

每个标头都包含以下组件。每个消息可能有多个标头。

- 标头名称字节长度：标头名称的字节长度。
- 标头名称：标头的名称，指示标头类型。有效值：":message-type" ":event-type" ":error-code" ":error-message"
- 标头值类型：指示标头值类型的枚举。对于 Amazon S3 Select，此值始终为 7。
- 值字符串字节长度：(对于 Amazon S3 Select) 标头值字符串的字节长度。
- 标头值字符串：(对于 Amazon S3 Select) 标头字符串的值。此字段的有效值因标头的类型而异。有关每种标头类型和消息类型的有效值，请参阅以下各节。

对于 Amazon S3 Select，响应可以是以下类型的消息：

- 记录消息：可以包含单条记录、不完整的记录或多条记录。根据结果的大小，响应可能包含一条或多条这样的消息。
- 继续消息：Amazon S3 会定期发送此消息，以使 TCP 连接保持打开状态。这些消息随机出现在响应中。客户端必须检测消息类型并相应地进行处理。
- 进度消息：如果收到请求，Amazon S3 会定期发送此消息。它包含有关已开始但尚未完成的查询的进度信息。
- 统计数据消息：Amazon S3 在请求结束时发送此消息。它包含有关查询的统计数据。
- 结束消息：表示请求已完成，将不再发送消息。在客户端收到 End 消息之前，您不应认为请求已完成。
- RequestLevelError 消息：如果请求因任何原因而失败，Amazon S3 会发送此消息。它包含失败的错误代码和错误消息。如果 Amazon S3 发送了 RequestLevelError 消息，它就不会发送 End 消息。

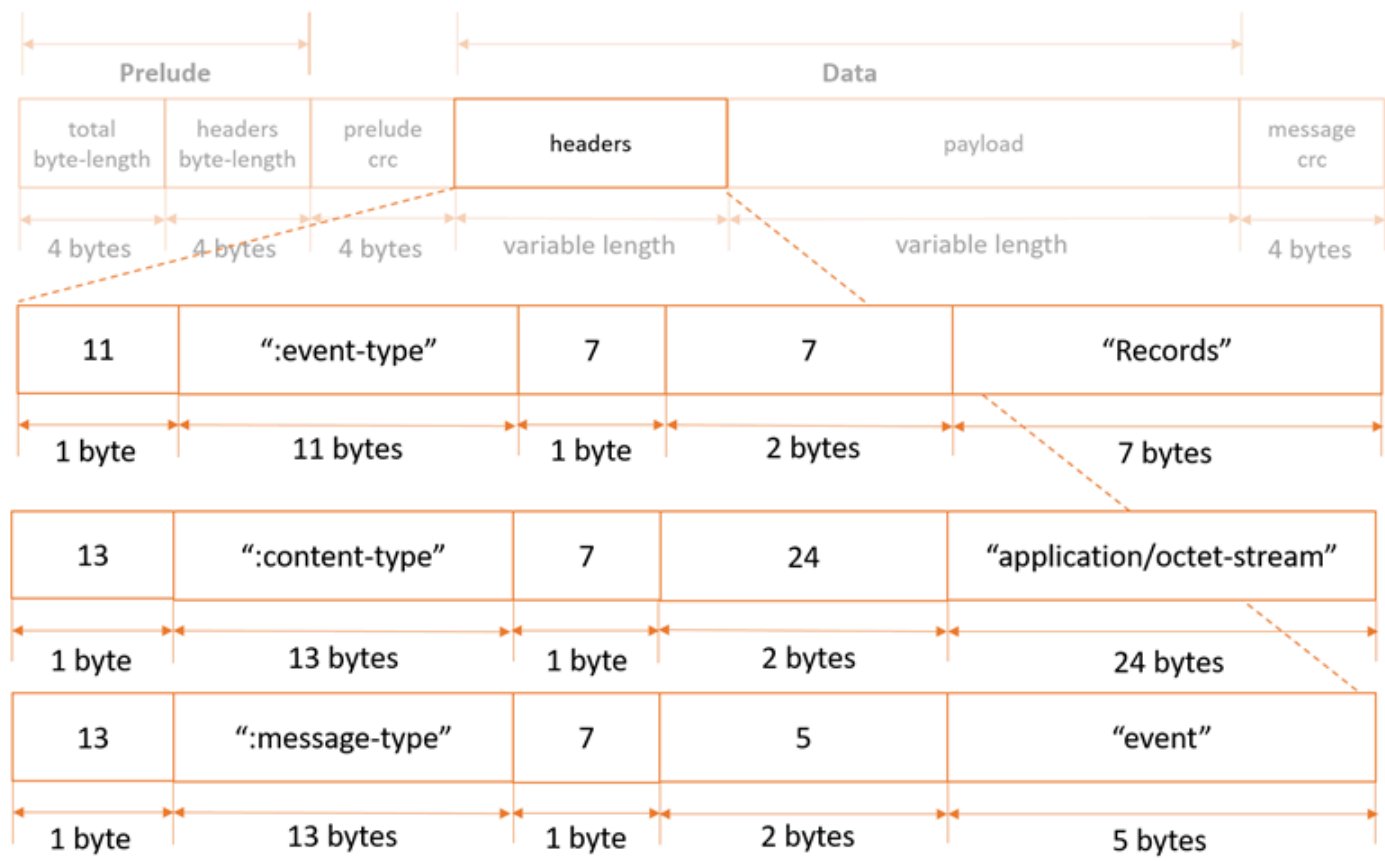
以下各节更详细地介绍了每种消息类型的结构。

有关使用此协议的示例代码和单元测试，请参阅 GitHub 网站上的 [AWS C Event Stream](#)。

记录消息

标头规范

记录消息包含三个标头，如下所示：



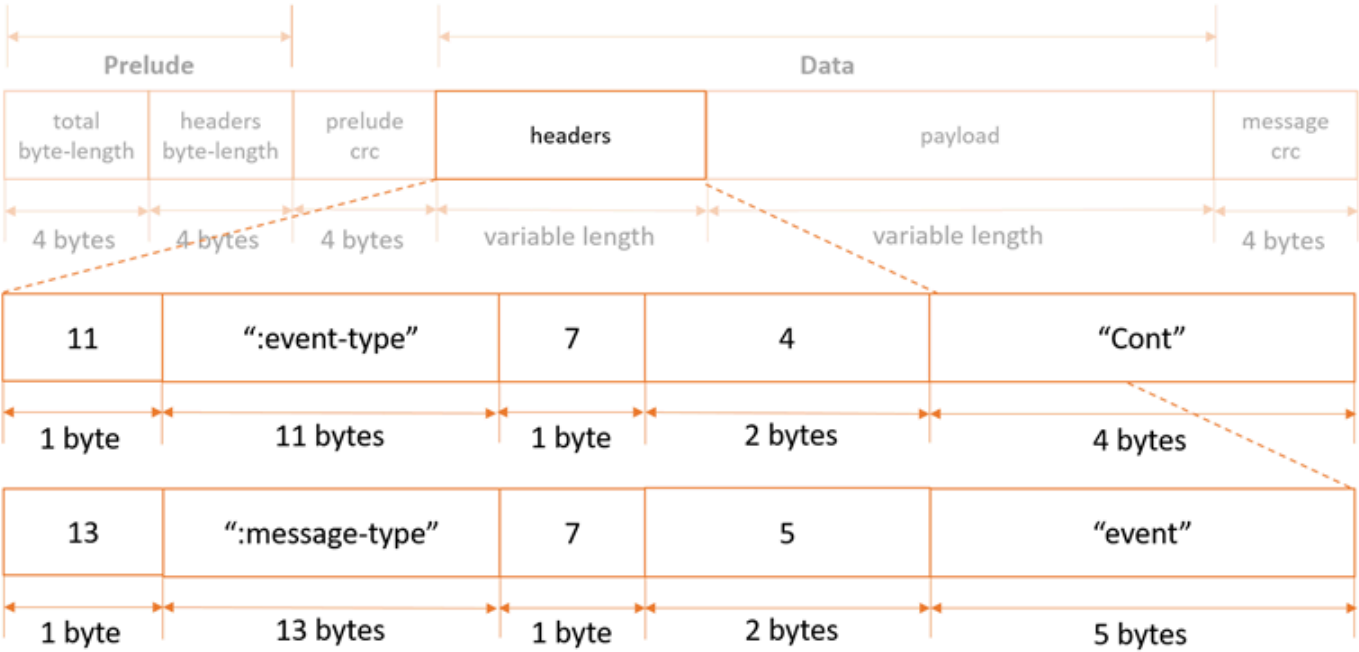
有效载荷规范

记录消息有效载荷可以包含单条记录、不完整的记录或多条记录。

继续消息

标头规范

继续消息包含两个标头，如下所示：



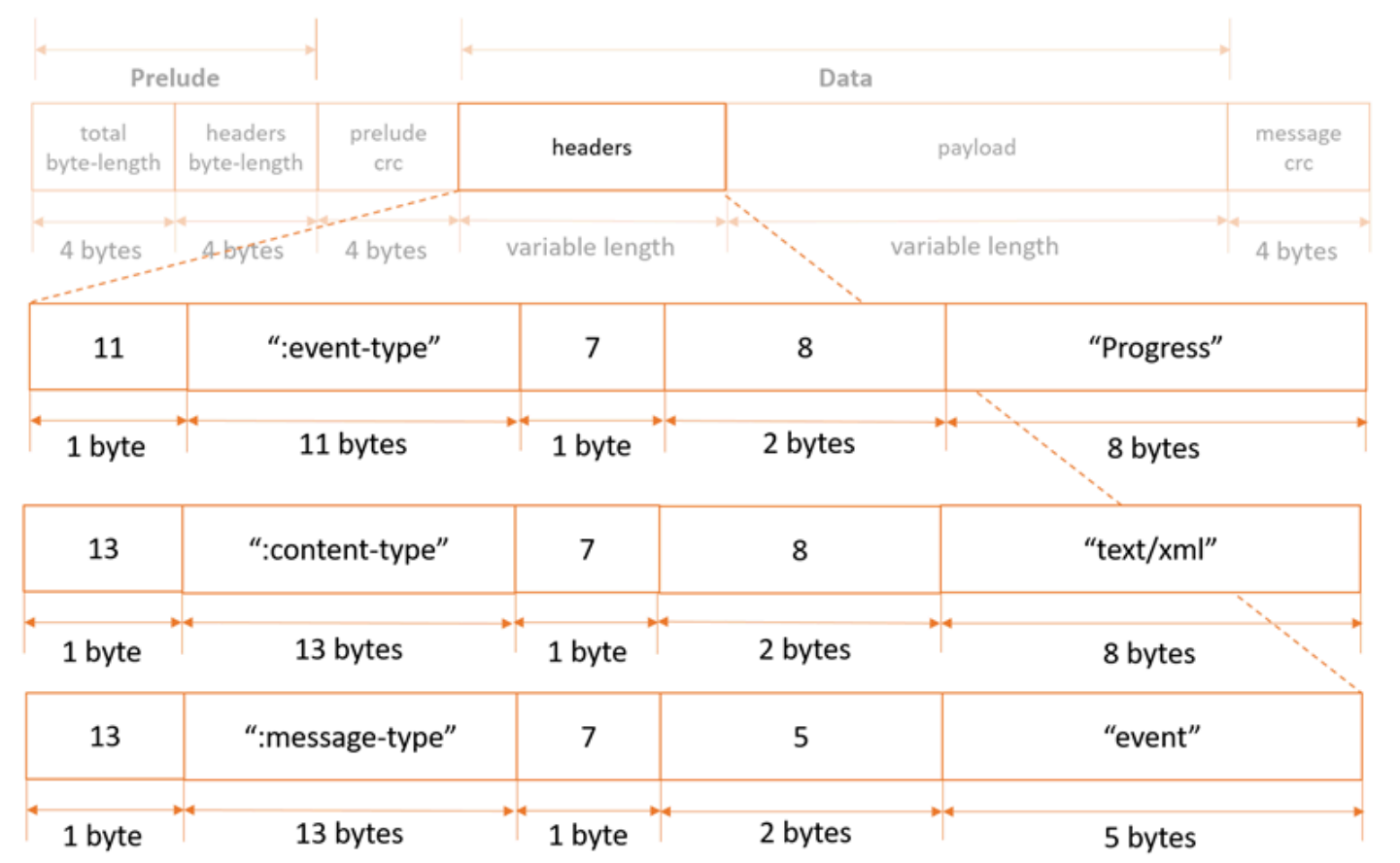
有效载荷规范

继续消息没有有效载荷。

进度消息

标头规范

进度消息包含三个标头，如下所示：



有效载荷规范

进度消息有效载荷是一个 XML 文档，其中包含有关请求进度的信息。

- BytesScanned => 解压缩前已处理的字节数（如果文件已压缩）。
- BytesProcessed => 解压缩后已处理的字节数（如果文件已压缩）。
- BytesReturned => Amazon S3 返回的记录有效载荷数据的当前字节数。

对于未压缩的文件，BytesScanned 和 BytesProcessed 相等。

示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Progress>
```

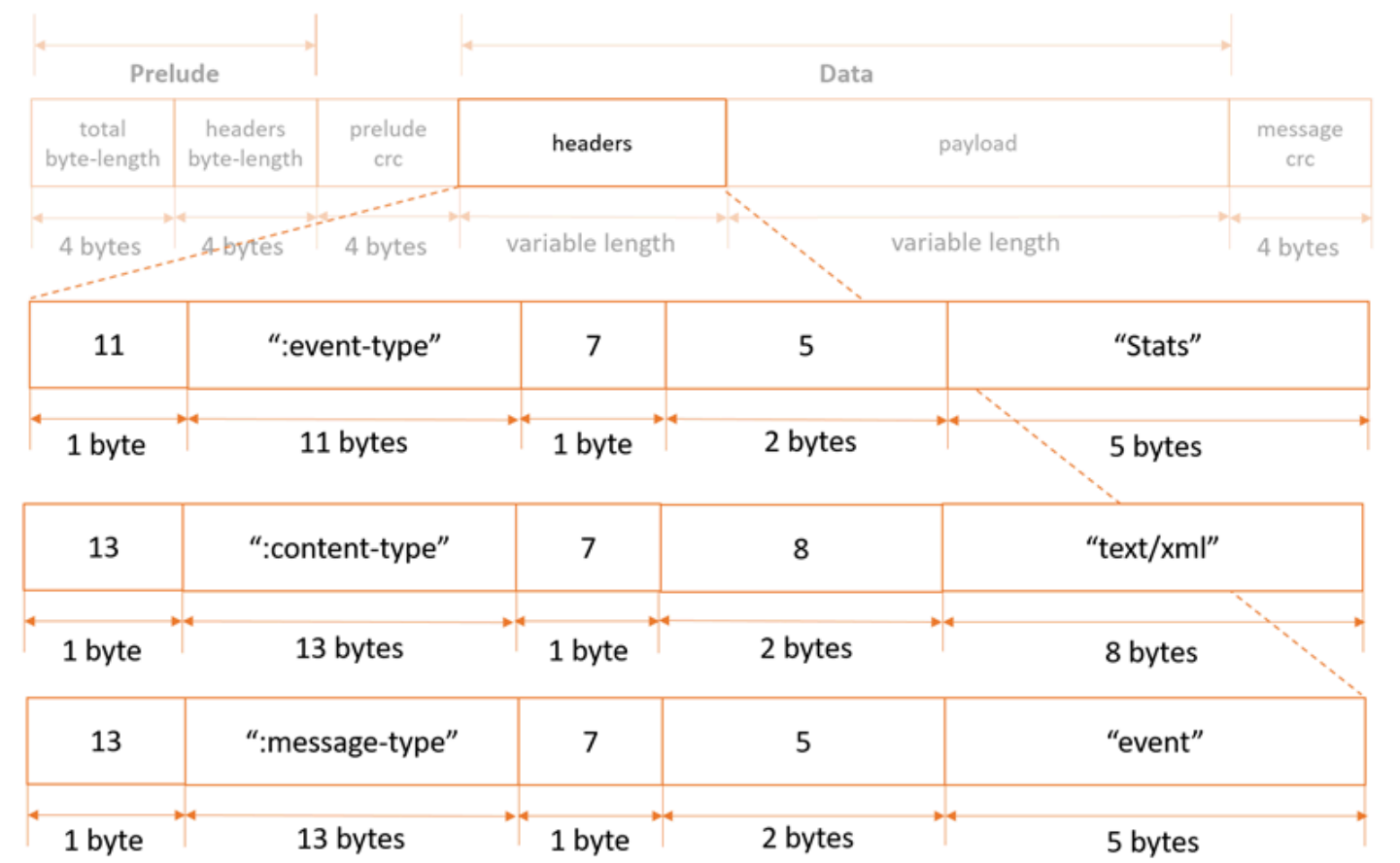


