

---

# Amazon Simple Storage Service

## 开发人员指南

### API 版本 2006-03-01



## Amazon Simple Storage Service: 开发人员指南

Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

## Table of Contents

什么是 Amazon S3 ? .....	1
如何... ? .....	1
介绍 .....	2
Amazon S3 和本指南的概述 .....	2
Amazon S3 的优势 .....	2
Amazon S3 概念 .....	2
存储桶 .....	3
对象 .....	3
键 .....	3
区域 .....	3
Amazon S3 数据一致性模型 .....	3
Amazon S3 功能 .....	5
存储类别 .....	5
存储桶策略 .....	6
AWS Identity and Access Management .....	6
访问控制列表 .....	6
版本控制 .....	7
操作 .....	7
Amazon S3 应用程序编程接口 (API) .....	7
REST 接口 .....	7
SOAP 接口 .....	7
为 Amazon S3 付费 .....	8
相关服务 .....	8
创建请求 .....	9
关于访问密钥 .....	9
AWS 账户访问密钥 .....	9
IAM 用户访问密钥 .....	9
临时安全凭证 .....	10
请求终端节点 .....	10
通过 IPv6 发出请求 .....	10
IPv6 入门 .....	10
在 IAM 策略中使用 IPv6 地址 .....	11
测试 IP 地址兼容性 .....	12
使用双堆栈终端节点 .....	13
使用 AWS 开发工具包创建请求 .....	16
使用 AWS 账户或 IAM 用户凭证 .....	16
使用 IAM 用户临时凭证 .....	23
使用联合身份用户临时凭证 .....	31
使用 REST API 创建请求 .....	41
双堆栈终端节点 (REST API) .....	42
存储桶的虚拟托管 .....	42
请求重定向和 REST API .....	46
存储桶 .....	49
创建存储桶 .....	49
关于权限 .....	50
访问存储桶 .....	51
存储桶配置选项 .....	51
限制 .....	53
命名规则 .....	53
创建存储桶的示例 .....	54
使用 Amazon S3 控制台 .....	54
使用 AWS SDK for Java .....	55
使用适用于 .NET 的 AWS 开发工具包 .....	55
使用适用于 Ruby 的 AWS 开发工具包版本 3 .....	57

使用其他 AWS 开发工具包 .....	57
删除或清空存储桶 .....	57
删除存储桶 .....	57
清空存储桶 .....	60
存储桶的默认加密 .....	61
如何设置 Amazon S3 默认存储桶加密 .....	61
从使用存储桶策略执行加密转至默认加密 .....	62
将默认加密用于跨区域复制 .....	62
使用 CloudTrail 和 CloudWatch 监控默认加密 .....	63
更多信息 .....	63
存储桶网站配置 .....	63
使用 AWS 管理控制台 .....	63
使用 AWS SDK for Java .....	63
使用 适用于 .NET 的 AWS 开发工具包 .....	65
使用 适用于 PHP 的开发工具包 .....	66
使用 REST API .....	67
Transfer Acceleration .....	67
为什么要使用 Transfer Acceleration ? .....	67
入门 .....	68
使用 Amazon S3 Transfer Acceleration 的要求 .....	69
Transfer Acceleration 示例 .....	69
申请方付款存储桶 .....	73
使用控制台进行配置 .....	74
使用 REST API 进行配置 .....	75
费用详细信息 .....	77
访问控制 .....	77
账单和使用率报告 .....	77
账单报告 .....	77
使用情况报告 .....	79
了解账单和使用率报告 .....	80
使用成本分配标签 .....	84
对象 .....	86
对象键和元数据 .....	86
对象键 .....	87
对象元数据 .....	89
存储类别 .....	90
经常访问对象的存储类 .....	91
不经常访问对象的存储类 .....	91
GLACIER 存储类 .....	92
存储类：比较持久性和可用性 .....	92
设置对象的存储类 .....	93
子资源 .....	94
版本控制 .....	94
对象标签 .....	96
与对象标签相关的 API 操作 .....	97
对象标签和其他信息 .....	98
管理对象标签 .....	101
生命周期管理 .....	104
我应何时使用生命周期配置 ? .....	104
如何配置生命周期 ? .....	105
其他注意事项 .....	105
生命周期配置元素 .....	110
生命周期配置的示例 .....	115
设置生命周期配置 .....	124
跨源资源共享 (CORS) .....	133
跨源资源共享：使用案例场景 .....	133
如何在我的存储桶上配置 CORS ? .....	133

Amazon S3 如何评估针对存储桶的 CORS 配置？	135
启用 CORS	135
CORS 问题排查	141
在对象上的操作	141
获取对象	142
上传对象	150
复制对象	187
列出对象键	197
删除对象	203
从对象中选择内容	220
恢复存档对象	223
查询存档对象	226
存储类分析	230
如果设置存储类分析	230
存储类分析	231
如何导出存储类分析数据？	233
存储类分析导出文件布局	234
Amazon S3 分析 REST API	234
清单	235
如何设置 Amazon S3 清单	235
Amazon S3 清单存储桶	235
设置 Amazon S3 清单	236
清单列表	237
清单一致性	238
清单列表的位置	238
什么是清单 Manifest？	239
在清单完成时发送通知	240
使用 Athena 查询清单	240
Amazon S3 清单 REST API	241
关闭访问	242
介绍	242
概述	243
Amazon S3 如何对请求授权	248
有关使用可用访问策略选项的准则	252
示例演练：管理访问	255
使用存储桶策略和用户策略。	279
访问策略语言概述	279
存储桶策略示例	304
用户策略示例	312
使用 ACL 管理访问	333
访问控制列表 (ACL) 概述	333
管理 ACL	338
保护数据	345
数据加密	345
服务器端加密	345
客户端加密	372
版本控制	380
如何对存储桶配置版本控制	380
MFA 删除	381
相关主题	382
示例	382
在启用了版本控制的存储桶中管理对象	384
管理已暂停版本控制的存储桶中的对象	398
托管静态网站	401
网站终端节点	402
Amazon 网站和 REST API 终端节点之间的主要差异	402
为网站托管配置存储桶	403

启用网站托管 .....	403
配置索引文档支持 .....	403
网站访问所需的权限 .....	405
(可选) 配置 Web 流量日志记录 .....	406
(可选) 自定义错误文档支持 .....	406
(可选) 配置重定向 .....	408
示例演练 .....	414
示例：设置静态网站 .....	414
示例：使用自定义域设置静态网站 .....	415
示例：使用 Amazon CloudFront 为网站提速 .....	421
清理示例资源 .....	423
通知功能 .....	425
概述 .....	425
如何启用事件通知 .....	426
事件通知类型和目标 .....	427
受支持的事件类型 .....	427
受支持的目标 .....	428
使用对象键名称筛选配置通知 .....	428
采用对象键名称筛选的有效通知配置的示例 .....	429
采用无效前缀/后缀重叠的通知配置的示例 .....	431
授予将事件通知消息发布到目标的权限 .....	432
授予调用 AWS Lambda 函数的权限 .....	432
授予将消息发布到 SNS 主题或 SQS 队列的权限 .....	433
示例演练 1 .....	434
演练摘要 .....	434
步骤 1：创建 Amazon SNS 主题 .....	435
步骤 2：创建 Amazon SQS 队列 .....	436
步骤 3：将通知配置添加到存储桶 .....	437
步骤 4：测试设置 .....	439
示例演练 2 .....	439
事件消息结构 .....	439
跨区域复制 (CRR) .....	442
使用案例方案 .....	442
要求 .....	443
相关主题 .....	443
复制和不复制的内容 .....	444
复制的内容 .....	444
不复制的内容 .....	444
相关主题 .....	445
设置 CRR .....	445
为由同一个 AWS 账户拥有的存储桶设置跨区域复制 .....	446
为由其他 AWS 账户拥有的存储桶设置跨区域复制 .....	450
相关主题 .....	451
其他 CRR 配置 .....	451
CRR：更改副本拥有者 .....	451
CRR：复制使用 AWS KMS 托管加密密钥通过 SSE 创建的对象 .....	453
CRR 示例 .....	457
演练 1：同一 AWS 账户 .....	457
演练 2：不同的 AWS 账户 .....	458
CRR：其他演练 .....	462
使用控制台 .....	468
使用 AWS SDK for Java .....	468
使用适用于 .NET 的 AWS 开发工具包 .....	470
CRR 状态信息 .....	472
相关主题 .....	473
CRR 问题排查 .....	473
相关主题 .....	474

CRR : 其他注意事项 .....	474
生命周期配置和对象副本 .....	474
版本控制配置和复制配置 .....	474
日志记录配置和复制配置 .....	474
CRR 和目标区域 .....	475
暂停复制配置 .....	475
相关主题 .....	475
请求路由选择 .....	476
请求重定向和 REST API .....	476
DNS 路由选择 .....	476
临时请求重定向 .....	477
永久请求重定向 .....	478
请求重定向示例 .....	479
DNS 注意事项 .....	479
性能优化 .....	480
请求速率和性能指南 .....	480
GET 密集型工作负载 .....	480
TCP 窗口缩放 .....	480
TCP 选择性认可 .....	480
监控 .....	481
监控工具 .....	481
自动化工具 .....	481
手动工具 .....	481
使用 CloudWatch 监控指标 .....	482
指标与维度 .....	482
存储桶的 Amazon S3 CloudWatch 每日存储指标 .....	482
Amazon S3 CloudWatch 请求指标 .....	483
Amazon S3 CloudWatch 维度 .....	485
访问 CloudWatch 指标 .....	486
相关资源 .....	486
存储桶的指标配置 .....	487
最大努力 CloudWatch 指标传输 .....	487
筛选指标配置 .....	487
如何添加指标配置 .....	488
使用 AWS CloudTrail 记录 API 调用 .....	488
CloudTrail 中的 Amazon S3 信息 .....	489
使用 CloudTrail 日志和 Amazon S3 服务器访问日志及 CloudWatch Logs .....	493
示例 : Amazon S3 日志文件条目 .....	493
相关资源 .....	495
BitTorrent .....	496
BitTorrent 传输的收费方式 .....	496
使用 BitTorrent 来检索存储在 Amazon S3 中的对象 .....	496
使用 Amazon S3 和 BitTorrent 发布内容 .....	497
错误处理 .....	498
REST 错误响应 .....	498
响应标头 .....	498
错误响应 .....	499
SOAP 错误响应 .....	499
Amazon S3 排错最佳实践 .....	500
重试 InternalErrors .....	500
针对重复的 SlowDown 错误调整应用程序 .....	500
隔离错误 .....	500
Amazon S3 疑难解答 .....	502
根据征兆对 Amazon S3 进行故障排除 .....	502
启用版本控制后，对存储桶请求的 HTTP 503 响应显著增加 .....	502
访问具有 CORS 设置的存储桶时出现意外行为 .....	502
获取 AWS Support 需要的 Amazon S3 请求 ID .....	502

使用 HTTP 获得请求 ID .....	503
使用 Web 浏览器获得请求 ID .....	503
使用 AWS SDK 获得请求 ID .....	503
使用 AWS CLI 获得请求 ID .....	504
相关主题 .....	504
服务器访问日志记录 .....	506
如何启用服务器访问日志记录 .....	506
日志对象键格式 .....	507
如何传输日志？ .....	507
最大努力服务器日志传输 .....	507
存储桶日志记录状态更改将逐渐生效 .....	508
使用控制台启用日志记录 .....	508
以编程方式启用日志记录 .....	508
启用日志记录 .....	508
向日志传输组授予 WRITE 和 READ_ACP 权限 .....	509
示例：适用于 .NET 的 AWS 开发工具包 .....	509
更多信息 .....	511
日志格式 .....	511
自定义访问日志信息 .....	514
可扩展服务器访问日志格式的编程注意事项 .....	514
复制操作的其他日志记录 .....	514
删除日志文件 .....	517
更多信息 .....	517
AWS 开发工具包和 Explorer .....	518
在请求身份验证中指定签名版本 .....	519
设置 AWS CLI .....	520
使用 AWS SDK for Java .....	521
Java API 组织 .....	521
测试 Amazon S3 Java 代码示例 .....	522
使用 适用于 .NET 的 AWS 开发工具包 .....	522
.NET API 组织 .....	522
运行 Amazon S3 .NET 代码示例 .....	523
使用 适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 .....	523
适用于 PHP 的 AWS 开发工具包级别 .....	523
运行 PHP 示例 .....	524
相关资源 .....	524
使用 适用于 Ruby 的 AWS 开发工具包 - 版本 3 .....	524
Ruby API 组织 .....	524
测试 Ruby 脚本示例 .....	524
使用 AWS SDK for Python (Boto) .....	525
使用适用于 iOS 和 Android 的 AWS 移动开发工具包 .....	525
更多信息 .....	525
使用 AWS Amplify JavaScript 库 .....	525
更多信息 .....	526
附录 .....	527
附录 A：使用 SOAP API .....	527
常见 SOAP API 元素 .....	527
对 SOAP 请求进行身份验证 .....	527
使用 SOAP 设置访问策略 .....	528
附录 B：对请求进行身份验证 (AWS 签名版本 2) .....	529
使用 REST API 对请求进行身份验证 .....	531
签署和对 REST 请求进行身份验证 .....	532
使用 POST 的基于浏览器的上传 .....	541
Resources .....	556
SQL 参考 .....	557
SELECT 命令 .....	557
SELECT 列表 .....	557

FROM 子句 .....	557
WHERE 子句 .....	558
LIMIT 子句 (仅限 Amazon S3 Select) .....	558
属性访问 .....	558
标头/属性名称区分大小写 .....	559
将保留关键字作为用户定义的术语 .....	559
标量表达式 .....	560
数据类型 .....	560
数据类型转换 .....	560
受支持数据类型 .....	561
运算符 .....	561
逻辑运算符 .....	561
比较运算符 .....	561
模式匹配运算符 .....	562
数学运算符 .....	562
运算符优先顺序 .....	562
保留关键字 .....	562
SQL 函数 .....	566
聚合函数 (仅限 Amazon S3 Select) .....	566
条件函数 .....	567
转换函数 .....	568
日期函数 .....	569
字符串函数 .....	574
文档历史记录 .....	577
早期更新 .....	577
AWS 词汇表 .....	588

# 什么是 Amazon S3 ?

Amazon Simple Storage Service 是互联网存储解决方案。该服务旨在降低开发人员进行网络规模级计算的难度。

Amazon S3 提供了一个简单 Web 服务接口，可用于随时在 Web 上的任何位置存储和检索任何数量的数据。此服务让所有开发人员都能访问同一个具备高扩展性、可靠性、安全性和快速价廉的数据存储基础设施，Amazon 用它来运行其全球的网站网络。此服务旨在为开发人员带来最大化的规模效益。

本指南讲解 Amazon S3 的核心概念 (如存储桶和对象) 以及如何使用 Amazon S3 应用程序编程接口 (API) 来使用这些资源。

## 如何... ?

信息	相关部分
一般产品概述和定价	<a href="#">Amazon S3</a>
获取 Amazon S3 的快速实践简介	<a href="#">Amazon Simple Storage Service 入门指南</a>
了解 Amazon S3 的主要术语和概念	<a href="#">Amazon S3 简介 (p. 2)</a>
如何使用存储桶？	<a href="#">使用 Amazon S3 存储桶 (p. 49)</a>
如何使用对象？	<a href="#">使用 Amazon S3 对象 (p. 86)</a>
如何进行请求？	<a href="#">创建请求 (p. 9)</a>
如何管理对我的资源的访问权限？	<a href="#">管理对 Amazon S3 资源的访问权限 (p. 242)</a>

# Amazon S3 简介

本 Amazon Simple Storage Service 简介旨在向您介绍此 Web 服务的详细摘要。在阅读本部分之后，您应当对 Web 服务提供的内容及其如何适应您的业务有比较深入的了解。

## 主题

- [Amazon S3 和本指南的概述 \(p. 2\)](#)
- [Amazon S3 的优势 \(p. 2\)](#)
- [Amazon S3 概念 \(p. 2\)](#)
- [Amazon S3 功能 \(p. 5\)](#)
- [Amazon S3 应用程序编程接口 \(API\) \(p. 7\)](#)
- [为 Amazon S3 付费 \(p. 8\)](#)
- [相关服务 \(p. 8\)](#)

## Amazon S3 和本指南的概述

Amazon S3 提供了一个简单 Web 服务接口，可用于随时在 Web 上的任何位置存储和检索任何数量的数据。

本指南介绍如何发送请求以创建存储桶、如何存储和检索对象，以及如何管理资源的权限。本指南还介绍访问控制和身份验证过程。访问控制定义哪些人可以访问 Amazon S3 中的对象和存储桶以及访问类型（例如，READ 和 WRITE）。身份验证过程验证尝试访问 Amazon Web Services (AWS) 的用户的身份。

## Amazon S3 的优势

Amazon S3 特意内置了着重于简易性和稳健性的最小功能集。以下是 Amazon S3 服务的一些优势：

- 创建存储桶 – 创建和命名存储数据的存储桶。存储桶是 Amazon S3 中用于数据存储的基础容器。
- 在存储桶中存储数据 – 在存储桶中存储无限量的数据。可将所需数量的对象上传到 Amazon S3 存储桶。每个对象可包含最多 5 TB 的数据。使用开发人员分配的唯一键值存储和检索每个对象。
- 下载数据 – 下载您的数据或允许其他人进行下载。随时下载您的数据或允许其他人执行相同的操作。
- 权限 - 对于要在您的 Amazon S3 存储桶中上传或下载数据的其他人员，您可以授予其访问权限或拒绝其访问。将上传和下载的许可授予三种类型的用户。身份验证机制可帮助确保数据安全，以防未授权访问。
- 标准接口 – 使用基于标准的 REST 和 SOAP 接口，它们可与任何 Internet 开发工具包搭配使用。

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

## Amazon S3 概念

## 主题

- [存储桶 \(p. 3\)](#)
- [对象 \(p. 3\)](#)
- [键 \(p. 3\)](#)
- [区域 \(p. 3\)](#)