```
</Progress>
```

统计数据消息

标头规范

统计数据消息包含三个标头，如下所示：



有效载荷规范

统计数据消息有效载荷是一个 XML 文档，其中包含处理完成时请求的统计数据。

- BytesScanned => 解压缩前已处理的字节数（如果文件已压缩）。
- BytesProcessed => 解压缩后已处理的字节数（如果文件已压缩）。
- BytesReturned => Amazon S3 返回的记录有效载荷数据的总字节数。

对于未压缩的文件，BytesScanned 和 BytesProcessed 相等。

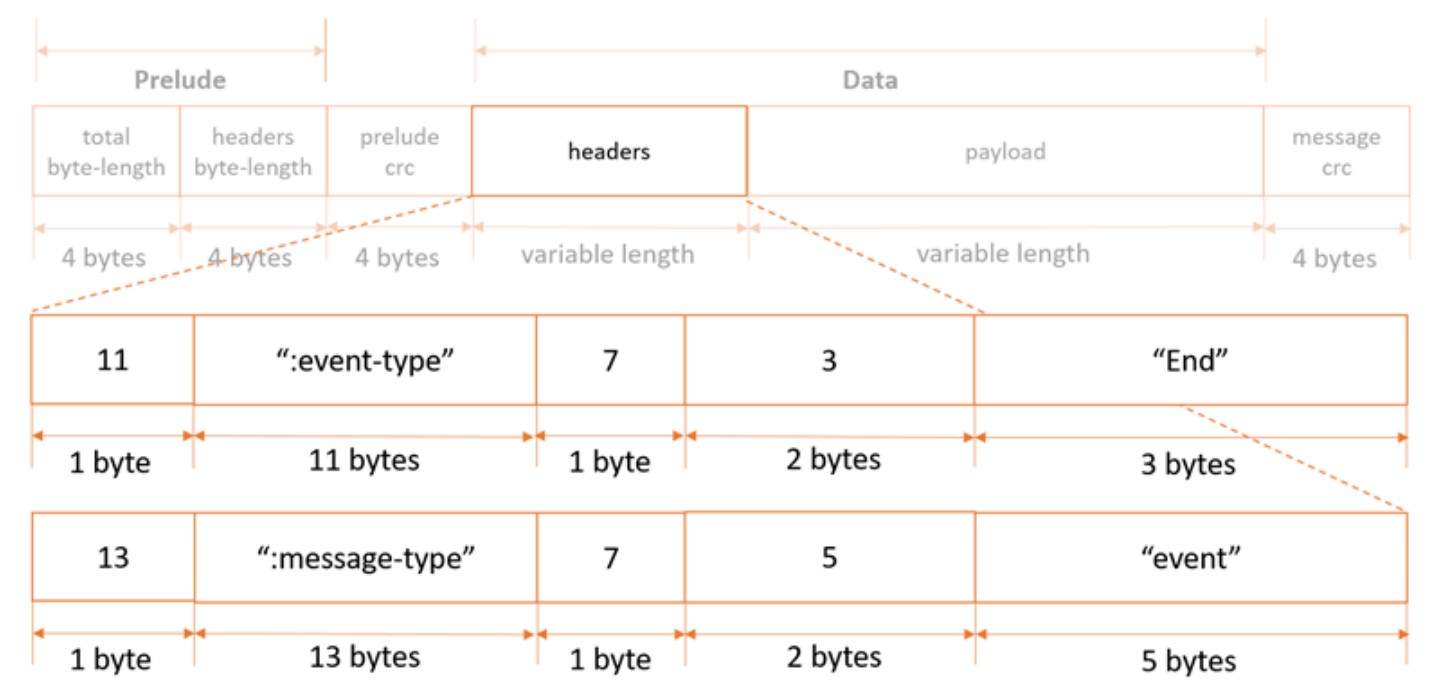
示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

结束消息

标头规范

结束消息包含两个标头，如下所示：



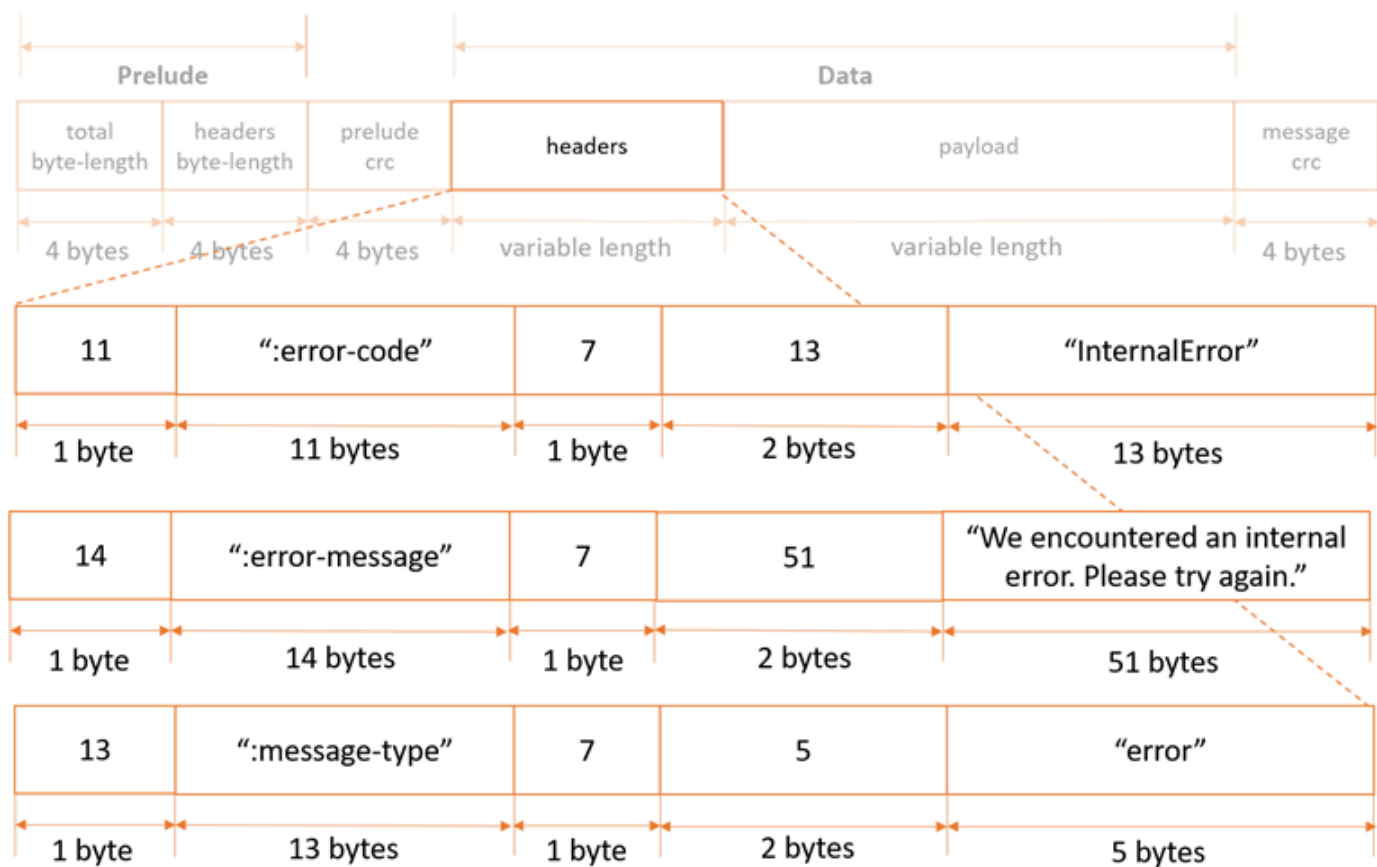
有效载荷规范

结束消息没有有效载荷。

请求级错误消息

标头规范

请求级错误消息包含三个标头，如下所示：



有关可能的错误代码和错误消息的列表，请参阅 [SELECT 对象内容错误代码列表](#)。

有效载荷规范

请求级错误消息没有有效载荷。

相关资源

- [SelectObjectContent](#)
- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

附录：OPTIONS 对象

说明

浏览器可以向 Amazon S3 发送此预检请求，以确定它能否发送包含特定源、HTTP 方法和标头的实际请求。

Amazon S3 通过可让您在存储桶上添加 cors 子资源来支持跨源资源共享 (CORS)。当浏览器发送此预检请求时，Amazon S3 通过评估在 cors 配置中定义的规则进行响应。

如果未在存储桶上启用 cors，则 Amazon S3 会返回 403 Forbidden 响应。

有关 CORS 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[启用跨源资源共享](#)。

请求

语法

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

请求参数

此操作不引入任何特定的请求参数，但它可能包含实际请求所需的任何请求参数。

请求标头

名称	描述	必填
Origin	标识向 Amazon S3 发出的跨源请求的源。例如，http://www.example.com。 类型：字符串	是

名称	描述	必填
	默认值：无	
Access-Control-Request-Method	标识将在实际请求中使用哪种 HTTP 方法。 类型：字符串 默认值：无	是
Access-Control-Request-Headers	将在实际请求中发送的 HTTP 标头的以逗号分隔的列表。 例如，要放入采用服务器端加密的对象，此预检请求将确定它是否可以在请求中包含 x-amz-server-side-encryption 标头。 类型：字符串 默认值：无	否

请求元素

此操作的实施不使用请求元素。

响应

响应标头

标题	说明
Access-Control-Allow-Origin	您在请求中发送的源。如果不支持您的请求中的源，则 Amazon S3 不会在响应中包含此标头。 类型：字符串
Access-Control-Max-Age	可以缓存预检请求的结果的时长（以秒为单位）。

标题	说明
	类型：字符串
Access-Control-Allow-Methods	在原始请求中发送的 HTTP 方法。如果不支持请求中的方法，则 Amazon S3 不会在响应中包含此标头。 类型：字符串
Access-Control-Allow-Headers	浏览器可以在实际请求中发送的 HTTP 标头的以逗号分隔的列表。如果不支持任何请求的标头，Amazon S3 将不会在响应中包含该标头，而响应也不会包含任何带有 Access-Control 前缀的标头。 类型：字符串
Access-Control-Expose-Headers	HTTP 标头的以逗号分隔的列表。此标头为 JavaScript 客户端提供了在针对实际请求的响应中访问这些标头的权限。 类型：字符串

响应元素

此操作的实施不返回响应元素。

示例

示例：向已启用 **cors** 的存储桶发送预检 OPTIONS 请求

浏览器可以将此预检请求发送到 Amazon S3，以确定它能否将实际的 PUT 请求从 `http://www.example.com` 源发送到名为 `examplebucket` 的 Amazon S3 存储桶。

示例请求

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

示例响应

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LL171SPPtSnBvDdGmnk1X1HfU17uS2m1DF6td6KWKNjYMXZ
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Etag: "1f1a1af1f1111111111111111c11aed1da1"
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
```

相关资源

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

附录：SOAP API

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止（EOL）。我们建议您使用 REST API 或 AWS SDK。

本节介绍与服务、存储桶和对象操作有关的 SOAP API。请注意，必须使用 SSL 将经身份验证的和匿名的 SOAP 请求发送到 Amazon S3。若您通过 HTTP 发送 SOAP 请求，Amazon S3 将返回错误。

可通过 docs.aws.amazon.com/2006-03-01/AmazonS3.wSDL 获得最新的 Amazon S3 WSDL。

主题

- [服务上的操作 \(SOAP API \)](#)
- [对存储桶的操作 \(SOAP API \)](#)
- [针对对象的操作 \(SOAP API \)](#)
- [对 SOAP 请求进行身份验证](#)

- [使用 SOAP 设置访问策略](#)
- [常见元素](#)
- [SOAP 错误响应](#)

服务上的操作 (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

本节介绍您可以在 Amazon S3 服务上执行的操作。

主题

- [ListAllMyBuckets \(SOAP API \)](#)

ListAllMyBuckets (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

ListAllMyBuckets 操作返回请求发送者所拥有的所有存储桶的列表。

Example

Sample Request

```
<ListAllMyBuckets xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```


Sample Response

```
<ListAllMyBucketsResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1fffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

响应正文

- Owner:

这提供了 Amazon S3 用于表示您的身份的信息，以用于身份验证和访问控制目的。ID 是发出请求的开发人员的唯一且永久的标识符。DisplayName 是一个便于阅读的名称，表示发出请求的开发人员。它不是唯一的，可能会随着时间推移而发生变化。我们建议您将 DisplayName 与论坛名称相匹配。

- Name:

存储桶的名称。请注意，如果您的某个存储桶最近被删除，则已删除存储桶的名称可能在一段时间内仍会出现在此列表中。

- CreationDate:

创建存储桶的时间。

访问控制

您必须使用有效的 AWS 访问密钥 ID 进行身份验证。从不支持匿名请求列出存储桶，您只能列出您是其拥有者的存储桶。

对存储桶的操作 (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

本节介绍您可以对 Amazon S3 存储桶执行的操作。

主题

- [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

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

CreateBucket 操作创建一个存储桶。并非每个字符串都是可接受的存储桶名称。有关存储桶命名限制的信息，请参阅[使用 Amazon S3 存储桶](#)。

Note

要确定存储桶名称是否存在，请使用 ListBucket 并将 MaxKeys 设置为 0。NoSuchBucket 响应表示该存储桶可用，AccessDenied 响应表示其他人拥有该存储桶，Success 响应表示您拥有该存储桶或有权访问该存储桶。

Example 创建一个名为“quotes”的存储桶

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>
```

元素

- **Bucket**：您尝试创建的存储桶的名称。
- **AccessControlList**：新存储桶的访问控制列表。此元素是可选的。如果未提供，则使用向请求者授予 FULL_CONTROL 访问权限的访问策略创建存储桶。

访问控制

您必须使用有效的 AWS 访问密钥 ID 进行身份验证。从不支持匿名请求创建存储桶。

相关资源

- [ListBucket \(SOAP API \)](#)

DeleteBucket (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

DeleteBucket 操作会删除存储桶。必须先删除存储桶中的所有对象，然后才能删除存储桶本身。

Example

此示例删除“quotes”存储桶。

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>
```

元素

- **Bucket**：您要删除的存储桶的名称。

访问控制

无论存储桶的访问控制策略如何，都只支持存储桶的拥有者将其删除。

ListBucket (SOAP API)

 Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止（EOL）。我们建议您使用 REST API 或 AWS SDK。

ListBucket 操作返回有关存储桶中某些项目的信息。

有关列表操作的一般介绍，请参阅[列出对象键](#)。

请求

此示例列出了“quotes”存储桶中最多 1000 个带有前缀“notes”的键。

语法

```
<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>
```

参数

名称	描述	必填
prefix	将响应限制为以指定的前缀开头的键。您可以使用前缀将存储桶分成不同的键集，其方式与文件系统使用文件夹的方式类似。 类型：字符串 默认值：无	否

名称	描述	必填
marker	指明要在存储桶中的哪个位置开始列出。该列表将仅包括按字典顺序出现在标记之后的键。这便于分页：要获取下一页的结果，请使用当前页面的最后一个键作为标记。 类型：字符串 默认值：无	否
max-keys	您希望在响应正文中看到的最大键数。服务器返回的键数可能少于这个数量，但返回的键数不会多于此数量。 类型：字符串 默认值：无	否
delimiter	导致在前缀和首次出现的分隔符之间包含相同字符串的键汇总到 CommonPrefixes 集合的单个结果元素中。这些汇总的键不会在响应中的其它地方返回。 类型：字符串 默认值：无	否

成功响应

此响应假设存储桶包含以下键：

```
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
```

语法

```
<?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>
```

如您所见，响应中的许多字段都与请求参数相呼应。IsTruncated、Contents 和 CommonPrefixes 是仅有的可以包含新信息的响应元素。

响应元素

名称	描述
Contents	有关返回的每个对象的元数据。 类型：XML 元数据 原级：ListBucketResult
CommonPrefixes	仅当您指定 delimiter 时，响应才会包含 CommonPrefixes 。当您这样做时，CommonPrefixes 包含 Prefix 与由 delimiter 指定的字符串的下一次出现之间的所有（如果有的话）键。实际上，CommonPrefixes 会列出在 Prefix 指定的目录中充当子目录的键。例如，如果 prefix 是 notes/，delimiter 是斜杠 (/)，则在 notes/summer/july 中，常见前缀是 notes/summer/ 。

名称	描述
Delimiter	类型：字符串
	原级：ListBucketResult
	导致在前缀和首次出现的分隔符之间包含相同字符串的键汇总到 CommonPrefixes 集合的单个结果元素中。这些汇总的键不会在响应中的其它地方返回。 类型：字符串 原级：ListBucketResult
IsTruncated	指定是 (true) 否 (false) 返回了所有结果。如果结果数超过 MaxKeys 指定的数量，则可能无法返回所有结果。
	类型：字符串
	原级：布尔值
Marker	指明要在存储桶中的哪个位置开始列出。
	类型：字符串
	原级：ListBucketResult
MaxKeys	要在响应正文中返回的最大键数。
	类型：字符串
	原级：ListBucketResult
Name	存储桶的名称。
	类型：字符串
	原级：ListBucketResult
Prefix	以指定的前缀开头的键。
	类型：字符串
	原级：ListBucketResult

响应正文

有关列出响应的信息，请参阅[列出键响应](#)。

访问控制

要列出存储桶的键，您需要获得对该存储桶的 READ 访问权限。

GetBucketAccessControlPolicy (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

GetBucketAccessControlPolicy 操作提取存储桶的访问控制策略。

Example

此示例检索“quotes”存储桶的访问控制策略。

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>a9a7b886d6f41bf9b1c61be666e9</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>
```

响应正文

响应包含存储桶的访问控制策略。有关此响应的说明，请参阅 [SOAP 访问策略](#)。

访问控制

您必须对存储桶拥有 READ_ACP 权限，才能检索存储桶的访问控制策略。

SetBucketAccessControlPolicy (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

SetBucketAccessControlPolicy 操作为现有存储桶设置访问控制策略。如果成功，则存储桶之前的访问控制策略将完全替换为指定的访问控制策略。

Example

向指定的用户 (通常是拥有者) 授予对“quotes”存储桶的 FULL_CONTROL 访问权限。

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>
```

访问控制

您必须对存储桶拥有 WRITE_ACP 权限，才能设置存储桶的访问控制策略。

GetBucketLoggingStatus (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止（EOL）。我们建议您使用 REST API 或 AWS SDK。

GetBucketLoggingStatus 检索现有存储桶的日志记录状态。

有关此功能的一般介绍，请参阅[服务器日志](#)。

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>
```

访问控制

仅允许存储桶的拥有者调用此操作。

SetBucketLoggingStatus (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

SetBucketLoggingStatus 操作更新现有存储桶的日志记录状态。

有关此功能的一般介绍，请参阅[服务器日志](#)。

Example

此示例请求对“mybucket”存储桶启用服务器访问日志记录，并将日志配置为传输到前缀“access_log-”下的“mylogs”

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>
```

访问控制

仅允许存储桶的拥有者调用此操作。

针对对象的操作 (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

本节介绍您可以对 Amazon S3 对象执行的操作。

主题

- [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

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

PutObjectInline 操作将对象添加到存储桶。该对象的数据在 SOAP 消息的正文中提供。

如果存储桶中已存在对象，则新对象将覆盖该对象，因为 Amazon S3 存储了上次写入请求。但是，Amazon S3 是一个分布式系统。如果 Amazon S3 几乎同时收到针对同一对象的多个写入请求，则可能会存储所有对象，尽管最终只有一个请求胜出。Amazon S3 不提供对象锁定；如果您需要此功能，请务必将其内置到您的应用程序层中。

为了确保对象在通过网络时不会损坏，您可以计算对象的 MD5，将其 PUT 到 Amazon S3，并将返回的 Etag 与计算的 MD5 值进行比较。

PutObjectInline 不适合用于大型对象。系统将此操作限制为处理 1 MB 或更小的对象。如果 Data 参数对大于 1 MB 的对象进行编码，PutObjectInline 将失败并显示 InlineDataTooLargeError 状态代码。要上传大型对象，可以考虑使用非内联 PutObject API，或者改用 REST API。

Example

此示例将一些文本和元数据写入“quotes”存储桶中的“Nelson”对象，授予用户（通常是拥有者）对该对象的 FULL_CONTROL 访问权限，并使匿名方可以读取该对象。

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
```

```

<Data>aGEtaGE=</Data>
<ContentLength>5</ContentLength>
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>

```

Sample Response

```

<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>

```

元素

- **Bucket**: 要在其中添加对象的存储桶。
- **Key**: 要分配给对象的键。
- **Metadata**: 您可以在元数据元素中提供名称-值元数据对。这些将与对象一起存储。
- **Data**: 数据的 base 64 编码形式。
- **ContentLength**: 数据的长度，以字节为单位。

- **AccessControlList**: 资源的访问控制列表。此元素是可选的。如果忽略，则向请求者授予对该对象的 **FULL_CONTROL** 访问权限。如果对象已存在，则将替换先前存在的访问控制策略。

响应

- **ETag**: 实体标签是对象的 MD5 哈希值，您可以使用它通过 **GetObjectExtended** 对于对象进行有条件提取。ETag 仅反映对于对象内容的更改，而不反映对其元数据的更改。
- **LastModified**: 已保存对象的 Amazon S3 时间戳。

访问控制

要将对象放入存储桶，您必须对存储桶拥有 **WRITE** 访问权限。

相关资源

- [PutObject \(SOAP API \)](#)
- [CopyObject \(SOAP API \)](#)

PutObject (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

PutObject 操作将对象添加到存储桶。该对象的数据以 DIME 附件的形式附加。

为了确保对象在通过网络时不会损坏，您可以计算对象的 MD5，将其 PUT 到 Amazon S3，并将返回的 Etag 与计算的 MD5 值进行比较。

如果存储桶中已存在对象，则新对象将覆盖该对象，因为 Amazon S3 存储了上次写入请求。但是，Amazon S3 是一个分布式系统。如果 Amazon S3 几乎同时收到针对同一对象的多个写入请求，则可能会存储所有对象，尽管最终只有一个请求胜出。Amazon S3 不提供对象锁定；如果您需要此功能，请务必将其内置到您的应用程序层中。

Example

此示例将一些数据和元数据放入“quotes”存储桶的“Nelson”对象中，授予用户（通常是拥有者）对该对象的 FULL_CONTROL 访问权限，并使匿名方可以读取该对象。在此示例中，未显示实际附件。

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>"828ef3fd9a96f00ad9f27c383fc9ac7f"</ETag>
<LastModified>2006-03-01T12:00:00.183Z</LastModified>
</PutObjectResponse>
</PutObjectResponse>
```

元素

- **Bucket**: 要在其中添加对象的存储桶。
- **Key**: 要分配给对象的键。
- **Metadata**: 您可以在元数据元素中提供名称-值元数据对。这些将与对象一起存储。
- **ContentLength**: 数据的长度，以字节为单位。
- **AccessControlList**: 资源的访问控制列表。此元素是可选的。如果忽略，则向请求者授予对该对象的 FULL_CONTROL 访问权限。如果对象已存在，则将替换先前存在的访问控制策略。

响应

- **ETag**: 实体标签是对象的 MD5 哈希值，您可以使用它通过 GetObjectExtended 对于对象进行有条件提取。ETag 仅反映对于对象内容的更改，而不反映对其元数据的更改。
- **LastModified**: 已保存对象的 Amazon S3 时间戳。

访问控制

要将对象放入存储桶，您必须对存储桶拥有 WRITE 访问权限。

相关资源

- [CopyObject \(SOAP API \)](#)

CopyObject (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

说明

当您指定源对象的键和存储桶以及目标目的地的键和存储桶时，CopyObject 操作创建对象的副本。

复制对象时，您可以保留所有元数据（默认）或指定新的元数据。但是，ACL 不予以保留，而是针对发出请求的用户设置为 private。要覆盖默认 ACL 设置，请在生成复制请求时指定新的 ACL。有关更多信息，请参阅[使用 ACL](#)。

所有复制请求都必须进行身份验证。此外，您必须对源对象拥有读取访问权限，并对目标存储桶拥有写入访问权限。有关更多信息，请参阅[使用身份验证访问](#)。

要仅在某些条件下才复制对象，例如 Etag 是否匹配或者对象是在指定日期之前还是之后修改的，请使用请求参数

CopySourceIfUnmodifiedSince、CopyIfUnmodifiedSince、CopySourceIfMatch 或 CopySourceIfNoneMatch。

Note

您可能需要为复制大型对象配置 SOAP 堆栈套接字超时。

请求语法

```
<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>
</CopyObject>
```

```
</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>
```

请求参数

名称	描述	必填
SourceBucket	源存储桶的名称。 类型：字符串 默认值：无 约束：有效的源存储桶。	是
SourceKey	源对象的键名称。 类型：字符串 默认值：无 约束：您拥有 READ 访问权限的有效源对象的键。	是
DestinationBucket	目标 存储桶的名称。 类型：字符串 默认值：无 约束：您必须对目标存储桶拥有 WRITE 权限。	是

名称	描述	必填
DestinationKey	目标对象的键。 类型：字符串 默认值：无 约束：您必须对目标存储桶拥有 WRITE 权限。	是
MetadataDirective	指定是从源对象复制元数据，还是用请求中提供的元数据替换元数据。 类型：字符串 默认：COPY 有效值：COPY REPLACE 约束：除 COPY 或 REPLACE 以外的值将立即导致错误。除非指定 MetadataDirective 标头且其值设置为 REPLACE，否则您无法将对象复制到其自身。	否
Metadata	指定要为对象设置的元数据名称-值对。如果 MetadataDirective 设置为 COPY，则忽略所有元数据。 类型：字符串 默认值：无 约束：无。	否

名称	描述	必填
AccessControlList	按电子邮件地址或规范用户 ID 向用户授予访问权限。 类型：字符串 默认值：无 约束：无	否
CopySourceIfMatch	如果对象的实体标签 (ETag) 与指定的标签匹配，则复制该对象；否则返回 PreconditionFailed。 类型：字符串 默认值：无 约束：无。如果 Etag 不匹配，则不会复制对象。	否
CopySourceIfNoneMatch	如果对象的实体标签 (ETag) 与指定的 Etag 不同，则复制该对象；否则返回错误。 类型：字符串 默认值：无 约束：无。	否
CopySourceIfUnmodifiedSince	如果自指定的时间以来未修改过对象，则复制该对象；否则返回 PreconditionFailed。 类型：dateTime 默认值：无	否

名称	描述	必填
CopySourceIfModifiedSince	如果对象自指定的时间以来已被修改，则复制该对象；否则返回错误。 类型：dateTime 默认值：无	否

响应语法

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

响应元素

以下是响应元素的列表。

 **Note**
SOAP API 不返回额外的空格。额外的空格只会由 REST API 返回。

名称	描述
Etag	返回新对象的 ETag。ETag 仅反映对于对象内容的更改，而不反映对其元数据的更改。 类型：字符串 原级：CopyObjectResult
LastModified	返回上次修改对象的日期。 类型：字符串

名称	描述
	原级：CopyObjectResult

有关常规响应元素的信息，请参阅[使用 REST 错误响应标头](#)。

特殊错误

此操作没有特殊错误。有关一般 Amazon S3 错误的信息，请参阅[错误代码列表](#)。

示例

本示例将 flotsam 存储桶中的 pacific 对象复制到 jetsam 存储桶的 atlantic 对象，同时保留其元数据。

示例请求

```
<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>
```

示例响应

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

此示例将“tweedledee”对象从 wonderland 存储桶复制到 wonderland 存储桶的“tweedledum”对象，同时替换其元数据。

示例请求

```
<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>
```

示例响应

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

相关资源

- [PutObject \(SOAP API \)](#)
- [PutObjectInline \(SOAP API \)](#)

GetObject (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

GetObject 操作返回对象的当前版本。如果您尝试对以删除标记作为其当前版本的对象执行 GetObject，S3 将返回 404 错误。您不能使用 SOAP API 来检索对象的指定版本。要执行这一操作，请使用 REST API。有关更多信息，请参阅[版本控制](#)。有关更多选项，请使用 [GetObjectExtended \(SOAP API \)](#) 操作。

Note

对于[虚拟托管类型请求](#)，不支持值为“soap”的对象键名称。对于使用“soap”的对象键名称值，必须改用[路径类型 URL](#)。

Example

本示例从“quotes”存储桶获取“Nelson”对象。

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>
```

元素

- **Bucket**：要从中检索对象的存储桶。
- **Key**：用于标识对象的键。
- **GetMetadata**：如果为 `true`，则元数据将随对象一起返回。
- **GetData**：如果为 `true`，则返回对象数据。
- **InlineData**：如果为 `true`，则以 base 64 编码的数据将作为响应的 SOAP 正文的一部分返回。如果为 `false`，则数据作为 SOAP 附件返回。InlineData 选项不适合用于大型对象。系统将此操作限制为处理 1 MB 或更小的数据。如果生成的 Data 参数编码后的大小超过 1 MB，则设置了 InlineData 标志的 GetObject 请求将失败并显示 `InlineDataTooLargeError` 状态代码。要下载大型对象，请考虑在不设置 InlineData 标志的情况下调用 GetObject，或者改用 REST API。

返回的元素

- **Metadata**：随对象一起存储的名称-值对元数据。
- **Data**：如果请求中 InlineData 为 `true`，则它包含以 base 64 编码的对象数据。
- **LastModified**：在 Amazon S3 中存储对象的时间。
- **ETag**：对象的实体标签。这是对象的哈希值，可用于执行有条件获取。ETag 仅反映对于对象内容的更改，而不反映对其元数据的更改。

访问控制

仅当被授予对象的 READ 访问权限后，您才能读取该对象。

SOAP 分块下载和可恢复下载

为了提供 GET 灵活性，Amazon S3 支持分块下载和可恢复下载。

从以下各项中进行选择：

- 对于大型对象下载，您可能希望将其分成较小的分块。有关更多信息，请参阅 [Range GET](#)。
- 对于失败的 GET 操作，您可以将应用程序设计为下载剩余内容而不是整个文件。有关更多信息，请参阅 [REST GET 错误恢复](#)。

Range GET

对于某些客户端，您可能希望将大型下载分解为若干较小的下载。要将 GET 分解成较小的单位，请使用 Range。

在将 GET 分解为较小的单位之前，必须确定其大小。例如，以下请求获取大文件对象的大小。

```
<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 返回以下响应。

```
<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>
```

以下是从大文件对象下载第一个兆字节的请求。

```
<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 返回文件的第一个兆字节和文件的 Etag。