- [Amazon S3 数据一致性模型 \(p. 3\)](#)

本部分介绍了有效使用 Amazon S3 必需了解的主要概念和术语。这些概念和术语是按照您最有可能遇到它们的顺序显示的。

## 存储桶

存储桶是 Amazon S3 中用于存储对象的容器。每个对象都储存在一个存储桶中。例如，如果名为 photos/puppy.jpg 的对象存储在 johnsmith 存储桶中，则可使用 URL <http://johnsmith.s3.amazonaws.com/photos/puppy.jpg> 对该对象进行寻址。

存储桶有以下几种用途：组织最高等级的 Amazon S3 命名空间、识别负责存储和数据传输费用的账户、在访问控制中发挥作用以及用作使用率报告的汇总单位。

您可以配置存储桶，以便在特定区域进行创建。有关详细信息，请参阅[存储桶和区域 \(p. 51\)](#)。您也可以配置存储桶，以便在每次向它添加对象时，Amazon S3 都会生成一个唯一的版本 ID 并将其分配给对象。有关详细信息，请参阅[版本控制 \(p. 380\)](#)。

有关存储桶的更多信息，请参阅[使用 Amazon S3 存储桶 \(p. 49\)](#)。

## 对象

对象是 Amazon S3 中存储的基本实体。对象由对象数据和元数据组成。数据部分对 Amazon S3 不透明。元数据是一组描述对象的名称-值对。其中包括一些默认元数据（如上次修改日期）和标准 HTTP 元数据（如 Content-Type）。您还可以在存储对象时指定自定义元数据。

在存储桶中，对象将由键（名称）和版本 ID 进行唯一地标识。有关详细信息，请参阅[键 \(p. 3\)](#)和[版本控制 \(p. 380\)](#)。

## 键

键是指存储桶中对象的唯一标识符。存储桶内的每个对象都只能有一个键。由于将存储桶、键和版本 ID 组合在一起可唯一地标识每个对象，可将 Amazon S3 视为一种“存储桶 + 键 + 版本”与对象本身间的基本数据映射。将 Web 服务终端节点、存储桶名、键和版本（可选）组合在一起，可唯一地寻址 Amazon S3 中的每个对象。例如，在 URL <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl> 中，“doc”是存储桶的名称，而“2006-03-01/AmazonS3.wsdl”是键。

有关对象键的更多信息，请参阅[对象键](#)。

## 区域

您可以选择一个地理区域供 Amazon S3 存储您创建的存储桶。您可以选择一个区域，以便优化延迟、尽可能降低成本或满足法规要求。在某一区域存储的对象将一直留在该区域，除非您特意将其传输到另一区域。例如，在欧洲（爱尔兰）区域存储的对象将一直留在该区域。

有关 Amazon S3 区域和终端节点的列表，请参阅 AWS 一般参考 中的[区域和终端节点](#)。

## Amazon S3 数据一致性模型

Amazon S3 在所有区域为 S3 存储桶中的新对象的 PUTS 提供写后读一致性，不过有一条说明。说明如下，如果在创建对象之前对键名发出 HEAD 或 GET 请求（查看该对象是否存在）Amazon S3 提供写后读最终一致性。

Amazon S3 在所有区域提供最终一致性用于覆盖 PUTS 和 Deletes。

单个键的更新是原子更新。例如，如果您对一个现有键执行 PUT 操作，则后续读取可能会返回旧数据或已更新的数据，但它永远不会返回损坏的数据或部分数据。

Amazon S3 通过在卓越亚马逊数据中心内的多个服务器之间复制数据，从而实现高可用性。如果 PUT 请求成功，则数据已安全存储。但是，有关更改的信息必须在 Amazon S3 间进行复制，这可能需要一些时间，因此您可能会观察到以下行为：

- 这是一个过程，会将一个新对象写入 Amazon S3，并立即列出其存储桶内的键。在充分传播此更改前，此对象可能不会显示在列表中。
- 这是一个过程，会替换一个现有的对象，并立即尝试读取此对象。在充分传播此更改前，Amazon S3 可能会返回先前的数据。
- 这是一个过程，会删除一个现有的对象，并立即尝试读取此对象。在充分传播此删除前，Amazon S3 可能会返回删除的数据。
- 这是一个过程，会删除一个现有的对象，并立即列出其存储桶内的键。在充分传播此删除前，Amazon S3 可能会列出删除的对象。

#### Note

Amazon S3 目前不支持对象锁定。如果同时对同一键发出两个 PUT 请求，则以带有最新时间戳的请求为准。如果这会导致问题，您需要在应用程序中创建对象锁定机制。

更新是基于键值的；无法跨键值实现原子更新。例如，无法根据一个键值的更新对另一个键值进行更新，除非将此功能设计到应用程序中。

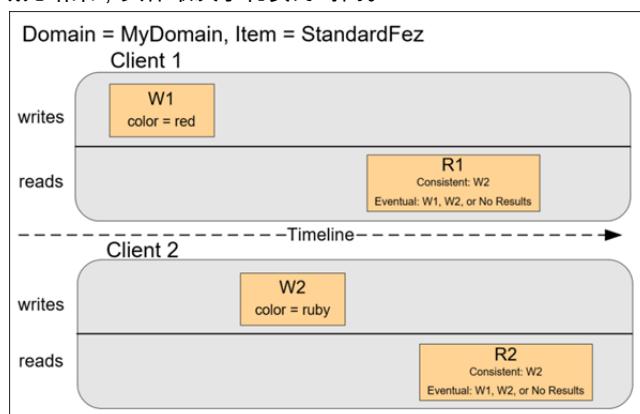
下表描述了最终一致性读取和一致性读取的特征。

最终一致性读取	一致性读取
过时读取可能性	无过时读取
最低读取延迟	潜在的更高读取延迟
最高读取吞吐量	潜在的更低读取吞吐量

## 并发应用程序

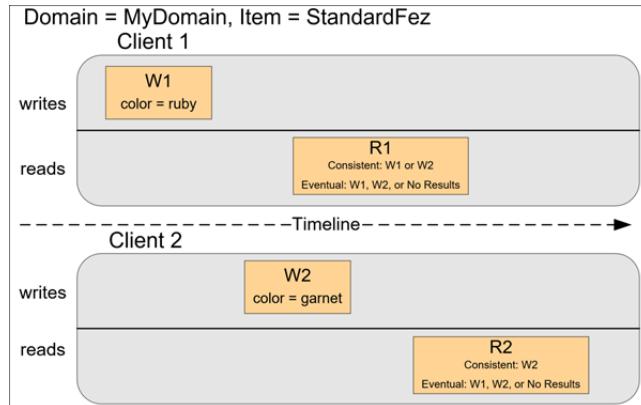
本节提供了在多个客户端写入同一项目时，最终一致性读取和一致性读取请求的示例。

在本示例中，W1（写入 1）和 W2（写入 2）会在 R1（读取 1）和 R2（读取 2）启动之前完成。为了实现一致性读取，R1 和 R2 都会返回 color = ruby。为了实现最终一致性读取，R1 和 R2 可能会返回 color = red、color = ruby 或无结果，具体取决于耗费的时间。

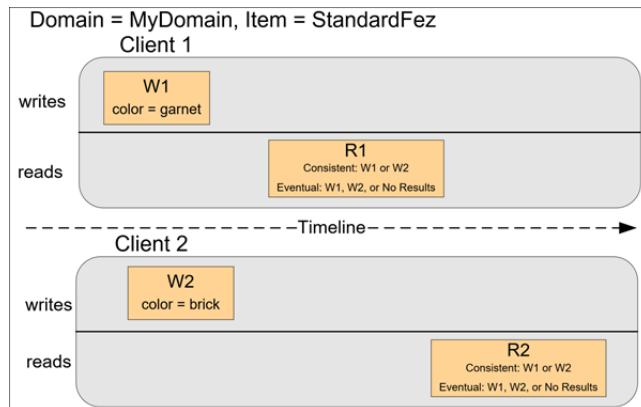


在下一个示例中，W2 不会在 R1 启动之前完成。因此，R1 可能会为一致性读取或最终一致性读取返回 color = ruby 或 color = garnet。此外，根据耗费的时间，最终一致性读取可能不会返回结果。

为了实现一致性读取，R2 会返回 color = garnet。为了实现最终一致性读取，R2 可能会返回 color = ruby、color = garnet 或无结果，具体取决于耗费的时间。



在最后一个示例中，客户端 2 会在 Amazon S3 为 W1 返回成功结果之前执行 W2，因此最终值的结果未知 (color = garnet 或 color = brick)。接下来的任意读取 (一致性读取或最终一致性读取) 可能会返回二者中的任意值。此外，根据耗费的时间，最终一致性读取可能不会返回结果。



## Amazon S3 功能

### 主题

- [存储类别 \(p. 5\)](#)
- [存储桶策略 \(p. 6\)](#)
- [AWS Identity and Access Management \(p. 6\)](#)
- [访问控制列表 \(p. 6\)](#)
- [版本控制 \(p. 7\)](#)
- [操作 \(p. 7\)](#)