```
<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>
```

要确保文件在下载前面的部分后不会更改，请指定 IfMatch 元素。尽管 IfMatch 元素不是必需的，但对于可能发生变化的内容，建议使用该元素。

以下是使用 IfMatch 请求标头获取文件剩余部分的请求。

```
<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 将返回以下响应和文件的剩余部分。

```

<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>

```

受版本控制的 GetObject

以下请求返回存储桶中对象的指定版本。

```

<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>
```

示例响应

```
<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>
```

REST GET 错误恢复

如果对象 GET 失败，则可以通过指定要下载的范围来获取文件的剩余部分。为此，必须使用 `ListBucket` 获取对象的大小，并对文件的剩余部分执行 Range GET。有关更多信息，请参阅 [GetObjectExtended \(SOAP API \)](#)。

相关资源

[针对对象的操作 \(SOAP API \)](#)

GetObjectExtended (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

GetObjectExtended 与 [GetObject \(SOAP API \)](#) 完全相同，区别在于它支持以下额外元素，可用于实现 HTTP GET 标头提供的许多相同功能 (请访问 <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>)。

GetObjectExtended 除了 GetObject 支持的那些元素以外，还支持以下元素：

- ByteRangeStart, ByteRangeEnd: 这些元素指定只应检索对象数据的一部分。它们遵循 HTTP 字节范围的行为 (请访问 <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35>)。
- IfModifiedSince: 仅当对象的时间戳晚于指定的时间戳时才返回该对象。(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- IfUnmodifiedSince: 仅当对象的时间戳早于或等于指定的时间戳时才返回该对象。(请访问 <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28>)
- IfMatch: 仅当对象的 ETag 与提供的标签匹配时，才返回该对象。(请访问 <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24>)
- IfNoneMatch: 仅当对象的 ETag 与提供的标签不匹配时，才返回该对象。(请访问 <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26>)
- ReturnCompleteObjectOnConditionFailure: 如果为 true，则当请求包含一个范围元素以及一个或两个 IfUnmodifiedSince/IfMatch 元素且条件失败时，返回整个对象而不是故障。这将启用 If-Range 功能 (请访问 <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27>)。

DeleteObject (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

DeleteObject 操作从 Amazon S3 中移除指定的对象。删除对象后，就无法还原或取消删除该对象。

Note

如果您删除不存在的对象，Amazon S3 将返回成功（而不是错误消息）。

Example

本示例从“quotes”存储桶删除“Nelson”对象。

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>
```

元素

- Bucket：包含对象的存储桶。
- Key：用于标识对象的键。

访问控制

仅当您对存储桶拥有 WRITE 访问权限时，才能删除对象，而无论该对象归属于谁或被授予了何种权限。

GetObjectAccessControlPolicy (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

GetObjectAccessControlPolicy 操作提取对象的访问控制策略。

Example

本示例将为“quotes”存储桶中的“Nelson”对象检索访问控制策略。

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>
```

响应正文

响应包含存储桶的访问控制策略。有关此响应的解释，请参阅 [SOAP 访问策略](#)。

访问控制

您必须拥有对象的 READ_ACP 权限，才能检索该对象的访问控制策略。

SetObjectAccessControlPolicy (SOAP API)

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS SDK。

SetObjectAccessControlPolicy 操作为现有对象设置访问控制策略。如果成功，该对象先前的访问控制策略将完全替换为指定的访问控制策略。

Example

本示例向指定的用户（通常是拥有者）授予对“quotes”存储桶中“Nelson”对象的 FULL_CONTROL 访问权限。

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

访问控制

您必须对该对象拥有 WRITE_ACP 权限才能设置存储桶的访问控制策略。

对 SOAP 请求进行身份验证

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止（EOL）。我们建议您使用 REST API 或 AWS 开发工具包。

所有非匿名的请求都必须包含身份验证信息，以便确立提出请求的委托人的身份。在 SOAP 中，身份验证信息放置在 SOAP 请求的以下元素中：

- 您的 AWS 访问密钥 ID

Note

创建经身份验证的 SOAP 请求时，不支持临时的安全凭证。有关凭证类型的更多信息，请参阅 [提出请求](#)。

- **Timestamp:** 它必须采用协调世界时 (Greenwich Mean Time) 时区的日期时间 (转到 <http://www.w3.org/TR/xmlschema-2/#dateTime>)，例如 2009-01-01T12:00:00.000Z。如果这个时间戳与 Amazon S3 服务器上的时间相差 15 分钟以上，授权将失败。
- **Signature:** “AmazonS3” + OPERATION + Timestamp 串联的 RFC 2104 HMAC-SHA1 摘要 (转至 <http://www.ietf.org/rfc/rfc2104.txt>) 将您的 AWS 秘密访问密钥用作密钥。例如，在下面的 CreateBucket 示例请求中，签名元素将包含“AmazonS3CreateBucket2009-01-01T12:00:00.000Z”值的 HMAC-SHA1 摘要：

例如，在下面的 CreateBucket 示例请求中，签名元素将包含“AmazonS3CreateBucket2009-01-01T12:00:00.000Z”值的 HMAC-SHA1 摘要：

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

必须使用 SSL 将经身份验证的和匿名的 SOAP 请求发送到 Amazon S3。若您通过 HTTP 发送 SOAP 请求，Amazon S3 将返回错误。

Important

由于对如何丢弃额外的时间精度存在不同的解释，.NET 用户应注意不要使用过于具体的时间戳来发送 Amazon S3。可以通过手动构建只有毫秒精度的 DateTime 对象完成此操作。

使用 SOAP 设置访问策略

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS 开发工具包。

在编写存储桶或对象时，可以通过将“AccessControlList”元素随附在 CreateBucket、PutObjectInline 或 PutObject 的请求中，设置访问控制。[Amazon S3 的身份和访问管理](#)中描述了 AccessControlList 元素。如果没有使用这些操作指定访问控制列表，使用默认访问策略创建的资源将给予请求者 FULL_CONTROL 访问权限 (即使请求是用于已存在的对象的 PutObjectInline 或 PutObject 请求)。

下面是向对象写入数据、允许匿名委托人读取对象，以及授予特定用户对存储桶的 FULL_CONTROL 权限 (大多数开发人员希望授予他们自己对存储桶的 FULL_CONTROL 访问权限) 的请求。

Example

下面是向对象写入数据并允许匿名委托人读取对象的请求。

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
```

```
<Grantee xsi:type="Group">
  <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
</Grantee>
<Permission>READ</Permission>
</Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

可以使用

`GetBucketAccessControlPolicy`、`GetObjectAccessControlPolicy`、`SetBucketAccessControlPolicy` 和 `SetObjectAccessControlPolicy` 方法为现有存储桶或对象读取或设置访问控制策略。有关详细信息，请参阅这些方法的详细说明。

常见元素

您可以包含以下具有任何 SOAP 请求的与授权相关的元素：

- `AWSAccessKeyId`: 请求者的 AWS 访问密钥 ID
- `Timestamp`: 系统上的当前时间
- `Signature`: 用于该请求的签名

SOAP 错误响应

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止 (EOL)。我们建议您使用 REST API 或 AWS 开发工具包。

在 SOAP 中，错误结果将作为 SOAP 错误 (使用 HTTP 响应代码 500) 返回到客户端。若您没有收到 SOAP 错误，则您的请求已成功。Amazon S3 SOAP 错误代码是由标准 SOAP 1.1 错误代码 (“Server”或“Client”) 与特定于 Amazon S3 的错误代码相连接而组成的。例如，“Server.InternalError”或“Client.NoSuchBucket”。SOAP 错误字符串元素包括一个通用的、用户可读的错误消息 (英语)。最后，SOAP 错误详细信息元素将包括与错误相关的其他信息。

例如，如果您尝试删除对象“Fred”，但该对象不存在，则 SOAP 响应的正文将包含“NoSuchKey”SOAP 错误。

下面的示例介绍一个示例 SOAP 错误响应。

```
<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>
```

下表说明了 SOAP 错误响应元素

名称	描述
Detail	错误中涉及的键的容器 类型：容器 原级：Body.Fault
Fault	错误信息的容器。 类型：容器 原级：Body
Faultcode	故障代码是用于唯一标识错误条件的字符串。这意味着按类型检测和处理错误的程序将读取和理解错误代码。有关更多信息，请参阅 错误代码列表 。 类型：字符串

名称	描述
	原级：Body.Fault
Faultstring	故障字符串包含错误条件的一般描述（英语）。它主要面向用户受众。如果用户遇到不知如何处理或不愿处理的错误条件，简单的程序将直接向最终用户显示消息。支持更详尽的错误处理和适当国际化的复杂程序更可能忽略故障字符串。 类型：字符串 原级：Body.Fault
Key	标识错误中涉及的键 类型：字符串 原级：Body.Fault

附录：对请求进行身份验证（AWS 版本签名 2）

Important

此部分介绍如何使用 AWS Signature Version 2 对请求进行身份验证。Signature Version 2 会关闭（弃用），Amazon S3 将仅接受使用 Signature Version 4 进行签名的 API 请求。有关更多信息，请参阅 [已为 Amazon S3 关闭（弃用）AWS 签名版本 2](#)。

所有 AWS 区域都支持 Signature Version 4，这是新区域唯一支持的版本。有关更多信息，请参阅《Amazon Simple Storage Service API》参考中的[验证请求 \(AWS Signature Version 4\)](#)。

Amazon S3 使您能够确定用于签署请求的 API 签名版本。请务必确定您的任何工作流是否正在使用 Signature Version 2 签名并将其升级到使用 Signature Version 4 以防影响您的业务。

- 如果您使用的是 CloudTrail 事件日志（推荐选项），请参阅[使用 CloudTrail 识别 Amazon S3 签名版本 2 请求](#)，以了解如何查询和识别此类请求。
- 如果您使用的是 Amazon S3 服务器访问日志，请参阅[使用 Amazon S3 服务器访问日志来确定请求](#)

主题

- [使用 REST API 对请求进行身份验证 \(AWS 版本签名 2 \)](#)
- [对 REST 请求进行签名和身份验证 \(AWS 签名版本 2 \)](#)
- [使用 POST \(AWS Signature Version 2\) 的基于浏览器的上传](#)

使用 REST API 对请求进行身份验证 (AWS 版本签名 2)

使用 REST 访问 Amazon S3 时，您必须在请求中提供以下项目，以便对请求进行身份验证：

请求元素

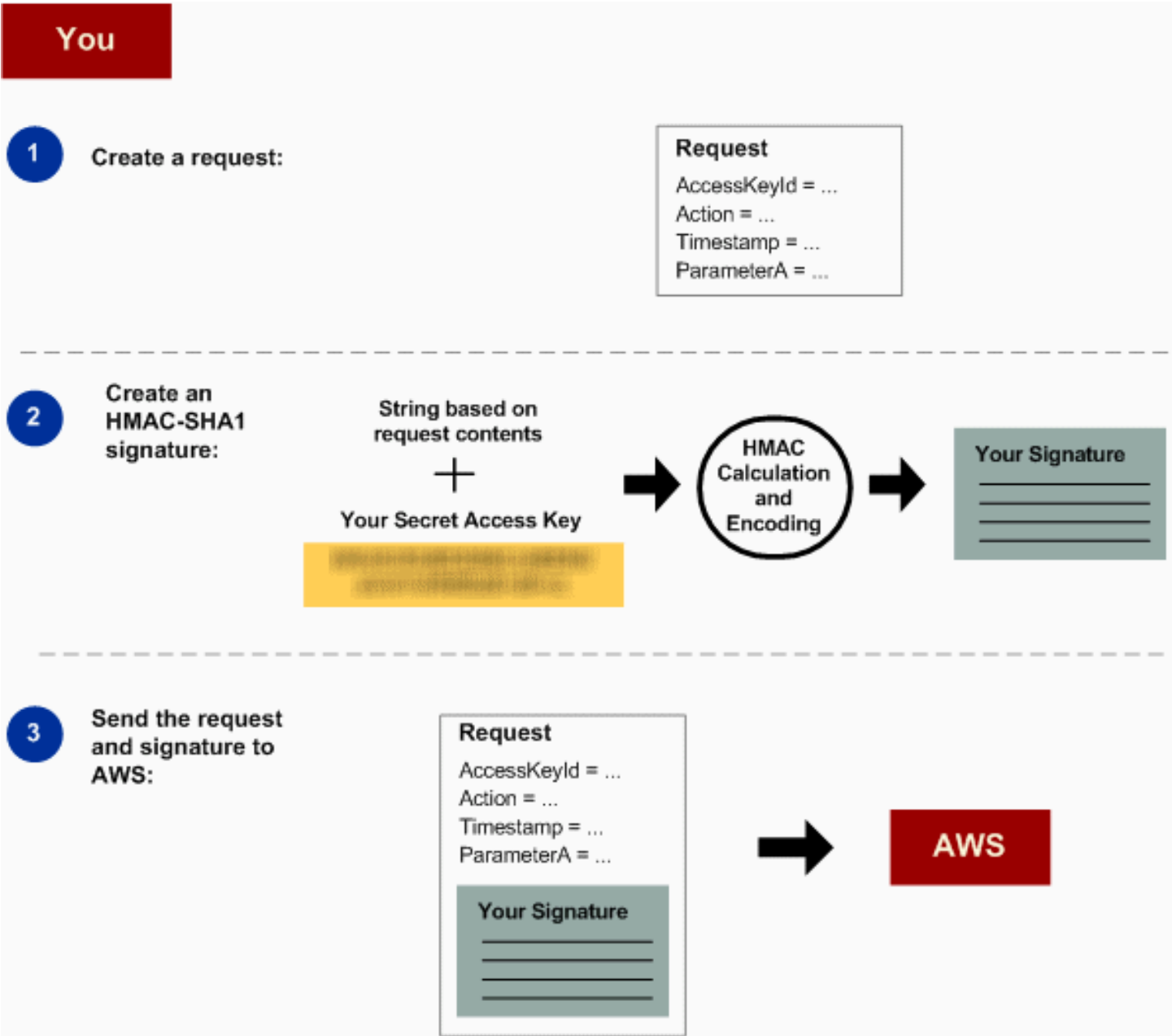
- AWS 访问密钥 ID – 每个请求必须包含用于发送请求的身份的访问密钥 ID。
- 签名 - 每个请求必须包含一个有效的请求签名，否则系统会拒绝请求。

将使用您的秘密访问密钥计算出请求签名，该密钥是一个共享密钥，相关知晓方只有您和 AWS。

- 时间戳 – 每个请求必须包含请求的创建日期和时间，并表示为 UTC 形式的字符串。
- 日期 – 每个请求都必须包含请求的时间戳。

根据您正在使用的 API 操作，您可以不提供时间戳，改为提供请求的过期日期和时间，或者也可以二者都提供。要确定特定操作需要的内容，请参阅该操作的身份验证主题。

下面是针对发送到 Amazon S3 的请求进行身份验证的常规步骤。此处假设您已拥有了必要的安全凭证、访问密钥 ID 和秘密访问密钥。

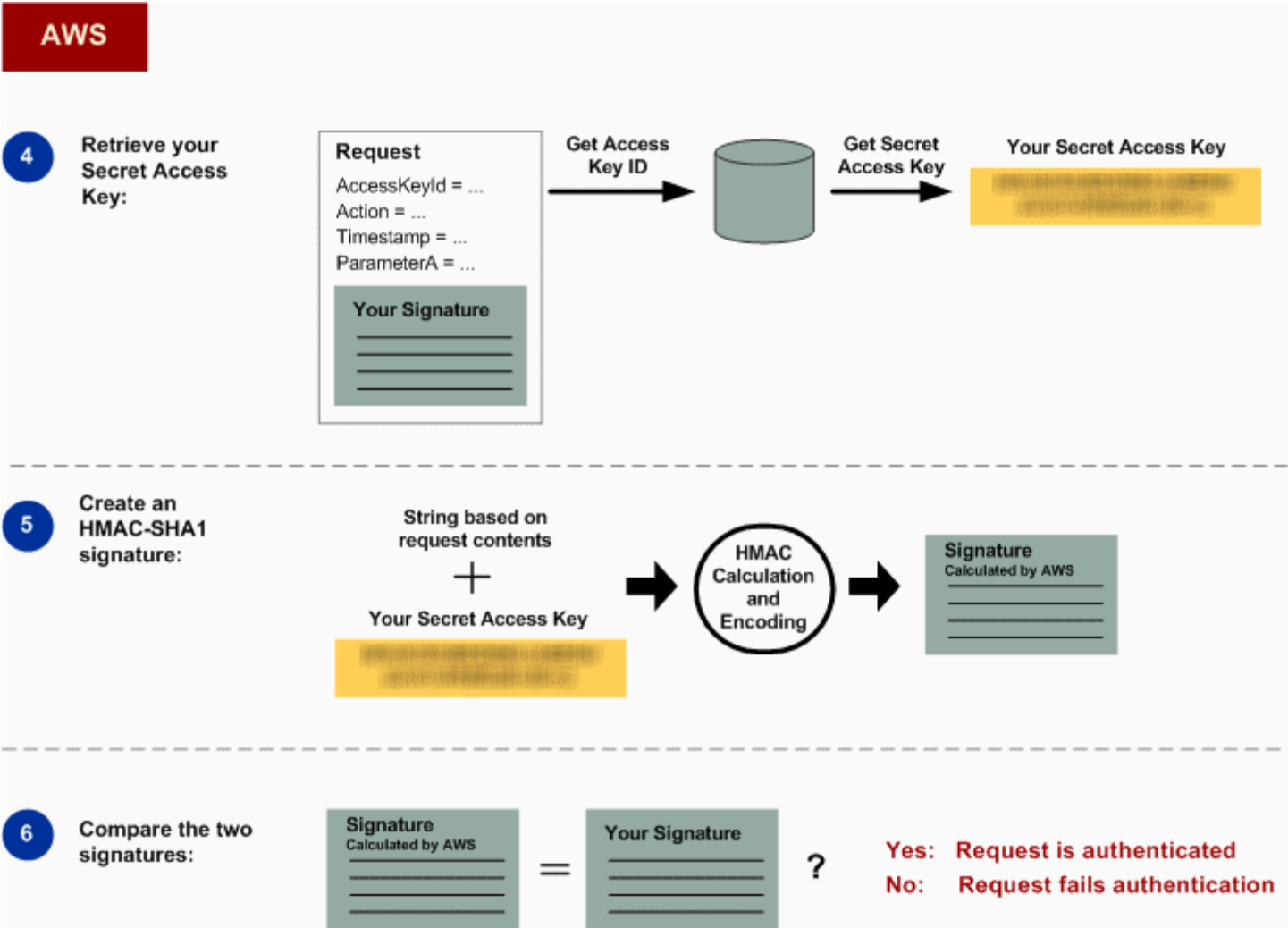


- 1

构建要发送到 AWS 的请求。
- 2

使用您的秘密访问密钥计算签名。
- 3

向 Amazon S3 发送请求。在请求中包含您的访问密钥 ID 和签名。Amazon S3 执行下面三个步骤。



对 REST 请求进行签名和身份验证 (AWS 签名版本 2)

主题

- [使用临时安全凭证](#)
- [身份验证标头](#)
- [用于签名的请求规范](#)
- [构建 CanonicalizedResource 元素](#)
- [构建 CanonicalizedAmzHeaders 元素](#)
- [位置和命名 HTTP 标头 StringToSign 元素的对比](#)
- [时间戳要求](#)
- [身份验证示例](#)
- [REST 请求签名问题](#)
- [查询字符串请求身份验证替代项](#)

Note

本主题说明使用签名版本 2 对请求进行身份验证。Amazon S3 现在支持最新的 Signature Version 4。所有区域都支持此最新签名版本，而 2014 年 1 月 30 日之后的所有新区域都只支持签名版本 4。有关更多信息，请转至《Amazon Simple Storage Service API 参考》中的[验证请求 \(AWS Signature Version 4\)](#)。

身份验证是指向系统证明身份的过程。身份对于 Amazon S3 访问控制决策非常重要。将根据请求者的身份允许请求或拒绝部分请求。例如，将为已注册的开发人员保留创建存储桶的权利，并且 (默认) 将为相关的存储桶所有者保留在存储桶中创建对象的权利。作为开发人员，您需要提出请求来调用这些权限，因此您需要对您的请求进行身份验证以向系统证明您的身份。本节向您演示了应如何进行操作。

Note

本节中的内容不适用于 HTTP POST。有关更多信息，请参阅 [使用 POST \(AWS Signature Version 2\) 的基于浏览器的上传](#)。

Amazon S3 REST API 使用基于密钥 HMAC (Hash Message Authentication Code) 的自定义 HTTP 方案进行身份验证。要对请求进行身份验证，您首先需要合并请求的选定元素以形成一个字符串。然后，

您可以使用 AWS 秘密访问密钥来计算该字符串的 HMAC。通常我们将此过程称为“签署请求”并且我们将输出 HMAC 算法称为“签名”，因为它会模拟真实签名的安全属性。最后，您可以使用本部分中介绍的语法，作为请求的参数添加此签名。

系统收到经身份验证的请求时，将提取您申领的 AWS 秘密访问密钥，并以相同的使用方式将它用于计算已收到的消息的签名。然后，它会将计算出的签名与请求者提供的签名进行对比。如果两个签名相匹配，则系统认为请求者必须拥有对 AWS 秘密访问密钥的访问权限，因此充当向其颁发密钥的委托人的颁发机构。如果两个签名不匹配，那么请求将被丢弃，同时系统将返回错误消息。

Example 经身份验证的 Amazon S3 REST 请求

```
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=
```

使用临时安全凭证

如果您使用临时安全凭证签署请求 (参阅 [提出请求](#))，您必须通过添加 x-amz-security-token 标头将相应的安全令牌包含在您的请求中。

使用 AWS Security Token Service API 获取临时安全凭证时，响应包含临时安全凭证和会话令牌。向 Amazon S3 发送请求时，您可以在 x-amz-security-token 标头中提供会话令牌值。有关 IAM 提供的 AWS Security Token Service API 的信息，请转到《AWS Security Token Service API 参考指南》中的[操作](#)。

身份验证标头

Amazon S3 REST API 使用标准的 HTTP Authorization 标头来传递身份验证信息。（标准标头的名称是不可取的，因为它承载的是身份验证信息，而不是授权。）在 Amazon S3 身份验证方案下，授权标头具有以下形式：

```
Authorization: AWS AWSAccessKeyId:Signature
```

注册后，会向开发人员发放 AWS 访问密钥 ID 和 AWS 秘密访问密钥。对于请求身份验证，AWSAccessKeyId 元素将标识用于计算签名的访问密钥 ID 和进行请求的开发人员 (间接)。

Signature 元素是请求中选定元素的 RFC 2104HMAC-SHA1，因此授权标头的 Signature 部分会因请求的不同而各异。如果系统计算出的请求签名与请求随附的 Signature 相匹配，则请求者拥有经证明的 AWS 秘密访问密钥。然后，在该身份下，借助获得此密钥的开发人员的授权来处理请求。

以下是说明 Authorization 请求标头结构的伪语法。(在该示例中，\n 表示 Unicode 码位 U+000A，这通常称为换行符)。

```
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 是由 [RFC 2104 \(用于消息身份验证的哈希密钥\)](#) 定义的算法。该算法要求输入两个字节字符串：一个密钥和一个消息。对于 Amazon S3 请求身份验证，请将您的 AWS 秘密访问密钥 (YourSecretAccessKey) 用作密钥，并将 UTF-8 编码格式的 StringToSign 用作消息。HMAC-SHA1 的输出也是字节字符串，称为摘要。通过对此摘要进行 Base64 编码来构成 Signature 请求参数。

用于签名的请求规范

在系统收到经身份验证的请求时撤销，系统会将计算出的请求签名与 StringToSign 中提供的签名进行对比。基于此原因，您必须采用 Amazon S3 使用的相同方法计算签名。根据适用于签名的统一格式放置请求的过程称为标准化。

构建 CanonicalizedResource 元素

CanonicalizedResource 表示请求的目标 Amazon S3 资源。为 REST 请求构建它，如下所示：

1. 使用空字符串 ("") 启动。
2. 如果请求使用 HTTP 主机标头 (虚拟托管类型) 来指定请求，请在 "/" (例如，"/bucketname") 的后面附加存储桶名称。对于路径类型请求和不会寻址存储桶的请求，不执行任何操作。有关虚拟托管类型请求的更多信息，请参阅[存储桶的虚拟托管](#)。

对于虚拟托管类型请求“https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg”，CanonicalizedResource 是“/awsexamplebucket1”。

对于路径类型请求“https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg”，CanonicalizedResource 为 ""。

3. 附加未解码的 HTTP 请求-URI 的路径部分 (取决于但不包括查询字符串)。

对于虚拟托管类型请求“https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg”，CanonicalizedResource 是“/awsexamplebucket1/photos/puppy.jpg”。

对于路径类型的请求，“https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg”，CanonicalizedResource 是“/awsexamplebucket1/photos/puppy.jpg”。此时，虚拟托管类型和路径类型请求的 CanonicalizedResource 相同。

对于不寻址存储桶的请求 (例如，[GET Service](#))，请附加“/”。

4. 如果请求将寻址子资源 (例如，?versioning、?location、?acl、?lifecycle 或 ?versionid)，请附加子资源、其值 (如果有) 和问号。请注意，如果存在多个子资源，子资源必须按子资源名称的字典顺序排序并使用“&”进行分隔 (例如，?acl&versionId=#)。

构建 CanonicalizedResource 元素时必须包含的子资源为：

acl、lifecycle、location、logging、notification、partNumber、policy、requestPayment、uploadId、uploadPartNumber 和 website。

如果请求指定覆盖响应标头值的查询字符串参数 (请参阅 [Get Object](#))，请附加查询字符串参数及其值。签名时，请勿对这些值进行编码；但是进行请求时，您必须对这些参数值进行编码。GET 请求中的查询字符串参数包括 response-content-type、response-content-language、response-expires、response-cache-control、response-content-disposition 和 response-content-encoding。

为多对象删除请求创建 CanonicalizedResource 时，必须包括 delete 查询字符串参数。

必须按照元素在 HTTP 请求中的显示方式，使用字母 (包括 URL 编码元字符) 签署来自 HTTP 请求-URI 的 CanonicalizedResource 元素。

CanonicalizedResource 可能与 HTTP 请求-URI 不同。尤其是，如果您的请求使用 HTTP Host 标头来指定存储桶，存储桶不会在 HTTP 请求 URI 中显示。但是，CanonicalizedResource 仍然会包含该存储桶。查询字符串参数可能也会在请求-URI 中显示，但是不会包括在 CanonicalizedResource 中。有关更多信息，请参阅[存储桶的虚拟托管](#)。

构建 CanonicalizedAmzHeaders 元素

要构建 StringToSign 的 CanonicalizedAmzHeaders 部分，请选择所有以“x-amz-”开头的 HTTP 请求标头 (使用不区分大小写的对比方式) 并遵循以下步骤。

1. 将每个 HTTP 标头名称转换为小写。例如，“X-Amz-Date”改为“x-amz-date”。
2. 根据标头名称按字典顺序排列标头集。
3. 将相同名称的标头字段合并为一个“header-name:comma-separated-value-list”对，并按照 RFC 2616 中第 4.2 节中的规定，两个值之间不留空格。例如，可以将元数据标头“x-amz-meta-username: fred”和“x-amz-meta-username: barney”合并为单个标头“x-amz-meta-username: fred,barney”。
4. 通过将折叠空格 (包括新建行) 替换为单个空格，“展开”跨多个行的长标头 (按照 RFC 2616 中第 4.2 节允许的方式)。
5. 删除标头中冒号周围的空格。例如，标头“x-amz-meta-username: fred,barney”将变为“x-amz-meta-username:fred,barney”。
6. 最后，请向生成的列表中的每个标准化标头附加换行字符 (U+000A)。通过将此列表中所有的标头规范化为单个字符串，构建 CanonicalizedResource 元素。

位置和命名 HTTP 标头 StringToSign 元素的对比

StringToSign 的头几个标头元素 (Content-Type、Date 和 Content-MD5) 属于位置标头。StringToSign 不包括这些标头的名称，仅包括它们在请求中的值。相反，“x-amz-”元素会进行命名。标头名称和标头值都会在 StringToSign 中显示。

如果在 StringToSign 定义中调用的位置标头不在请求中 (例如，Content-Type 或 Content-MD5 对于 PUT 请求是可选的，并且对于 GET 请求没有任何意义)，请使用空字符串 ("") 替换该位置。

时间戳要求

有效的时间戳对于经身份验证的请求是必须的 (使用 HTTP Date 标头或 x-amz-date 替代项)。此外，经身份验证的请求随附的客户端时间戳必须处于收到请求时的 Amazon S3 系统时间的 15 分钟之内。否则，请求将失败并出现 RequestTimeTooSkewed 错误代码。施加这些限制的目的是为了防止对方重新使用已拦截的请求。要更好地防范窃听，请对经身份验证的请求使用 HTTPS 传输。

Note

对请求日期的验证限制仅适用于不使用查询字符串身份验证的经身份验证的请求。有关更多信息，请参阅 [查询字符串请求身份验证替代项](#)。

某些 HTTP 客户端库不提供为请求设置 Date 标头的功能。如果您在标准化标头中包含“Date”标头的值时遇到困难，您可以改用“x-amz-date”标头为请求设置时间戳。x-amz-date 标头的值必须采用其中一种 RFC 2616 格式 (<http://www.ietf.org/rfc/rfc2616.txt>)。x-amz-date 标头位于请求中时，系统将在计算请求签名时忽略任何 Date 标头。因此，如果包含了 x-amz-date 标头，请在构建 Date 时使用 StringToSign 的空字符串。有关示例，请参阅下一节。

身份验证示例

本节中的示例使用下表中的 (非有效) 凭证。

参数	值
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

在示例 StringToSign 中，格式并不重要；\n 是指 Unicode 码位 U+000A，通常称为换行符。此外，该示例使用“+0000”指定时区。您可以改用“GMT”来指定时区，但是在示例中显示的签名将有所不同。

对象 GET

此示例将从 awsexamplebucket1 存储桶获取一个对象。

请求	StringToSign
GET /photos/puppy.jpg HTTP/1.1 Host: awsexamplebucket1.us-west-1.s3.amazonaws.com	GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n

请求	StringToSign
Date: Tue, 27 Mar 2007 19:36:42 +0000 <i>Authorization: AWS AKIAIOSF0 DNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</i>	/awsexamplebucket1/photos/puppy.jpg

请注意，CanonicalizedResource 包括存储桶名称，但 HTTP 请求 URI 不包括。（存储桶由主机标头指定）。

Note

以下 Python 脚本使用提供的参数来计算前面的签名。您可以使用此脚本来构建您自己的签名，并根据需要替换密钥和 StringToSign。

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
    hmac.new(
        secret_key, string_to_sign, sha1
    ).digest()
).strip()

print(f"AWS {access_key.decode()}:{signature.decode()}")
```

对象 PUT

此示例将向 awsexamplebucket1 存储桶放入一个对象。

请求	StringToSign
<pre>PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: iqRzw+ileNPu1fhspnRs8n0jjIA=</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

请注意请求和 StringToSign 中的 Content-Type 标头。另请注意，由于 Content-MD5 不在请求中，它在 StringToSign 中将保留为空白。