本部分介绍重要的 Amazon S3 功能。

## 存储类别

Amazon S3 提供了一系列存储类，适合不同的使用情形。其中包括适用于经常访问的数据的通用型存储 (Amazon S3 STANDARD); 适用于长期需要但访问频率不太高的数据的 Amazon S3 STANDARD\_IA，以及适用于长期存档的 GLACIER。

有关更多信息，请参阅 [存储类别 \(p. 90\)](#)。

## 存储桶策略

基于各种条件，包括 Amazon S3 操作、请求者、资源和请求的其他方面（例如，IP 地址），存储桶策略可提供对存储桶和对象的集中访问控制。这些策略使用访问策略语言进行描述并且允许对许可进行集中管理。附加到存储桶的许可适用于该存储桶中的所有对象。

个人和公司都可以使用存储桶策略。在公司向 Amazon S3 进行注册时，它们会创建一个账户。然后，该公司即等同于此账户。账户在财务上承担支付公司（及其员工）所创建的 Amazon 资源的责任。账户有权基于各种条件，授予存储桶策略许可以及分配员工许可。例如，账户可以创建一个策略，授予用户对以下内容的写入权限：

- 特殊的 S3 存储桶
- 从账户的企业网络
- 在工作时间

账户不仅可以授予用户有限的读写访问权限，还可以允许其他人创建和删除存储桶。账户允许多个现场办公室将他们的日常报告存储在单个存储桶中，并使每个办公室仅可以从办公室的 IP 地址范围写入特定的名称集（例如，“Nevada/\*”或“Utah/\*”）。

与仅可以为单个对象添加（授予）许可的访问控制列表（如下所述）不同，策略可以为存储桶内的所有对象（或子集）添加或拒绝许可。在一个请求中，账户可以为存储桶内任意数量的对象设置权限。账户可以对 Amazon 资源名称（ARN）和其他值使用通配符（类似于正则表达式操作符），从而控制对以常见前缀开头或以指定扩展名（例如，.html）结尾的对象组的访问权限。

仅允许存储桶拥有者将策略与存储桶关联。采用访问策略语言编写的策略将基于以下条件允许或拒绝请求：

- Amazon S3 存储桶操作（如 `PUT ?acl`）和对象操作（如 `PUT Object` 或 `GET Object`）
- 请求者
- 在策略中指定的条件

账户可以基于特定的 Amazon S3 操作（如 `GetObject`、`GetObjectVersion`、`DeleteObject` 或 `DeleteBucket`）控制访问权限。

这些条件可以是 IP 地址、CIDR 表示中的 IP 地址范围、日期、用户代理、HTTP 引用站点和传输（HTTP 和 HTTPS）。

有关更多信息，请参阅 [使用存储桶策略和用户策略 \(p. 279\)](#)。

## AWS Identity and Access Management

例如，您可以将 IAM 用于 Amazon S3，控制用户或用户组对您的 AWS 账户所拥有 Amazon S3 存储段的具体部分的访问类型。

有关 IAM 的更多信息，请参阅下文：

- [AWS Identity and Access Management \(IAM\)](#)
- [入门](#)
- [IAM 用户指南](#)

## 访问控制列表

有关更多信息，请参阅 [使用 ACL 管理访问 \(p. 333\)](#)。

## 版本控制

有关更多信息，请参阅 [对象版本控制 \(p. 94\)](#)。

## 操作

下面是您将通过 API 执行的最常见操作。

### 常见操作

- 创建存储桶 – 创建和命名要在其中存储对象的您自己的存储桶。
- 写入对象 – 通过创建或覆盖对象存储数据。在编写对象时，在存储桶的命名空间中指定唯一键值。此时，也适合为对象指定任何您想要的访问控制。
- 读取对象 – 读回数据。您可以通过 HTTP 或 BitTorrent 下载数据。
- 删除对象 – 删除您的某些数据。
- 列出键 – 列出包含在某个存储桶中的键。您可以基于前缀筛选键列表。

有关此内容以及其他所有功能的详细信息在本指南的后续部分有详细介绍。

## Amazon S3 应用程序编程接口 (API)

Amazon S3 架构的设计与编程语言无关，它使用我们支持的接口来存储和检索对象。

Amazon S3 提供 REST 和 SOAP 接口。它们非常相似，但也有某些不同。例如，在 REST 接口中，元数据在 HTTP 标头中返回。由于我们仅支持最大 4 KB 的 HTTP 请求（不包括正文），因此您能提供的元数据量是受限的。

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

## REST 接口

REST API 是面向 Amazon S3 的 HTTP 接口。借助 REST，您可以使用标准的 HTTP 请求创建、提取和删除存储桶和对象。

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。

## SOAP 接口

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

SOAP API 使用文档文字编码来提供 SOAP 1.1 接口。使用 SOAP 的最常见方法是下载 WSDL（转到 <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl>），使用 SOAP 工具包（例如，Apache Axis 或 Microsoft .NET）创建绑定，然后编写代码以使用这些绑定调用 Amazon S3。

## 为 Amazon S3 付费

Amazon S3 定价的设计让您不必为应用程序的存储需求制定计划。大多数存储提供商都会强迫您购买预定量的存储和网络传输容量：如果超过此容量，就会关闭您的服务或对您收取高额的超容量费用。如果没有超过此容量，又要按全部容量支付使用费用。

Amazon S3 仅按照您的实际使用容量收费，没有任何隐性收费和超容量收费。这为开发人员提供了一种可变成本服务，这种服务可以伴随他们的业务共同发展，同时还可以享受 Amazon 的基础设施成本优势。

在 Amazon S3 中存储任何内容之前，您需要注册服务并提供一个每月月底付费时使用的付款方式。开始使用服务时，没有安装费。月底时，将自动通过您的付款方式扣除当月使用费。

有关为 Amazon S3 存储付费的信息，请参阅 [Amazon S3 定价](#)。

## 相关服务

一旦将数据加载 Amazon S3 中之后，便可以将其用于我们提供的其他服务。以下可能是您会频繁使用的服务：

- Amazon Elastic Compute Cloud – 此 Web 服务提供云中的虚拟计算资源。有关更多信息，请参阅 [Amazon EC2 产品详细信息页面](#)。
- Amazon EMR – 此 Web 服务可让企业、研究人员、数据分析师和开发人员轻松地、经济实惠地处理海量数据。该服务的实现利用了在 Amazon EC2 和 Amazon S3 的 Web 规模基础设施上运行的托管 Hadoop 框架。有关更多信息，请参阅 [Amazon EMR 产品详细信息页面](#)。
- AWS Import/Export - 通过 AWS Import/Export，您可以将存储设备（如 RAID 驱动器）寄至 Amazon，我们可以将您的数据 (TB 字节) 上传到 Amazon S3。有关更多信息，请转到 [AWS Import/Export Developer Guide](#)。

# 创建请求

## 主题

- [关于访问密钥 \(p. 9\)](#)
- [请求终端节点 \(p. 10\)](#)
- [通过 IPv6 向 Amazon S3 发出请求 \(p. 10\)](#)
- [使用 AWS 开发工具包创建请求 \(p. 16\)](#)
- [使用 REST API 创建请求 \(p. 41\)](#)

Amazon S3 是 REST 服务。可以使用 REST API 或是打包底层 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的 AWS 访问密钥 (访问密钥 ID 和秘密访问密钥) 生成的。有关创建访问密钥的更多信息，请参阅[如何获取安全凭证？\(在 AWS General Reference 中\)](#)。

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。

## 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，您也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。

在这些情况下，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合适的资源访问许可。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组中所有用户的组级许可。

这些用户被称为 IAM 用户，您可以在 AWS 内创建和管理这些用户。父账户将控制用户访问 AWS 的权限。父 AWS 账户负责控制和支付 IAM 用户创建的任何资源。这些 IAM 用户可以使用自己的安全凭证，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 还允许您向任何 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您的 AWS 服务和资源。您也可以在 AWS 之外的系统中管理用户。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凭证。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，您还必须在每个发送至 Amazon S3 的请求中包含令牌。

IAM 用户可以请求这些临时安全凭证以供自己使用，也可以分发给联合身份用户或应用程序使用。请求适用于联合身份用户的临时安全凭证时，您必须提供用户名和 IAM 策略（定义需要与这些临时安全凭证关联的许可）。联合身份用户可获取的许可不能超过请求临时凭证的父 IAM 用户的许可。

您可以使用上述临时安全凭证对 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。如果您使用过期凭证发送请求，Amazon S3 会拒绝该请求。

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 [签署和对 REST 请求进行身份验证 \(p. 532\)](#)。有关使用 AWS 开发工具包发送请求的信息，请参阅 [使用 AWS 开发工具包创建请求 \(p. 16\)](#)。

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 资源时进行多因素身份验证（MFA）。有关信息，请参阅 [添加存储桶策略以请求 MFA \(p. 308\)](#)。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 IAM 用户指南 中的 [AWS Multi-Factor Authentication](#) 详细信息页面和 [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 服务及其相应终端节点的列表，请参阅 AWS General Reference 中的 [区域和终端节点](#)。