列表

此示例将列出 awsexamplebucket1 存储桶的内容。

请求	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>

请注意 CanonicalizedResource 上的尾部斜杠以及查询字符串参数的缺失。

取回

本示例将为“awsexamplebucket1”存储桶提取访问控制策略子资源。

请求	StringToSign
<pre>GET /?acl HTTP/1.1 Host: awsexamplebucket1.s3.us-west-1.amazo naws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: 82ZHiFIjc+WbcwFKGUEQspPn+0=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n /awsexamplebucket1/?acl</pre>

请注意子资源查询字符串参数包含在 CanonicalizedResource 中的方式。

删除

本示例使用路径类型和日期替代项从“awsexamplebucket1”存储桶删除对象。

请求	StringToSign
<pre>DELETE /awsexamplebucket1/photos/p uppy.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 AKIAIOSFODNN7EXAMP LE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

请注意我们如何使用“x-amz-date”替代方法来指定日期 (假设是因为我们的客户端库阻止我们设置日期)。在这种情况下，x-amz-date 优先于 Date 标头。因此，签名中的日期条目必须包含 x-amz-date 标头的值。

上传

本示例将对象上传到使用元数据的别名记录虚拟托管类型存储桶。

请求	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@example.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339 Authorization: AWS AKIAIOSFODNN7EXAMPLE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</pre>	<pre>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>

请注意对“x-amz-”标头排序、删减多余空格和转换为小写的方式。另请注意，具有相同名称的多个标头使用逗号连接，以分隔值。

请注意，如何仅使 Content-Type 和 Content-MD5 HTTP 实体标头显示在 StringToSign 中。其他 Content-* 实体标头不显示。

同样请注意，CanonicalizedResource 包括存储桶名称，但 HTTP 请求 URI 不包括。（存储桶由主机标头指定）。

列出我的所有存储桶

请求	StringToSign
<pre>GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com</pre>	<pre>GET\n \n</pre>

请求	StringToSign
Date: Wed, 28 Mar 2007 01:29:59 +0000 <i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh60qZehG9s=</i>	\n Wed, 28 Mar 2007 01:29:59 +0000\n /

Unicode 密钥

请求	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 AKIAIOSFODNN7EXAMPLE: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

将逐字读取来自请求-URI 的 StringToSign 中的元素，包括 URL 编码和大写。

REST 请求签名问题

REST 请求身份验证失败后，系统将使用 XML 错误文档对请求做出响应。此错误文档中包含的信息将帮助开发人员诊断问题。特别是，StringToSign 错误文档的 SignatureDoesNotMatch 元素将明确告诉您系统使用的请求规范。

某些工具包可能会在您事先不知情的情况下自动插入标头，例如在 PUT 操作期间添加标头 Content-Type。在大部分情况下，插入标头的值保持不变，从而使您可以使用诸如 Ethereal 或 tcpmon 等工具来发现丢失的标头。

查询字符串请求身份验证替代项

您可以通过传递请求信息作为查询字符串参数，而不是使用 Authorization HTTP 标头来验证特定类型的请求。这在允许第三方浏览器直接访问您的私有 Amazon S3 数据，而无需代理请求时非常有

用。其概念是构建一个“预签名”的请求并将其编码为最终用户浏览器可检索的 URL。此外，您还可以通
过指定过期时间来限制预签名请求。

有关使用查询参数对请求进行身份验证的更多信息，请参阅《Amazon Simple Storage Service API 参
考》中的[对请求进行身份验证：使用查询参数 \(AWS Signature Version 4\)](#)。有关使用 AWS SDK 生成
预签名 URL 的示例，请参阅[使用预签名 URL 共享对象](#)。

创建签名

以下是查询字符串验证的 Amazon S3 REST 请求示例。

```
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
```

查询字符串请求身份验证方法不要求任何特殊的 HTTP 标头，而是将要求的身份验证元素指定为查询
字符串参数：

查询字符串参数 名称	示例值	说明
AWSAccess KeyId	AKIAIOSFODNN7EXAMPLE	您的 AWS 访问密钥 ID。指定用于签 署请求的 AWS 秘密访问密钥以及（ 间接）进行请求的开发人员的身份。
Expires	1141889120	将签名过期时间指定为自 Epoch (1970 年 1 月 1 日 00:00:00 UTC) 以 来的秒数。将拒绝在此时间（根据服务 器）之后收到的请求。
Signature	vjbyPxybdZaNmGa%2B yT272YEAiv4%3D	采用 StringToSign 的 HMAC-SHA1 Base64 编码的 URL 编码。

查询字符串请求身份验证方法与普通的方法稍有差异，不同之处仅在于 Signature 请求参数和
StringToSign 元素的格式。下面的伪语法演示了查询字符串请求身份验证方法。

```
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` 是在您注册成为 Amazon Web Services 开发人员时，Amazon 分配给您的 AWS 秘密访问密钥 ID。请注意如何对 `Signature` 进行 URL 编码，以使其适合放置在查询字符串中。另请注意，在 `StringToSign` 中，HTTP Date 位置元素已替换为 `Expires`。`CanonicalizedAmzHeaders` 和 `CanonicalizedResource` 是相同的。

Note

在查询字符串身份验证方法中，在计算要签署的字符串时，请勿使用 `Date` 或 `x-amz-date request` 标头。

查询字符串请求身份验证

请求	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>

我们假设，在浏览器创建 GET 请求时，它没有提供 `Content-MD5` 或 `Content-Type` 标头，也没有设置任何 `x-amz-` 标头，因此 `StringToSign` 的这些部分都保留为空白。

使用 Base64 编码

必须使用 Base64 编码 HMAC 请求签名。Base64 编码将签名转换为可附加到请求的简单 ASCII 字符串。如果要在 URI 中使用诸如加号 (+)、正斜杠 (/) 和等号 (=) 等可在签名中显示的字符，必须对它们进行编码。例如，如果身份验证代码包括一个加号 (+) 标志，请在请求中将其编码为 %2B。将正斜杠编码为 %2F，并将等号编码为 %3D。

有关 Base64 编码的示例，请参考 Amazon S3 [身份验证示例](#)。

使用 POST (AWS Signature Version 2) 的基于浏览器的上传

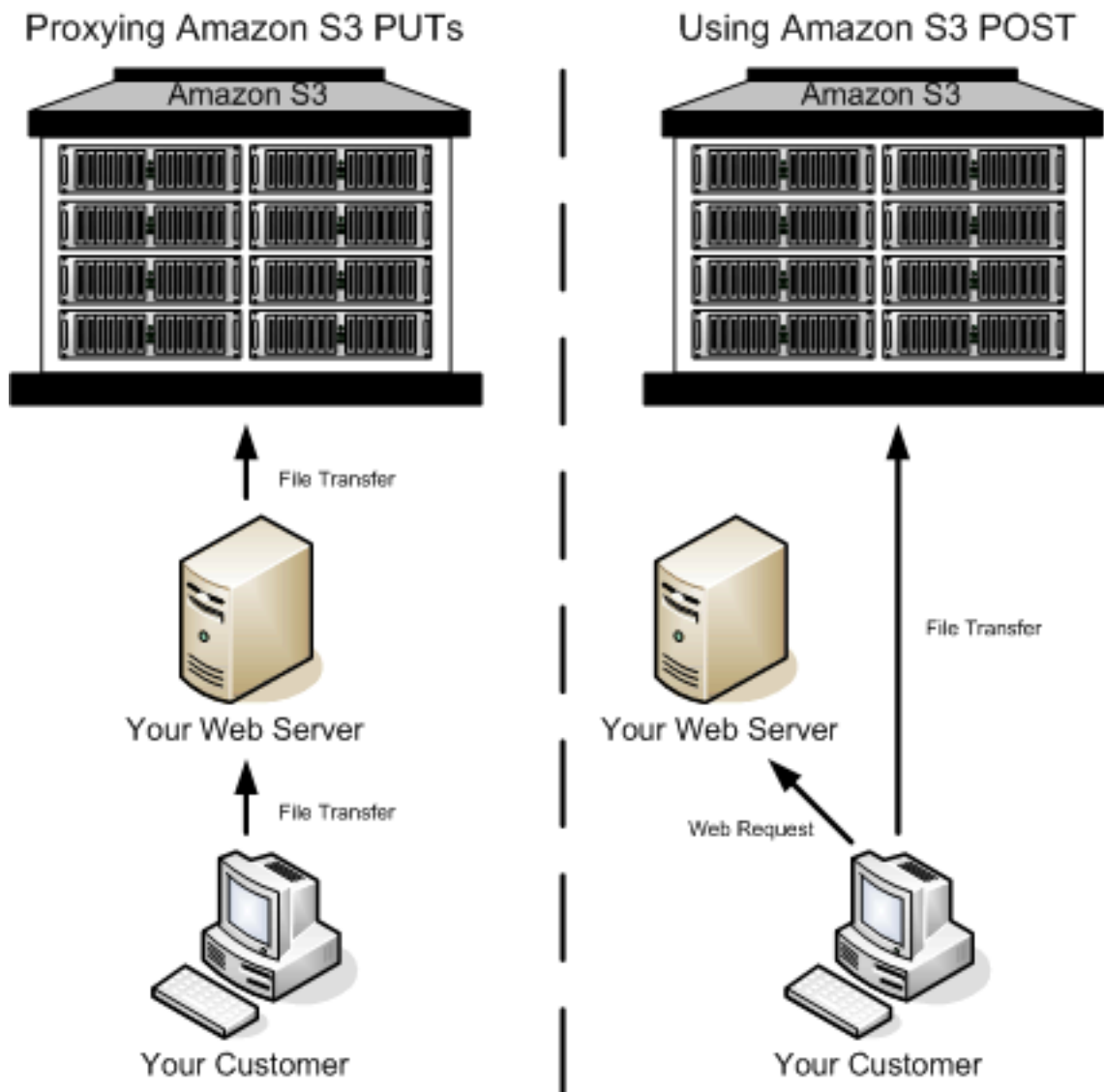
Amazon S3 支持 POST，使您的用户可以直接将内容上传到 Amazon S3。POST 旨在简化上传过程和缩短上传延迟，而且可以节省用于上传数据以存储于 Amazon S3 中所用应用程序的开支。

Note

本节中讨论的请求身份验证基于 AWS Signature Version 2，这是一种对 AWS 服务的入站 API 请求进行身份验证的协议。

Amazon S3 现在在所有 AWS 区域支持 Signature Version 4，后者是一种用于对 AWS 服务入站 API 请求进行身份验证的协议。目前，于 2014 年 1 月 30 日前创建的 AWS 区域将继续支持之前的协议：Signature Version 2。于 2014 年 1 月 30 日后创建的所有新区域将只支持 Signature Version 4，因此，发往这些区域的所有请求都必须采用 Signature Version 4。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的[使用 POST \(AWS Signature Version 4\) 在基于浏览器的上传中验证请求](#)。

下图演示了使用 Amazon S3 POST 的上传。



使用 POST 上传

- 1 用户打开 Web 浏览器并访问您的 Web 页面。
- 2 Web 页面包含一个 HTTP 表格，其中包含了用户将内容上传到 Amazon S3 时必需的所有信息。
- 3 用户直接将内容上传到 Amazon S3。

Note

POST 不支持查询字符串身份验证。

HTML 表单 (AWS Signature Version 2)

主题

- [HTML 表单编码](#)
- [HTML 表单声明](#)
- [HTML 表单字段](#)
- [策略构建](#)
- [构建签名](#)
- [重定向](#)

与 Amazon S3 进行通信时，通常可以使用 REST 或 SOAP API 来执行放置、获取、删除和其他操作。借助 POST，用户可通过其浏览器将数据直接上传到 Amazon S3，浏览器无法处理 SOAP API 或创建 REST PUT 请求。

Note

Amazon S3 的 SOAP API 不适用于新客户，并将于 2025 年 8 月 31 日接近使用寿命终止（EOL）。我们建议您使用 REST API 或 AWS 开发工具包。

要允许用户使用其浏览器将内容上传到 Amazon S3，要使用 HTML 表单。HTML 表单由表单声明和表单字段组成。表单声明包含关于请求的高级别信息。表单字段包含关于请求的详细信息，同时还包含用于对请求进行身份验证并确保其满足指定条件的策略。

Note

表单数据和边界（不包括文件的内容）不得超过 20 KB。

本部分说明如何使用 HTML 表单。

HTML 表单编码

必须采用 UTF-8 编码表单和策略。您可以将 UTF-8 编码应用于表单，方法是在 HTML 标题中指定它或将它指定为请求标头。

 Note

HTML 表单声明不接受查询字符串身份验证参数。

下面是 HTML 标题中 UTF-8 编码的示例：

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

下面是请求标头中 UTF-8 编码的示例：

```
Content-Type: text/html; charset=UTF-8
```

HTML 表单声明

表单声明包含三个部分：操作、方法和附件类型。如果这些值当中的任意一个设置不正确，请求将失败。

操作将指定处理请求的 URL，必须将它设置为存储桶的 URL。例如，如果您的存储桶名称为 `awsexamplebucket1`，区域为美国西部（加利福尼亚北部），则 URL 为 `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`。

 Note

在表单字段中指定键名。

方法必须是 POST。

必须为文件上传和文本区域上传都指定附件类型 (enctype) 并将其设置为“分段/表单数据”。有关更多信息，请转到 [RFC 1867](https://tools.ietf.org/html/rfc1867)。

Example

以下示例是用于存储桶“awsexamplebucket1”的表单声明。

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
  enctype="multipart/form-data">
```

HTML 表单字段

下表介绍可以在 HTML 表单中使用的字段。

 Note

变量 `${filename}` 将自动替换为用户提供的文件的名称，并且可由所有表单字段识别。如果浏览器或客户端提供完整或部分文件路径，则只会使用跟在最后一个斜杠 (/) 或反斜杠 (\) 之后的文本。例如，“C:\Program Files\directory1\file.txt”会解释为“file.txt”。如果没有提供文件或文件名，变量将替换为空字符串。

字段名称	说明	必需
AWSAccessKeyId	存储桶拥有者的 AWS 访问密钥 ID，该拥有者授予匿名用户对满足策略中一组约束的请求的访问权限。如果请求包含策略文档，则此字段为必填字段。	条件
acl	Amazon S3 访问控制列表 (ACL)。如果指定了无效的访问控制列表，将产生错误。 类型：字符串 默认值：private 有效值：private public-read public-read-write aws-exec-read authenticated-read	否

字段名称	说明	必需
	bucket-owner-read bucket-owner-full-control	
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	特定于 REST 的标头。有关更多信息，请参阅 PUT Object 。	否
key	已上传的键的名称。 要使用由用户提供的文件名，请使用 <code>\${filename}</code> 变量。例如，如果用户 Betty 上传了文件 <code>lolcatz.jpg</code> 并且您指定了 <code>/user/betty/\${filename}</code>，则该文件将存储为 <code>/user/betty/lolcatz.jpg</code>。 有关更多信息，请参阅使用对象元数据。	是
policy	描述请求中允许的内容的安全策略。不带安全策略的请求被认为是匿名的，只会在公共可写的存储桶上成功。	否

字段名称	说明	必需
success_action_redirect, redirect	上传成功后客户端重定向到的 URL。Amazon S3 将存储桶、密钥和 etag 值作为查询字符串参数附加到 URL。 如果未指定 success_action_redirect , Amazon S3 将返回在 success_action_status 字段中指定的空文档类型。 如果 Amazon S3 无法解释 URL , 它将忽略该字段。 如果上传失败 , Amazon S3 将显示错误并且不会将用户重定向到某个 URL。 有关更多信息 , 请参阅重定向。  Note 已弃用重定向字段名称并且将在以后移除对重定向字段名称的支持。	否

字段名称	说明	必需
success_action_status	如果没有指定 success_action_redirect，上传成功后状态代码将返回到客户端。 有效值为 200、201 或 204 (默认)。 如果值设置为 200 或 204，Amazon S3 将返回一个空文档和一个 200 或 204 状态代码。 如果值设置为 201，Amazon S3 将返回一个 XML 文档和一个 201 状态代码。有关 XML 文档内容的信息，请参阅 POST Object。 如果没有设置值或者设置了无效的值，Amazon S3 将返回一个空文档和一个 204 状态代码。  Note 某些版本的 Adobe Flash player 无法正确处理使用空白正文的 HTTP 响应。要通过 Adobe Flash 支持上传，建议您将 success_action_status 设置为 201。	否
signature	HMAC 签名，使用与提供的 AWSSecretAccessKey 对应的秘密访问密钥构建。如果策略文档随请求一起提供，则此字段为必填字段。 有关更多信息，请参阅 Amazon S3 的身份和访问权限管理。	有条件

字段名称	说明	必需
x-amz-security-token	会话凭证使用的安全令牌 如果请求使用 Amazon DevPay，则它需要两个 x-amz-security-token 表单字段：一个用于产品令牌，另一个用于用户令牌。 如果请求使用会话凭证，则它需要一个 x-amz-security-token 表单。有关更多信息，请参阅《IAM 用户指南》中的临时安全凭证。	否
其他以 x-amz-meta- 为前缀的 字段名称	特定于用户的元数据。 Amazon S3 没有验证或使用此数据。 有关更多信息，请参阅 PUT Object。	否
file	文件或文本内容。 文件或内容必须是表单中的最后一个字段。其下的任何字段都会忽略。 您一次仅能上传一个文件。	是

策略构建

主题

- [过期](#)
- [条件](#)
- [条件匹配](#)
- [字符转义](#)

策略是使用 UTF-8 和 Base64 编码的 JSON 文档，它指定了请求必须满足的条件并且用于对内容进行身份验证。根据您的设计策略文档的方式，您可以对每次上传、每个用户、所有上传或根据其他能够满足您需要的设计来使用它们。

Note

尽管策略文档是可选的，我们强烈建议您使用它来创建公开可写的存储桶。

下面是策略文档的示例：

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "awsexamplebucket1" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

策略文档包括过期和条件。

过期

过期元素采用 ISO 8601 UTC 日期格式来指定策略的过期日期。例如，“2007-12-01T12:00:00.000Z”指定策略在 2007 年 12 月 1 日午夜 UTC 之后失效。在策略中过期是必需的。

条件

策略文档中的条件验证上传的对象的内容。您在表单中指定的每个表单字段 (AWSAccessKeyId、签名、文件、策略和带 x-ignore- 前缀的字段名称除外) 必须包含在条件列表中。

 Note

如果您拥有多个具有相同名称的字段，必须使用逗号分隔值。例如，如果您拥有两个名为“x-amz-meta-tag”的字段，第一个字段的值为“Ninja”，第二个字段的值为“Stallman”，您可以将策略文档设置为 `Ninja,Stallman`。

表单中的所有变量会在验证策略之前进行扩展。因此，应该针对扩展的字段执行所有条件匹配。例如，如果您将键字段设置为 `user/betty/${filename}`，您的策略可能是 `[ "starts-with", "$key", "user/betty/" ]`。请勿输入 `[ "starts-with", "$key", "user/betty/${filename}" ]`。有关更多信息，请参阅 [条件匹配](#)。

下表介绍策略文档条件。

元素名称	说明
acl	指定 ACL 必须满足的条件。 支持精确匹配和 <code>starts-with</code> 。
content-length-range	指定已上传内容允许的最小和最大大小。 支持范围匹配。
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	特定于 REST 的标头。 支持精确匹配和 <code>starts-with</code> 。
key	已上传的键的名称。 支持精确匹配和 <code>starts-with</code> 。
success_action_redirect, 重定向	上传成功后客户端重定向到的 URL。 支持精确匹配和 <code>starts-with</code> 。
success_action_status	

元素名称	说明
	如果没有指定 success_action_redirect，上传成功后状态代码将返回到客户端。
	支持精确匹配。
x-amz-security-token	Amazon DevPay 安全令牌。 使用 Amazon DevPay 的每个请求都需要两个 x-amz-security-token 表单字段：一个用于产品令牌，一个用于用户令牌。因此，必须使用逗号来分隔值。例如，如果用户令牌是 eW91dHVIZQ==，产品令牌是 b0hnNVNKWVJIQTA=，您可以将策略条目设置为：{ "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKWVJIQTA=" }。
其他以 x-amz-meta- 为前缀的字段名称	特定于用户的元数据。 支持精确匹配和 starts-with。

 **Note**

如果您的工具包添加了其他字段 (例如，Flash 添加了文件名)，您必须将它们添加到策略文档。如果您可以控制此功能，将 x-ignore- 添加为字段的前缀以使 Amazon S3 忽略此功能并使其不影响此功能的未来版本。

条件匹配

下表介绍条件匹配类型。尽管您必须为您在表单中指定的每个表单字段指定一个条件，您也可以通过为某个表单字段指定多个条件来创建更复杂的匹配条件。

Condition	说明
精确匹配	精确匹配将验证字段是否匹配特定的值。此示例指示 ACL 必须设置为公共读取：

Condition	说明
	<pre>{"acl": "public-read" }</pre> 此示例是指示 ACL 必须设置为公共读取的替代方法： <pre>["eq", "\$acl", "public-read"]</pre>
Starts With	如果值必须从某个特定的值开始，请使用 starts-with。本示例指示键必须以 user/betty 开头： <pre>["starts-with", "\$key", "user/betty/"]</pre>
匹配任何内容	要配置策略以允许字段中的任何内容，请使用 starts-with 和一个空值。本示例允许任何 success_action_redirect： <pre>["starts-with", "\$success_action_redirect", ""]</pre>
指定范围	对于接受范围的字段，请使用逗号来分隔上限和下限值。本示例允许 1 到 10 MB 的文件大小： <pre>["content-length-range", 1048579, 10485760]</pre>

字符转义

下表介绍策略文档中必须进行转义的字符。

转义序列	说明
\\	反斜杠

转义序列	说明
\\$	美元符号
\b	退格键
\f	换页
\n	新建行
\r	回车
\t	水平选项卡
\v	垂直选项卡
\u xxxx	所有 Unicode 字符

构建签名

步骤	说明
1	使用 UTF-8 对策略进行编码。
2	使用 Base64 对这些 UTF-8 字节进行编码。
3	使用 HMAC SHA-1，通过您的秘密访问密钥签署策略。
4	使用 Base64 对 SHA-1 签名进行编码。

有关身份验证的一般信息，请参阅 [Amazon S3 的身份和访问管理](#)。

重定向

本节描述了如何处理重定向。

一般重定向

完成 POST 请求后，用户将重定向至您在 `success_action_redirect` 字段中指定的位置。如果 Amazon S3 无法解释 URL，它将忽略 `success_action_redirect` 字段。

如果未指定 `success_action_redirect`，Amazon S3 将返回在 `success_action_status` 字段中指定的空文档类型。

如果 POST 请求失败，Amazon S3 将显示错误并且不会提供重定向。

预先上传重定向

如果您的存储桶是使用 `<CreateBucketConfiguration>` 创建的，您的最终用户可能会需要重定向。如果出现这种要求，某些浏览器可能无法正确处理重定向。这是比较罕见的，但在创建存储桶之后最有可能发生这种情况。

上传示例 (AWS Signature Version 2)

主题

- [文件上传](#)
- [文本区域上传](#)

Note

本节中讨论的请求身份验证基于 AWS Signature Version 2，这是一种对 AWS 服务的入站 API 请求进行身份验证的协议。

Amazon S3 现在在所有 AWS 区域支持 Signature Version 4，后者是一种用于对 AWS 服务入站 API 请求进行身份验证的协议。目前，于 2014 年 1 月 30 日前创建的 AWS 区域将继续支持之前的协议：Signature Version 2。于 2014 年 1 月 30 日后创建的所有新区域将只支持 Signature Version 4，因此，发往这些区域的所有请求都必须采用 Signature Version 4。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的[示例：使用 HTTP POST 的基于浏览器的上传（使用 AWS Signature Version 4）](#)。

文件上传

此示例演示构造用于上传文件附件的策略和表单的完整过程。

策略和表单构建

以下策略支持针对 `awsexamplebucket1` 存储桶上传到 Amazon S3。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

此策略要求以下内容：

- 上传必须在 2007 年 12 月 1 日 12:00 UTC 之前进行。
- 内容必须上传到 `awsexamplebucket1` 存储桶。
- 键必须以“user/eric/”开头。
- ACL 必须设置为公共读取。
- `success_action_redirect` 设置为 `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`。
- 对象是一个图像文件。
- `x-amz-meta-uuid` 标签必须设置为 14365123651274。
- `x-amz-meta-tag` 可以包含任意值。

下面是此策略的 Base64 编码版本。

```
eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgcyJidWN
```

使用您的凭证创建签名，例如 0RavWzkygo6QX9caELEqKi9kDbU= 是先前的策略文档的签名。

以下格式支持对使用此策略的 amzn-s3-demo-bucket 存储桶发出 POST 请求。

```
<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>
```

示例请求

此请求假设上传的图像为 117,108 字节；不包括图像数据。

```
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"

eyJhZiZxhwaXJhdGlvbml6ICl0MDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgcyJidWN
--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--
```

示例响应

```
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
```

文本区域上传

主题

- [策略和表单构建](#)
- [示例请求](#)
- [示例响应](#)

以下示例演示构造策略和表单以上传文本区域的完整过程。上传文本区域对于提交用户创建的内容 (如博客文章) 十分有用。

策略和表单构建

以下策略支持针对 awsexamplebucket1 存储桶将文本区域上传到 Amazon S3。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
```

```

    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}

```

此策略要求以下内容：

- 上传必须在 2007 年 12 月 1 日 12:00 GMT 之前进行。
- 内容必须上传到 `awsexamplebucket1` 存储桶。
- 键必须以“`user/eric/`”开头。
- ACL 必须设置为公共读取。
- `success_action_redirect` 设置为 `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html`。
- 对象是 HTML 文本。
- `x-amz-meta-uuid` 标签必须设置为 14365123651274。
- `x-amz-meta-tag` 可以包含任意值。

下面是此策略的 Base64 编码版本。

eyJhZGhwaXJhdGlvbiI6ICJyMDA3LTYeLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWnrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlciLAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3Q0IiAiaHR0cDovL2pC5zMy5hbWV6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0CAGIHsieC1hbXotbWV0YS1ldWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1IsICIiXQogIF0KfQo=

使用您的凭证创建签名。例如，qA7FWXKq6VvU68lI9KdveT1cWgF= 是先前的策略文档的签名。

以下格式支持对使用此策略的 `amzn-s3-demo-bucket` 存储桶发出 POST 请求。

```
<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>

```

示例请求

此请求假设上传的图像为 117,108 字节；不包括图像数据。

```

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"
eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyMDAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgcyJidWN
--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--
```

示例响应

```
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
```

使用 Adobe Flash 进行 POST (AWS 签名版本 2)

本节描述了如何使用 POST 和 Adobe Flash。

Adobe Flash Player 的安全性

在默认情况下，Adobe Flash Player 安全模型禁止 Adobe Flash Players 创建指向位于域 (服务 SWF 文件) 之外的服务器的网络连接。

要覆盖默认设置，您必须将公共可读的 crossdomain.xml 文件上传到接受 POST 上传的存储桶。下面是示例 crossdomain.xml 文件。

```
<?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

有关 Adobe Flash 安全模型的更多信息，请转到 Adobe 网站。
向存储桶添加 crossdomain.xml 文件可允许所有 Adobe Flash Player 连接到您存储桶中的 crossdomain.xml 文件；但是，它不会授予对实际 Amazon S3 存储桶的访问权限。

Adobe Flash 注意事项

Adobe Flash 中的 FileReference API 将 Filename 表单字段添加到 POST 请求。在构建使用 FileReference API 操作上传到 Amazon S3 的 Adobe Flash 应用程序时，请在策略中包括下面的条件：

```
['starts-with', '$Filename', '']
```

某些版本的 Adobe Flash Player 无法正确处理带空白正文的 HTTP 响应。要将 POST 配置为返回不带空白正文的响应，请将 success_action_status 设置为 201。Amazon S3 随后将返回一个具有 201 状态代码的 XML 文档。有关 XML 文档内容的信息，请参阅 [POST Object](#)。有关表单字段的信息，请参阅 [HTML 表单字段](#)。

附录：生命周期配置 API (已弃用)

存储桶生命周期配置已更新，以支持基于对象标签的筛选条件。也即，您现在可以指定一条规则，该规则指定键名称前缀、一个或多个对象标签或者同时指定这两者，以选择该规则适用的对象子集。API 相应进行了更新。以下主题介绍 PUT 和 GET 存储桶生命周期操作的先前版本，以实现向后兼容。

主题

- [PUT 存储桶生命周期 \(已弃用 \)](#)
- [GET 存储桶生命周期 \(已弃用 \)](#)

PUT 存储桶生命周期 (已弃用)

说明

Important

有关此 API 的已更新版本，请参阅 [PutBucketLifecycleConfiguration](#)。此版本已弃用。现有的生命周期配置将生效。对于新的生命周期配置，请使用已更新的 API。

为存储桶创建新的生命周期配置，或替换现有的生命周期配置。有关生命周期配置的信息，请参阅《Amazon Simple Storage Service 用户指南》中的[对象生命周期管理](#)。

权限

默认情况下，所有 Amazon S3 资源都是私有的，包括存储桶、对象和相关子资源（例如，生命周期配置和网站配置）。只有资源拥有者和创建资源的 AWS 账户可以访问该资源。资源拥有者可以选择通过编写访问策略向他人授予访问权限。要执行此操作，用户必须获得 `s3:PutLifecycleConfiguration` 权限。

您也可以显式拒绝权限。显式拒绝也会取代任何其它权限。如果您想阻止用户或账户从您的存储桶中移除或删除对象，则必须拒绝他们执行以下操作的权限：

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

有关权限的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[管理对 Amazon S3 资源的访问权限](#)。

请求

语法

```
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

有关授权字符串的详细信息，请参阅[对请求进行身份验证 \(AWS 签名版本 4 \)](#)。

请求参数

此操作的实施不使用请求参数。

请求标头

名称	描述	必填
Content-MD5	数据的 base64 编码的 128 位 MD5 摘要。您必须将此标头用作消息完整性检查，以验证请求正文在传输过程中是否未损坏。有关更多信息，请参阅 RFC 1864。 类型：字符串 默认值：无	是

请求正文

在请求中，您可以在请求正文中指定生命周期配置。生命周期配置以 XML 形式指定。以下是一个基本生命周期配置的示例。它指定了一条规则。规则中的 Prefix 标识规则适用的对象。该规则还指定了两个操作 (Transition 和 Expiration)。每个操作都指定了 Amazon S3 应执行该操作的时间表。Status 指示规则是已启用还是已禁用。

```
<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>

```

如果存储桶的状态为已启用版本控制或已暂停版本控制，则可能有同一对象的许多版本：一个当前版本和零个或零个以上非当前版本。以下生命周期配置指定了特定于非当前对象版本的操作（NoncurrentVersionTransition、NoncurrentVersionExpiration）。

```

<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>

```

您可以使用分段上传 API 来分段上传大型对象。有关分段上传的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[分段上传概述](#)。利用生命周期配置，如果分段上传未在指定的天数内完成，您可以指示 Amazon S3 取消未完成的分段上传（通过在规则中指定的键名称前缀确定）。当 Amazon S3 取消分段上传时，它将删除与上传关联的所有分段。这可确保您不会有未完成的分段上传（未完成的分段上传会将剩余的分段存储在 Amazon S3 中），因此您不必为它们支付存储成本。下面是使用 AbortIncompleteMultipartUpload 操作指定规则的示例生命周期配置。此操作指示 Amazon S3 在分段上传启动 7 天后取消未完成的分段上传。