## 通过 IPv6 向 Amazon S3 发出请求

除了 IPv4 协议，Amazon Simple Storage Service(Amazon S3) 还支持使用 Internet 协议版本 6 (IPv6) 访问 S3 存储桶。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](#)。

### 主题

- [开始通过 IPv6 发出请求 \(p. 10\)](#)
- [在 IAM 策略中使用 IPv6 地址 \(p. 11\)](#)
- [测试 IP 地址兼容性 \(p. 12\)](#)
- [使用 Amazon S3 双堆栈终端节点 \(p. 13\)](#)

## 开始通过 IPv6 发出请求

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 双堆栈终端节点 \(p. 13\)](#)。
- 如果您在 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 \(p. 11\)](#)。
- 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 [服务器访问日志格式 \(p. 511\)](#) 和 [Amazon S3 服务器访问日志记录 \(p. 506\)](#)。

Note

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

## 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。无论是通过 IPv6 还是 IPv4 访问 Amazon S3，Amazon S3 API 操作的工作方式都是一样的。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 [双堆栈终端节点 \(p. 13\)](#)。

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 使用双堆栈终端节点 \(p. 13\)](#)。
- AWS 开发工具包，请参阅[从 AWS 开发工具包使用双堆栈终端节点 \(p. 14\)](#)。
- REST API，请参阅[通过使用 REST API 向双堆栈终端节点发出请求 \(p. 42\)](#)。

## 在 IPv6 上不可用的功能

通过 IPv6 访问 S3 存储桶时，以下功能当前不受支持：

- S3 存储桶的静态网站托管
- BitTorrent

## 在 IAM 策略中使用 IPv6 地址

在使用 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 IAM 托管访问权限的更多信息，请参阅[管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)。

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地址范围为 54.240.143.\*IPv4。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。

```
{  
    "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 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，您必须对使用 IP 地址筛选来允许 IPv6 地址范围的所有相关 IAM 用户和存储桶策略进行更新。建议您使用组织的 IPv6 地址范围以及现有的 IPv4 地址范围来更新您的 IAM 策略。有关同时允许通过 IPv6 和 IPv4 进行访问的存储桶策略示例，请参阅[限制对特定 IP 地址的访问权限 \(p. 305\)](#)。

您可以在 <https://console.aws.amazon.com/iam/> 使用 IAM 控制台检查您的 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[如何添加 S3 存储桶策略？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

## 测试 IP 地址兼容性

如果使用 Linux/Unix 或 Mac OS X，则通过使用下例所示的 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 Windows 7，则通过使用下例所示的 ping 命令，可以测试您能否通过 IPv6 或 IPv4 访问双堆栈终端节点。

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

## 使用 Amazon S3 双堆栈终端节点

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。本节介绍如何使用双堆栈终端节点。

### 主题

- [Amazon S3 双堆栈终端节点 \(p. 13\)](#)
- [从 AWS CLI 使用双堆栈终端节点 \(p. 13\)](#)
- [从 AWS 开发工具包使用双堆栈终端节点 \(p. 14\)](#)
- [从 REST API 使用双堆栈终端节点 \(p. 15\)](#)

## Amazon S3 双堆栈终端节点

当您向双堆栈终端节点发出请求时，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 \(p. 10\)](#)

当使用 REST API 时，您通过使用终端节点名称 (URI) 直接访问 Amazon S3 终端节点。通过使用虚拟托管类型或路径类型终端节点名称，您可以通过双堆栈终端节点访问 S3 存储桶。Amazon S3 只支持区域双堆栈终端节点名称，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终端节点名称使用以下命名惯例：

- 虚拟托管型双堆栈终端节点：

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

- 路径型双堆栈终端节点：

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

有关终端节点名称类型的更多信息，请参阅[访问存储桶 \(p. 51\)](#)。有关 Amazon S3 终端节点的列表，请参阅 AWS General Reference 中的[区域和终端节点](#)。

### Important

您可对双堆栈终端节点使用传输加速。有关更多信息，请参阅[Amazon S3 Transfer Acceleration 入门 \(p. 68\)](#)。

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。下面几节介绍如何从 AWS CLI 和 AWS 开发工具包使用双堆栈终端节点。

## 从 AWS CLI 使用双堆栈终端节点

本节提供用于向双堆栈终端节点发出请求的 AWS CLI 命令示例。有关设置 AWS CLI 的说明，请参阅[设置 AWS CLI \(p. 520\)](#)。

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终端节点。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 使用双堆栈终端节点时，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）使用双堆栈终端节点，可以使用以下任一方法：

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

```
$ 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终端节点来指定要使用的配置文件。

#### Note

当使用 AWS CLI 时，当前不能对双堆栈终端节点使用传输加速。但是，即将推出对 AWS CLI 的支持。有关更多信息，请参阅 [从 AWS Command Line Interface \(AWS CLI\) 使用 Transfer Acceleration \(p. 70\)](#)。

## 从 AWS 开发工具包使用双堆栈终端节点

本节提供一些示例，介绍如何使用 AWS 开发工具包来访问双堆栈终端节点。

### AWS SDK for Java 双堆栈终端节点示例

以下示例演示如何使用 AWS SDK for Java 在创建 Amazon S3 客户端时启用双堆栈终端节点。

有关创建和测试有效 Java 示例的说明，请参阅 [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

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

public class DualStackEndpoints {

    public static void main(String[] args) {
        String clientRegion = "*** Client 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 AWS SDK for Java，可能必须设置以下 Java 虚拟机 (JVM) 属性：

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

## AWS .NET 开发工具包双堆栈终端节点示例

在使用适用于 .NET 的 AWS 开发工具包时，请使用 `AmazonS3Config` 类来启用双堆栈终端节点，如下例所示。

```
var config = new AmazonS3Config  
{  
    UseDualstackEndpoint = true,  
    RegionEndpoint = RegionEndpoint.USWest2  
};  
  
using (var s3Client = new AmazonS3Client(config))  
{  
    var request = new ListObjectsRequest  
    {  
        BucketName = "myBucket"  
    };  
  
    var response = await s3Client.ListObjectsAsync(request);  
}
```

有关所列对象的完整 .NET 示例，请参阅[使用适用于 .NET 的 AWS 开发工具包 列出键 \(p. 200\)](#)。

有关如何创建和测试有效 .NET 示例的信息，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

## 从 REST API 使用双堆栈终端节点

有关如何使用 REST API 向双堆栈终端节点发出请求的信息，请参阅[通过使用 REST API 向双堆栈终端节点发出请求 \(p. 42\)](#)。

# 使用 AWS 开发工具包创建请求

## 主题

- 使用 AWS 账户或 IAM 用户凭证进行请求 (p. 16)
- 使用 IAM 用户临时凭证进行请求 (p. 23)
- 使用联合身份用户临时凭证创建请求 (p. 31)

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 用户凭证进行请求

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 AWS SDK for Java、适用于 .NET 的 AWS 开发工具包 和 适用于 PHP 的 AWS 开发工具包 发送经身份验证的请求。有关可用 AWS 开发工具包的列表，请转到[示例代码和库](#)。

## 主题

- 使用 AWS 账户或 IAM 用户凭证进行请求 - AWS SDK for Java (p. 17)
- 使用 AWS 账户或 IAM 用户凭证进行请求 - 适用于 .NET 的 AWS 开发工具包 (p. 18)
- 使用 AWS 账户或 IAM 用户凭证进行请求 - 适用于 PHP 的 AWS 开发工具包 (p. 20)
- 使用 AWS 账户或 IAM 用户凭证进行请求 – 适用于 Ruby 的 AWS 开发工具包 (p. 21)

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。

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：

### 创建本地 AWS 凭证文件

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。
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 IAM 用户的更多信息，请参阅[创建 IAM 用户 \(控制台\)](#)，并按照步骤 8 中的说明进行操作。
3. 选择 Download .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 账户或 IAM 用户凭证进行请求 - AWS SDK for Java

要使用 AWS 账户或 IAM 用户凭证向 Amazon S3 发出经身份验证的请求，请执行以下操作：

- 使用 `AmazonS3ClientBuilder` 类创建 `AmazonS3Client` 实例。
- 执行 `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请求中。

以下示例将执行上述任务。有关创建和测试有效示例的信息，请参阅 [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

### Example

```
import java.io.IOException;
import java.util.List;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class MakingRequests {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Get a list of objects in the bucket, two at a time, and
            // print the name and size of each object.
            ListObjectsRequest listRequest = new
ListObjectsRequest().withBucketName(bucketName).withMaxKeys(2);
            ObjectListing objects = s3Client.listObjects(listRequest);
            while(true) {
                List<S3ObjectSummary> summaries = objects.getObjectSummaries();
                for(S3ObjectSummary summary : summaries) {
                    System.out.printf("Object \"%s\" retrieved with size %d\n",
summary.getKey(), summary.getSize());
                }
                if(objects.isTruncated()) {
                    objects = s3Client.listNextBatchOfObjects(objects);
                }
                else {
                    break;
                }
            }
        } catch(AmazonServiceException e) {
            System.out.println("Caught an AmazonServiceException, which " +
"means your request was well-formed but failed for other reasons. " +
"Error Message: " + e.getMessage());
        } catch(SdkClientException e) {
            System.out.println("Caught an SdkClientException, which " +
"means the client could not understand the response from " +
"Amazon S3. Error Message: " + e.getMessage());
        }
    }
}
```

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

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 使用 AWS 账户或 IAM 用户凭证进行请求 - 适用于 .NET 的 AWS 开发工具包

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。
- 执行 `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其发送到 Amazon S3 的请求中。

以下 C# 示例说明如何执行上述任务。有关运行本指南中的 .NET 示例的信息以及有关如何将凭证存储在配置文件中的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

### Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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);
    }
}
}
```

#### 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。如果针对非公开可用的资源发送匿名请求，Amazon S3 将返回错误。

有关有效示例，请参阅 [使用 Amazon S3 对象 \(p. 86\)](#) 和 [使用 Amazon S3 存储桶 \(p. 49\)](#)。您可以使用 AWS 账户或 IAM 用户凭证测试这些示例。

例如，要列出您的存储桶中的所有对象键，请参阅 [使用适用于 .NET 的 AWS 开发工具包 列出键 \(p. 200\)](#)。

## 相关资源

- [使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)

## 使用 AWS 账户或 IAM 用户凭证进行请求 - 适用于 PHP 的 AWS 开发工具包

此部分说明如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类来通过 AWS 账户或 IAM 用户凭证发送经身份验证的请求。此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并且已正确安装适用于 PHP 的 AWS 开发工具包。

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

```
<?php

require 'vendor/autoload.php';

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

$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。

有关有效示例，请参阅[在对象上的操作 \(p. 141\)](#)。您可以使用 AWS 账户或 IAM 用户凭证测试这些示例。

有关列出存储桶中的对象键的示例，请参阅[使用适用于 PHP 的 AWS 开发工具包 列出键 \(p. 201\)](#)。

### 相关资源

- [用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 `Aws\S3\S3Client` 类](#)
- [适用于 PHP 的 AWS 开发工具包文档](#)

## 使用 AWS 账户或 IAM 用户凭证进行请求 – 适用于 Ruby 的 AWS 开发工具包

在您可以使用适用于 Ruby 的 AWS 开发工具包的版本 3 调用 Amazon S3 之前，必须设置 AWS 访问凭证以便 SDK 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 用户凭证进行请求 \(p. 16\)](#)。

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

1. 创建 `Aws::S3::Resource` 类的实例。
2. 使用 `Aws::S3::Resource` 的 `bucket` 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. 将对象键名称阵列打印到终端。

### Example

```
# Use the Amazon S3 modularized gem for version 3 of the AWS Ruby SDK.
require 'aws-sdk-s3'

# Get an Amazon S3 resource.
s3 = Aws::S3::Resource.new(region: 'us-west-2')

# Create an array of up to the first 100 object keynames in the bucket.
bucket = s3.bucket('example_bucket').objects.collect(&:key)

# Print the array to the terminal.
puts bucket
```

即使没有本地 AWS 凭证文件，您仍可以创建 `Aws::S3::Resource` 资源并针对 Amazon S3 存储桶和对象执行代码。使用适用于 Ruby 的开发工具包版本 3 发送的请求是匿名的，默认情况下没有签名。如果针对非公开可用的资源发送匿名请求，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。用于此示例的凭证来自运行此应用程序的计算机上的本地 AWS 凭证文件。这些凭证适用于可以在运行此应用程序时列出自己指定的存储桶中的对象的 IAM 用户。

```
# auth_request_test.rb
# Use the Amazon S3 modularized gem for version 3 of the AWS Ruby SDK.
require 'aws-sdk-s3'

# Usage: ruby auth_request_test.rb list BUCKET

# Set the name of the bucket on which the operations are performed.
# This argument is required
bucket_name = nil

# The operation to perform on the bucket.
operation = 'list' # default
operation = ARGV[0] if (ARGV.length > 0)

if ARGV.length > 1
  bucket_name = ARGV[1]
else
  exit 1
end
```

```
# Get an Amazon S3 resource.  
s3 = Aws::S3::Resource.new(region: 'us-west-2')  
  
# Get the bucket by name.  
bucket = s3.bucket(bucket_name)  
  
case operation  
  
when 'list'  
  if bucket.exists?  
    # Enumerate the bucket contents and object etags.  
    puts "Contents of '%s':" % bucket_name  
    puts '  Name => GUID'  
  
    bucket.objects.limit(50).each do |obj|  
      puts "    #{obj.key} => #{obj.etag}"  
    end  
  else  
    puts "The bucket '%s' does not exist!" % bucket_name  
  end  
  
else  
  puts "Unknown operation: '%s'! Only list is supported." % operation  
end
```

# 使用 IAM 用户临时凭证进行请求

## 主题

- 使用 IAM 用户临时凭证创建请求 - AWS SDK for Java (p. 23)
- 使用 IAM 用户临时凭证创建请求 - 适用于 .NET 的 AWS 开发工具包 (p. 25)
- 使用 AWS 账户或 IAM 用户临时凭证进行请求 - 适用于 PHP 的 AWS 开发工具包 (p. 27)
- 使用 IAM 用户临时凭证进行请求 - 适用于 Ruby 的 AWS 开发工具包 (p. 28)

AWS 账户或 IAM 用户可以请求临时安全凭证，然后使用它们向 Amazon S3 发送经身份验证的请求。本部分提供了一些示例，介绍如何使用 AWS SDK for Java、.NET 和 PHP 获取临时安全凭证，以及如何使用这些凭证对您发送至 Amazon S3 的请求进行身份验证。

## 使用 IAM 用户临时凭证创建请求 - AWS SDK for Java

IAM 用户或 AWS 账户可以使用 AWS SDK for Java 请求临时安全凭证（请参阅[创建请求 \(p. 9\)](#)），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。要使用 IAM 临时安全凭证，请执行以下操作：

1. 创建 `AWSSecurityTokenServiceClient` 类的实例。有关提供凭证的信息，请参阅[使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)。
2. 通过调用安全令牌服务 (STS) 客户端的 `assumeRole()` 方法代入所需角色。
3. 通过调用 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。  
此方法将返回临时安全凭证。
4. 将临时安全凭证打包到 `BasicSessionCredentials` 对象中。您可以使用此对象向您的 Amazon S3 客户端提供临时安全凭证。
5. 使用临时安全凭证创建 `AmazonS3Client` 类的实例。可使用此客户端向 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 用户凭证请求会话时，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。该示例将为两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。

如果您想要使用 IAM 用户凭证测试示例，需要在您的 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[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
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.AWSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.Credentials;
```

```
import com.amazonaws.services.securitytoken.model.GetSessionTokenRequest;
import com.amazonaws.services.securitytoken.model.GetSessionTokenResult;

public class MakingRequestsWithIAMPtempCredentials {
    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();

            // Assume 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);
            stsClientassumeRole(roleRequest);

            // Start a session.
            GetSessionTokenRequest getSessionTokenRequest = new GetSessionTokenRequest();
            // The duration can be set to more than 3600 seconds only if temporary
            // credentials are requested by an IAM user rather than an account owner.
            getSessionTokenRequest.setDurationSeconds(7200);
            GetSessionTokenResult sessionTokenResult =
                stsClient.getSessionToken(getSessionTokenRequest);
            Credentials sessionCredentials = sessionTokenResult.getCredentials();

            // Package the temporary security credentials as a BasicSessionCredentials
            object
            // for an Amazon S3 client object to use.
            BasicSessionCredentials basicSessionCredentials = 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 basicSessionCredentials object.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new
                    AWSStaticCredentialsProvider(basicSessionCredentials))
                .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();
}
}
```

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 使用 IAM 用户临时凭证创建请求 - 适用于 .NET 的 AWS 开发工具包

IAM 用户或 AWS 账户可以使用适用于 .NET 的 AWS 开发工具包请求临时安全凭证，然后使用这些凭证访问 Amazon S3。在会话持续时间结束后，这些凭证将过期。要获取临时安全凭证并访问 Amazon S3，请执行以下操作：

1. 创建 AWS Security Token Service 客户端 `AmazonSecurityTokenServiceClient` 的实例。有关提供凭证的信息，请参阅[使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)。
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。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。

### 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。为方便说明，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。

如果要使用 IAM 用户凭证测试示例，则需要在 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[创建请求 \(p. 9\)](#)。

有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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;
        }
    }
}
```

```
        }
    }
}
```

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 使用 AWS 账户或 IAM 用户临时凭证进行请求 - 适用于 PHP 的 AWS 开发工具包

本主题介绍如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类请求临时安全凭证并使用这些凭证访问 Amazon S3。本主题假定已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作并正确安装了适用于 PHP 的 AWS 开发工具包。

IAM 用户或 AWS 账户可使用版本 3 的适用于 PHP 的 AWS 开发工具包请求临时安全凭证。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 (1 到 36 小时)。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[创建请求 \(p. 9\)](#)。

### 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仅为一个小时。只有当您使用 IAM 用户凭证请求会话时，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。有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