```

<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>

```

```
</LifecycleConfiguration>
```

下表描述了生命周期配置中的 XML 元素。

名称	描述	必填
AbortIncompleteMultipartUpload	用于指定未完成的分段上传何时符合中止操作条件的容器。 子级：DaysAfterInitiation 类型：容器 原级：Rule	是，如果没有为规则指定其它操作
Date	您希望 Amazon S3 执行此操作的日期。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的生命周期规则：基于特定日期。 此日期值必须符合 ISO 8601 格式。时间始终为午夜 UTC。 类型：字符串 原级：Expiration 或 Transition	是，如果 Days 和 ExpiredObjectDeleteMarker 不存在
Days	指定对象创建后特定规则操作生效的天数。 类型：与 Transition 结合使用时为非负整数，与 Expiration 结合使用时为正整数 原级：Expiration 、 Transition	是，如果 Date 和 ExpiredObjectDeleteMarker 不存在
DaysAfterInitiation	指定启动分段上传后必须完成分段上传的天数。如果分段上传未在指定的天数内完成，则	是，如果指定了父标签

名称	描述	必填
	将符合中止操作的条件，而 Amazon S3 将取消未完成的分段上传。 类型：正整数 原级：AbortIncompleteMultipartUpload	

名称	描述	必填
Expiration	此操作指定在对象的生命周期中 Amazon S3 应采取适当过期操作的时段。Amazon S3 采取的操作取决于存储桶是否启用了版本控制。 • 如果从未在存储桶上启用版本控制，Amazon S3 将永久删除该对象的唯一副本。 • 如果存储桶启用了版本控制（或暂停了版本控制），则该操作仅适用于对象的当前版本。启用了版本控制的存储桶可能具有同一对象的许多版本：一个当前版本和零个或零个以上非当前版本。 Amazon S3 不是删除当前版本，而是通过添加删除标记作为新的当前版本，使当前版本成为非当前版本。  Important 如果存储桶的状态为已暂停版本控制，则 Amazon S3 将创建一个版本 ID 为 null 的删除标记。如果您有一个版本 ID 为 null 的版本，Amazon S3 会覆盖该版本。  Note 要为非当前对象设置过期，请使用 NoncurrentVersionExpiration 操作。	是，如果 Rule 中没有其它操作。

名称	描述	必填
	类型：容器 子级：Days 或 Date 原级：Rule	
ID	规则的唯一标识符。该值不能超过 255 个字符。 类型：字符串 原级：Rule	否
LifecycleConfiguration	生命周期规则的容器。您可以添加多达 1000 个规则。 类型：容器 子级：Rule 原级：无	是

名称	描述	必填
ExpiredObjectDeleteMarker	在受版本控制的存储桶（已启用版本控制或已暂停版本控制的存储桶）上，您可以在生命周期配置中添加此元素，以指示 Amazon S3 删除过期的对象删除标记。有关示例，请参阅《Amazon Simple Storage Service 用户指南》中的示例 8：移除过期的对象删除标记。不要将其添加到不受版本控制的存储桶中，因为该类型的存储桶不能包含删除标记。 类型：字符串 有效值：true false（支持值 false，但它是无操作的，这意味着 Amazon S3 不会执行任何操作） 原级：Expiration	是，如果 Date 和 Days 不存在
NoncurrentDays	指定在 Amazon S3 可以执行关联操作之前，对象作为非当前对象的天数。有关非当前版本天数计算的信息，请参阅《Amazon Simple Storage Service 用户指南》中的Amazon S3 如何计算对象成为非当前版本的时间。 类型：与 NoncurrentVersionTransition 结合使用时为非负整数，与 NoncurrentVersionExpiration 结合使用时为正整数 原级：NoncurrentVersionExpiration 或 NoncurrentVersionTransition	是

名称	描述	必填
NoncurrentVersionExpiration	指定非当前对象版本的到期时间。到期后，Amazon S3 将永久删除非当前对象版本。 在已启用（或已暂停）版本控制的存储桶上设置此生命周期配置操作，以指示 Amazon S3 在对象生命周期的特定时段删除非当前对象版本。 类型：容器 子级：NoncurrentDays 原级：Rule	是，如果 Rule 中没有其它操作
NoncurrentVersionTransition	描述非当前对象何时转换为 STANDARD_IA、ONEZONE_IA 或存储类别的转换规则的容器。 如果存储桶启用了版本控制（或者如果版本控制已暂停），则可以设置此操作，以指示 Amazon S3 在对象生命周期的特定时段转换非当前对象版本。 类型：容器 子级：NoncurrentDays 和 StorageClass 原级：Rule	是，如果 Rule 中没有其它操作
Prefix	标识要应用该规则的一个或多个对象的对象键前缀。 类型：字符串 原级：Rule	是

名称	描述	必填
Rule	生命周期规则的容器。生命周期配置可以包含多达 1000 条规则。 类型：容器 原级：LifecycleConfiguration	是
Status	如果启用，Amazon S3 将按计划执行该规则。 如果禁用，Amazon S3 将忽略该规则。 类型：字符串 原级：Rule 有效值：已启用、已禁用	是
StorageClass	指定要将对象转换到的 Amazon S3 存储类别。 类型：字符串 原级：Transition 和 NoncurrentVersionTransition 有效值：STANDARD_IA ONEZONE_IA GLACIER	是 仅当指定其一个或两个原级时，才需要此元素。

名称	描述	必填
Transition	此操作在对象的生命周期中指定一个时段，在此时段内，Amazon S3 应将对象转换为 STANDARD_IA 、 ONEZONE_IA 或存储类别。此操作生效后，Amazon S3 的行为取决于存储桶是否启用了版本控制。 • 如果从未在存储桶上启用版本控制，则 Amazon S3 将对象的唯一副本转换为指定的存储类别。 • 如果您的存储桶启用了版本控制（或者暂停了版本控制），则 Amazon S3 仅转换在规则中标识的对象当前版本。  Note 启用版本控制的存储桶可能包含对象的许多版本。此操作对非当前对象版本没有任何效果。要转换非当前对象，必须使用 NoncurrentVersionTransition 操作。 类型：容器 子级：Days 或 Date ， 以及 StorageClass 原级：Rule	是，如果 Rule 中没有其它操作

响应

响应标头

此操作的实施仅使用大多数响应通用的响应标头。有关更多信息，请参阅[通用响应标头](#)。

响应元素

此操作的实施不返回响应元素。

特殊错误

此操作的实施不返回特殊错误。有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

示例

示例 1：向未启用版本控制的存储桶添加生命周期配置

以下生命周期配置指定了两个规则，每个规则都有一个操作。

- 转换操作指示 Amazon S3 在创建带“documents/”前缀的对象后 30 天将其转换为相应的存储类别。
- 过期操作指示 Amazon S3 在创建带“logs/”前缀的对象后 365 天将其删除。

```
<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>
```

以下是将上述生命周期配置添加到 examplebucket 存储桶的示例 PUT `/?lifecycle` 请求。

```
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>
```

以下为示例响应。

```
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
```


示例 2：向启用版本控制的存储桶添加生命周期配置

以下生命周期配置指定了两个规则，每个规则都有一个要由 Amazon S3 执行的操作。当存储桶启用了版本控制或暂停了版本控制时，您可以指定这些操作：

- `NoncurrentVersionExpiration` 操作指示 Amazon S3 在带“logs/”前缀的对象变为非当前版本后的 100 天，使这些对象的非当前版本过期。
- `NoncurrentVersionTransition` 操作指示 Amazon S3 在带“documents/”前缀的对象变为非当前版本后的 30 天，将这些对象的非当前版本转换为存储类别。

```
<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>
```

以下是将上述生命周期配置添加到 examplebucket 存储桶的示例 PUT `/?lifecycle` 请求。

```
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>
```

以下为示例响应。

```
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
```

其他示例

有关将对象转换为 STANDARD_IA 或 ONEZONE_IA 等存储类别的更多示例，请参阅[生命周期配置示例](#)。

相关资源

- [GetBucketLifecycleConfiguration](#)
- [POST Object restore](#)
- 默认情况下，资源拥有者（在这种情况下为存储桶拥有者，即创建存储桶的 AWS 账户）可以执行任何操作。资源拥有者也可以向他人授予执行该操作的权限。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的以下主题：
 - [在策略中指定权限](#)

- [管理对 Amazon S3 资源的访问权限](#)

GET 存储桶生命周期 (已弃用)

说明

Important

有关此 API 的已更新版本，请参阅 [GetBucketLifecycleConfiguration](#)。如果您使用 <filter> 元素配置了存储桶生命周期，则应该会看到本主题的已更新版本。提供此主题是为了向后兼容。

返回在存储桶上设置的 lifecycle 配置信息。有关生命周期配置的信息，请转至《Amazon Simple Storage Service 用户指南》中的[对象生命周期管理](#)。

要使用此操作，您必须拥有执行 s3:GetLifecycleConfiguration 操作的权限。默认情况下，存储桶所有者拥有此权限。存储桶所有者可以将此权限授予他人。有关权限的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[管理对 Amazon S3 资源的访问权限](#)。

请求

语法

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see #####AWS ##### 4#)
```

请求参数

此操作的实施不使用请求参数。

请求标头

此操作的实施仅使用所有操作通用的请求标头。有关更多信息，请参阅[通用请求标头](#)。

请求元素

此操作的实施不使用请求元素。

响应

响应标头

此操作的实施仅使用大多数响应通用的响应标头。有关更多信息，请参阅[通用响应标头](#)。

响应元素

GET 的实施将返回以下响应元素。

名称	描述	必填
AbortIncompleteMultipartUpload	用于指定未完成的分段上传何时符合中止操作条件的容器。 子级：DaysAfterInitiation 类型：容器 原级：Rule	是，如果没有为规则指定其它操作
Date	您希望 Amazon S3 执行此操作的日期。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的生命周期规则：基于特定日期。 此日期值必须符合 ISO 8601 格式。时间始终为午夜 UTC。 类型：字符串 原级：Expiration 或 Transition	是，如果 Days 和 ExpiredObjectDeleteMarker 不存在
Days	指定对象创建后特定规则操作生效的天数。对象符合条件的时间按“创建时间 + 天数”计算，并将得出的时间四舍五入至次日的 UTC 午夜零点。	是，如果 Date 和 ExpiredObjectDeleteMarker 不存在

名称	描述	必填
	类型：与 Transition 结合使用时为非负整数，与 Expiration 结合使用时为正整数。 原级：Transition 或 Expiration	
DaysAfterInitiation	指定启动分段上传后必须完成分段上传的天数。如果分段上传未在指定的天数内完成，则将符合中止操作的条件，而 Amazon S3 将取消未完成的分段上传。 类型：正整数 原级：AbortIncompleteMultipartUpload	是，如果 Date 不存在

名称	描述	必填
Expiration	此操作指定在对象的生命周期中 Amazon S3 应采取适当过期操作的时段。过期操作仅适用于根据在子 Date 或 Days 元素中指定的时段符合条件的对象。Amazon S3 采取的操作取决于存储桶是否启用了版本控制。 • 如果从未在存储桶上启用版本控制，Amazon S3 将永久删除该对象的唯一副本。 • 否则，如果存储桶启用了版本控制（或暂停了版本控制），则该操作仅适用于对象的当前版本。启用版本控制或暂停版本控制的存储桶可能有同一对象的许多版本：一个当前版本和零个或零个以上非当前版本。 Amazon S3 不是删除当前版本，而是通过添加删除标记作为新的当前版本，使当前版本成为非当前版本。  Important 如果存储桶的状态为已暂停版本控制，则 Amazon S3 将创建一个版本 ID 为 null 的删除标记。如果您有一个版本 ID 为 null 的版本，则 Amazon S3 会覆盖该版本。	是，如果指定了父标签

名称	描述	必填
	 Note 要为非当前对象设置过期，必须使用 NoncurrentVersionExpiration 操作。 类型：容器 子级：Days 或 Date 原级：Rule	
ID	规则的唯一标识符。该值不能超过 255 个字符。 类型：字符串 原级：Rule	否
LifecycleConfiguration	生命周期规则的容器。您可以添加多达 1000 个规则。 类型：容器 子级：Rule 原级：无	是

名称	描述	必填
ExpiredObjectDeleteMarker	在受版本控制的存储桶（已启用版本控制或已暂停版本控制的存储桶）上，此元素指示 Amazon S3 是否将删除存储桶中任何过期的对象删除标记。有关示例，请参阅《Amazon Simple Storage Service 用户指南》中的示例 8：指定过期操作以移除过期的对象删除标记。 类型：字符串 有效值：true false（支持值 false，但它是无操作的，如果该值为 false，Amazon S3 不会执行任何操作） 原级：Expiration	是，如果 Date 和 Days 不存在
NoncurrentDays	指定在 Amazon S3 可以执行关联的操作之前，对象为非当前对象的天数。有关计算非当前天数的信息，请参阅《Amazon Simple Storage Service 用户指南》中的基于天数的生命周期规则。 类型：与 NoncurrentVersionTransition 结合使用时为非负整数，与 NoncurrentVersionExpiration 结合使用时为正整数 原级：NoncurrentVersionExpiration 或 NoncurrentVersionTransition	是，前提是原级存在

名称	描述	必填
NoncurrentVersionExpiration	指定非当前对象版本的到期时间。到期后，Amazon S3 将永久删除非当前对象版本。 在已启用（或已暂停）版本控制的存储桶上设置此生命周期配置操作，以请求 Amazon S3 在对象生命周期中的特定时段删除非当前对象版本。 类型：容器 子级：NoncurrentDays 原级：Rule	是，如果 Rule 中没有其它操作
NoncurrentVersionTransition	描述非当前对象何时转换为 STANDARD_IA、ONEZONE_IA 或存储类别的转换规则的容器。 如果存储桶启用了版本控制（或暂停了版本控制），则可以设置此操作以请求 Amazon S3 在对象生命周期中的特定时段，将非当前对象版本转换为存储类别。 类型：容器 子级：NoncurrentDays 和 StorageClass 原级：Rule	是，如果 Rule 中没有其它操作
Prefix	标识要应用该规则的一个或多个对象的对象键前缀。 类型：字符串 原级：Rule	是

名称	描述	必填
Rule	生命周期规则的容器。 类型：容器 原级：LifecycleConfiguration	是
Status	如果启用，Amazon S3 将按计划执行该规则。 如果禁用，Amazon S3 将忽略该规则。 类型：字符串 原级：Rule 有效值：启用或禁用	是
StorageClass	指定要将对象转换到的 Amazon S3 存储类别。 类型：字符串 原级：Transition 和 NoncurrentVersionTransition 有效值：STANDARD_IA ONEZONE_IA	是

名称	描述	必填
Transition	此操作在对象的生命周期中指定一个时段，在此时段内，Amazon S3 应将对象转换为 STANDARD_IA 、 ONEZONE_IA 或存储类别。此操作生效后，Amazon S3 的行为取决于存储桶是否启用了版本控制。 - 如果从未在存储桶上启用版本控制，则 Amazon S3 将对象的唯一副本转换为指定的存储类别。 - 当您的存储桶启用了版本控制 (或者暂停了版本控制) 时，Amazon S3 仅转换规则中标识的对象的当前版本。  Note 启用版本控制或暂停版本控制的存储桶可能包含对象的许多版本。此操作对非当前对象版本没有任何效果。要转换非当前对象，必须使用 NoncurrentVersionTransition 操作。 类型：容器 子级：Days 或 Date ， 以及 StorageClass 原级：Rule	是，如果 Rule 中没有其它操作

特殊错误

错误代码	说明	HTTP 状态代码	SOAP 故障代码前缀
NoSuchLifecycleConfiguration	生命周期配置不存在。	404 未找到	客户端

有关 Amazon S3 错误的一般信息以及错误代码列表，请参阅[错误响应](#)。

示例

示例 1：检索生命周期子资源

此示例是用于从指定的存储桶检索 lifecycle 子资源的 GET 请求，以及包含返回的生命周期配置的示例响应。

示例请求

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

示例响应

```
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>
```

相关资源

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

文档历史记录