如果要使用 IAM 用户凭证测试示例，则需要在 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[使用联合身份用户临时凭证创建请求 - 适用于 PHP 的 AWS 开发工具包 \(p. 36\)](#)。

```
<?php

require 'vendor/autoload.php';

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

$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' => [
        'accessKeyId' => $awsAccessKeyId,
        'secretAccessKey' => $awsSecretAccessKey,
        'sessionToken' => $sessionToken
    ]
]);

$objects = $s3->listObjects(['Bucket' => $bucket]);
foreach ($objects['Contents'] as $object) {
    echo $object['Key'];
}
```

```
'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;
}
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 [Aws\S3\S3Client](#) 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用 IAM 用户临时凭证进行请求 - 适用于 Ruby 的 AWS 开发工具包

IAM 用户或 AWS 账户可以使用适用于 Ruby 的 AWS 开发工具包 请求临时安全凭证，然后使用这些凭证访问 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（1 到 36 小时）。有关请求临时安全凭证的信息，请参阅[创建请求 \(p. 9\)](#)。

### 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仅为一个小时。只有当您使用 IAM 用户凭证请求会话时，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。

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

USAGE = <<DOC
Usage: assumerole_create_bucket_policy.rb -b BUCKET -u USER [-r REGION] [-d] [-h]

Assumes a role for USER to list items in BUCKET for one hour.

BUCKET is required and must already exist.
DOC
```

```
USER is required and if not found, is created.  
If REGION is not supplied, defaults to us-west-2.  
-d gives you extra (debugging) information.  
-h displays this message and quits.  
  
DOC  
  
$debug = false  
  
def print_debug(s)  
  if $debug  
    puts s  
  end  
end  
  
def get_user(region, user_name, create)  
  user = nil  
  iam = Aws::IAM::Client.new(region: 'us-west-2')  
  
  begin  
    user = iam.create_user(user_name: user_name)  
    iam.wait_until(:user_exists, user_name: user_name)  
    print_debug("Created new user #{user_name}")  
  rescue Aws::IAM::Errors::EntityAlreadyExists  
    print_debug("Found user #{user_name} in region #{region}")  
  end  
end  
  
# main  
region = 'us-west-2'  
user_name = ''  
bucket_name = ''  
  
i = 0  
  
while i < ARGV.length  
  case ARGV[i]  
    when '-b'  
      i += 1  
      bucket_name = ARGV[i]  
    when '-u'  
      i += 1  
      user_name = ARGV[i]  
    when '-r'  
      i += 1  
      region = ARGV[i]  
    when '-d'  
      puts 'Debugging enabled'  
      $debug = true  
    when '-h'  
      puts USAGE  
      exit 0  
  else  
    puts 'Unrecognized option: ' + ARGV[i]  
    puts USAGE  
    exit 1
```

```
end

i += 1
end

if bucket_name == ''
  puts 'You must supply a bucket name'
  puts USAGE
  exit 1
end

if user_name == ''
  puts 'You must supply a user name'
  puts USAGE
  exit 1
end

#Identify the IAM user that is allowed to list Amazon S3 bucket items for an hour.
user = get_user(region, user_name, true)

# Create a new Amazon STS client and get temporary credentials. This uses a role that was
already created.
creds = Aws::AssumeRoleCredentials.new(
  client: Aws::STS::Client.new(region: region),
  role_arn: "arn:aws:iam::111122223333:role/assumedrolelist",
  role_session_name: "assumerole-s3-list"
)

# Create an Amazon S3 resource with temporary credentials.
s3 = Aws::S3::Resource.new(region: region, credentials: creds)

puts "Contents of '%s':" % bucket_name
puts '  Name => GUID'

s3.bucket(bucket_name).objects.limit(50).each do |obj|
  puts "    #{obj.key} => #{obj.etag}"
end
```

## 使用联合身份用户临时凭证创建请求

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。本部分提供了一些示例，演示如何使用 AWS 开发工具包获取适用于联合身份用户或应用程序的临时安全凭证并使用这些凭证向 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凭证。然而，为了提高安全性，只有具有所需权限的 IAM 用户才能请求这些临时凭证，这样可确保联合身份用户最多获取进行请求的 IAM 用户的权限。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。

## 使用联合身份用户临时凭证创建请求 - AWS SDK for 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 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权限不能超过请求临时安全凭证的 IAM 用户。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。有关提供凭证的信息，请参阅[使用 AWS SDK for Java \(p. 521\)](#)。
- 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。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

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[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
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.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.securitytoken.AWSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;
import com.amazonaws.services.s3.model.ObjectListing;

public class MakingRequestsWithFederatedTempCredentials {

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

        try {
            AWSecurityTokenService stsClient = AWSecurityTokenServiceClientBuilder
                .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();
}
}

}
```

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 使用联合身份用户临时凭证创建请求 - 适用于 .NET 的 AWS 开发工具包

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 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[创建请求 \(p. 9\)](#)。

### 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权限不能超过请求临时安全凭证的 IAM 用户。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。有关提供凭证的信息，请参阅[使用适用于 .NET 的 AWS 开发工具包 \(p. 522\)](#)。
- 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 [创建您的第一个 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凭证向 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。有关如何创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
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;
}

}
}
```

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 使用联合身份用户临时凭证创建请求 - 适用于 PHP 的 AWS 开发工具包

本主题介绍如何使用版本 3 的适用于 PHP 的 AWS 开发工具包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。本主题假定已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作并正确安装了适用于 PHP 的 AWS 开发工具包。

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 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[创建请求 \(p. 9\)](#)。

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权限不能超过请求临时安全凭证的 IAM 用户。有关联合身份的信息，请参阅[AWS Identity and Access Management 常见问题](#)。

有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

### 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 用户安全凭证测试下面的示例。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：

```
<?php  
  
require 'vendor/autoload.php';  
  
use Aws\Sts\StsClient;  
use Aws\S3\S3Client;  
use Aws\S3\Exception\S3Exception;  
  
$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;
}
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 Aws\S3\S3Client 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用联合身份用户临时凭证创建请求 - 适用于 Ruby 的 AWS 开发工具包

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 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，如果使用 IAM 用户凭证来请求临时凭证，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，您可以显式地设置其他持续时间值。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[创建请求 \(p. 9\)](#)。

### 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权限不能超过请求临时安全凭证的 IAM 用户。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。

```
require 'aws-sdk-s3'
require 'aws-sdk-iam'

USAGE = <<DOC
Usage: federated_create_bucket_policy.rb -b BUCKET -u USER [-r REGION] [-d] [-h]

Creates a federated policy for USER to list items in BUCKET for one hour.

BUCKET is required and must already exist.

USER is required and if not found, is created.

If REGION is not supplied, defaults to us-west-2.

-d gives you extra (debugging) information.

-h displays this message and quits.

DOC

$debug = false

def print_debug(s)
  if $debug
    puts s
  end
end

def get_user(region, user_name, create)
  user = nil
  iam = Aws::IAM::Client.new(region: 'us-west-2')

begin
  user = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  print_debug("Created new user #{user_name}")
rescue Aws::IAM::Errors::EntityAlreadyExists
  print_debug("Found user #{user_name} in region #{region}")
end
end

# main
```

```
region = 'us-west-2'
user_name = ''
bucket_name = ''

i = 0

while i < ARGV.length
  case ARGV[i]

    when '-b'
      i += 1
      bucket_name = ARGV[i]

    when '-u'
      i += 1
      user_name = ARGV[i]

    when '-r'
      i += 1

      region = ARGV[i]

    when '-d'
      puts 'Debugging enabled'
      $debug = true

    when '-h'
      puts USAGE
      exit 0

    else
      puts 'Unrecognized option: ' + ARGV[i]
      puts USAGE
      exit 1
  end

  i += 1
end

if bucket_name == ''
  puts 'You must supply a bucket name'
  puts USAGE
  exit 1
end

if user_name == ''
  puts 'You must supply a user name'
  puts USAGE
  exit 1
end

#Identify the IAM user we allow to list Amazon S3 bucket items for an hour.
user = get_user(region, user_name, true)

# Create a new STS client and get temporary credentials.
sts = Aws::STS::Client.new(region: region)

creds = sts.get_federation_token({
  duration_seconds: 3600,
  name: user_name,
  policy: "{\"Version\":\"2012-10-17\", \"Statement\":[{\"Sid\":\"Stmt1\", \"Effect\":\"Allow\", \"Action\":\"s3:ListBucket\", \"Resource\":\"arn:aws:s3:::#{bucket_name}\"}]}",
})

# Create an Amazon S3 resource with temporary credentials.
```

```
s3 = Aws::S3::Resource.new(region: region, credentials: creds)

puts "Contents of '%s':\n" % bucket_name
puts '  Name => GUID'

s3.bucket(bucket_name).objects.limit(50).each do |obj|
  puts "    #{obj.key} => #{obj.etag}"
end
```

## 使用 REST API 创建请求

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 终端节点的列表，请参阅 AWS General Reference 中的[区域和终端节点](#)。

### 主题

- [通过使用 REST API 向双堆栈终端节点发出请求 \(p. 42\)](#)
- [存储桶的虚拟托管 \(p. 42\)](#)
- [请求重定向和 REST API \(p. 46\)](#)

在使用 REST API 创建请求时，您可以使用 Amazon S3 终端节点的虚拟托管类型和路径类型 URI。有关更多信息，请参阅[使用 Amazon S3 存储桶 \(p. 49\)](#)。

### Example 虚拟托管类型请求

下面是要求从名为 examplebucket 的存储桶删除 puppy.jpg 文件的虚拟托管类型请求的示例。

```
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
```

Amazon S3 支持在所有区域进行虚拟托管式和路径式访问。但是，路径式语法要求您在尝试访问存储段时必须使用特定于地区的终端节点。例如，如果您有一个位于欧洲（爱尔兰）区域的名为 mybucket 的存储桶，您希望使用路径式语法并且对象名为 puppy.jpg，则正确的 URI 为 `http://s3-eu-west-1.amazonaws.com/mybucket/puppy.jpg`。

您将收到一个“HTTP 响应代码 307 临时重定向”错误和一条消息，该消息指出当您尝试使用含有以下一种内容的路径式语法来访问美国东部（弗吉尼亚北部）区域之外的存储桶时，正确的资源 URI 为：

- `http://s3.amazonaws.com`
- 区域的终端节点不同于存储桶所在的终端节点。例如，对在 美国西部（加利福尼亚北部）区域创建的存储桶使用 `http://s3-eu-west-1.amazonaws.com`。

## 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，您也可以在请求中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 双堆栈终端节点 \(p. 13\)](#)。

## 存储桶的虚拟托管

### 主题

- [HTTP 主机标头存储桶规范 \(p. 43\)](#)
- [示例 \(p. 43\)](#)
- [使用别名记录自定义 Amazon S3 URL \(p. 45\)](#)
- [限制 \(p. 46\)](#)
- [向后兼容 \(p. 46\)](#)

一般而言，虚拟托管是指从单个 Web 服务器为多个 Web 站点提供服务的做法。区分站点的一种方法是通过使用请求的显式主机名，而不只是 URI 的路径名称部分。普通的 Amazon S3 REST 请求使用“请求-URI”路径的第一个斜杠分隔部分指定存储桶。或者，您可以通过使用 HTTP Host 标头，在 REST API 调用中使用 Amazon S3 虚拟托管对存储桶进行寻址。事实上，Amazon S3 会将 Host 解释为可在 `http://bucketname.s3.amazonaws.com` 上自动访问大多数存储桶（对于有限类型的请求）。此外，通过使用注册域名来命名您的存储桶，并将该名称设为 Amazon S3 的 DNS 别名，您可以完全自定义 Amazon S3 资源的 URL，例如：`http://my.bucketname.com/`。

除了自定义 URL 的好处外，虚拟托管的另一个好处是能够发布到存储桶的虚拟服务器的“根目录”。此功能很重要，因为许多现有应用程序在此标准位置搜索文件。例如，`favicon.ico`、`robots.txt`、`crossdomain.xml` 都应在根目录下。

### Important

Amazon S3 支持在所有区域进行虚拟托管式和路径式访问。但是，路径式语法要求您在尝试访问存储段时必须使用特定于地区的终端节点。例如，如果您有一个位于欧洲（爱尔兰）区域的名为 `mybucket` 的存储桶，您希望使用路径式语法并且对象名为 `puppy.jpg`，则正确的 URI 为 `http://s3-eu-west-1.amazonaws.com/mybucket/puppy.jpg`。

您将收到一个“HTTP 响应代码 307 临时重定向”错误和一条消息，该消息指出当您尝试使用含有以下一种内容的路径式语法来访问美国东部（弗吉尼亚北部）区域之外的存储桶时，正确的资源 URI 为：

- `http://s3.amazonaws.com`
- 区域的终端节点不同于存储桶所在的终端节点。例如，对在 美国西部（加利福尼亚北部）区域创建的存储桶使用 `http://s3-eu-west-1.amazonaws.com`。

#### Note

如果您使用美国东部（弗吉尼亚北部）终端节点 (`s3.amazonaws.com`)，而不是区域特定的终端节点（例如 `s3-eu-west-1`），则在默认情况下 Amazon S3 会将任何虚拟托管式请求路由到美国东部（弗吉尼亚北部）区域。`amazonaws.com`。当您在任何区域创建存储桶时，Amazon S3 都将更新 DNS 以将请求重新路由到正确的位置，这可能会花费一些时间。同时，将应用默认规则，您的虚拟托管式请求也会进入美国东部（弗吉尼亚北部）区域，Amazon S3 将使用 HTTP 307 redirect 将其重定向到正确的区域。有关更多信息，请参阅 [请求重定向和 REST API \(p. 476\)](#)。

当通过 SSL 使用虚拟托管式存储桶时，SSL 通配符证书仅匹配不包含句点的存储桶。要解决此问题，请使用 HTTP 或编写自己的证书验证逻辑。

## HTTP 主机标头存储桶规范

只要您的 GET 请求不使用 SSL 终端节点，您就可以通过使用 HTTP Host 标头为请求指定存储桶。REST 请求中的 Host 标头解释如下：

- 如果 Host 标头被省略，或其值为“`s3.amazonaws.com`”，则该请求的存储桶将成为“Request-URI”的第一个斜杠分隔的部分，而且该请求的键值将成为“Request-URI”的其余部分。这是常用方法，如本节第一个和第二个示例所示。省略主机标头仅对于 HTTP 1.0 请求有效。
- 否则，如果 Host 标头的值以“`.s3.amazonaws.com`”结尾，则存储桶名称是 Host 标头值中从开头到“`.s3.amazonaws.com`”的部分。该请求的键值为“Request-URI”。此解释将存储桶公开为 `s3.amazonaws.com` 的子域，如本节第三个和第四个示例所示。
- 否则，请求的存储桶是 Host 标头的小写值，请求的键值是“Request-URI”。如果注册的 DNS 名称与存储桶名称相同，并且已将该名称配置为 Amazon S3 的别名记录 (CNAME) 别名，则此解释很有用。注册域名和配置 DNS 的过程不在本指南的范围内，但本节中的最后一个示例展示了结果。

## 示例

本节提供示例 URL 和请求。

### Example 路径类型的方法

本示例使用 `johnsmith.net` 作为存储桶名称，使用 `homepage.html` 作为密钥名称。

URL 如下：

```
http://s3.amazonaws.com/johnsmith.net/homepage.html
```

请求如下：

```
GET /johnsmith.net/homepage.html HTTP/1.1
Host: s3.amazonaws.com
```

具有 HTTP 1.0 且省略 host 标头的请求如下：

```
GET /johnsmith.net/homepage.html HTTP/1.0
```

有关 DNS 兼容名称的信息，请参阅[限制 \(p. 46\)](#)。有关键值的更多信息，请参阅[键值 \(p. 3\)](#)。

#### Example 虚拟托管式方法

本示例使用 `johnsmith.net` 作为存储桶名称，使用 `homepage.html` 作为密钥名称。

URL 如下：

```
http://johnsmith.net.s3.amazonaws.com/homepage.html
```

请求如下：

```
GET /homepage.html HTTP/1.1
Host: johnsmith.net.s3.amazonaws.com
```

虚拟托管式方法要求存储桶名称符合 DNS 要求。

#### Example 针对非美国东部 (弗吉尼亚北部) 区域内的存储桶的虚拟托管式方法

本示例使用 `johnsmith.eu` 作为 欧洲 (爱尔兰) 区域中存储桶的名称，并使用 `homepage.html` 作为键名称。

URL 如下：

```
http://johnsmith.eu.s3-eu-west-1.amazonaws.com/homepage.html
```

请求如下：

```
GET /homepage.html HTTP/1.1
Host: johnsmith.eu.s3-eu-west-1.amazonaws.com
```

注意，无论存储桶位于哪个区域，您都可使用美国东部 (弗吉尼亚北部) 区域终端节点替代区域特定的终端节点。

```
http://johnsmith.eu.s3.amazonaws.com/homepage.html
```

请求如下：

```
GET /homepage.html HTTP/1.1
Host: johnsmith.eu.s3.amazonaws.com
```

#### Example 别名记录方法

本示例使用 `www.johnsmith.net` 作为存储桶名称，使用 `homepage.html` 作为密钥名称。要使用此方法，您必须配置 DNS 名称作为 `bucketname.s3.amazonaws.com` 的别名记录别名。

URL 如下：

```
http://www.johnsmith.net/homepage.html
```

示例如下：

```
GET /homepage.html HTTP/1.1
Host: www.johnsmith.net
```

## 使用别名记录自定义 Amazon S3 URL

根据您的需要，您可能不希望“s3.amazonaws.com”显示在网站或服务中。例如，如果在 Amazon S3 上托管网站映像，您可能会首选 `http://images.johnsmith.net/`，而不是 `http://johnsmith-images.s3.amazonaws.com/`。

存储桶名称必须与 CNAME 相同。因此，如果创建了别名记录，以便将 `http://images.johnsmith.net/filename` 映射到 `http://images.johnsmith.net.s3.amazonaws.com/filename`，则 `images.johnsmith.net` 将与 `images.johnsmith.net.s3.amazonaws.com` 相同。

具有 DNS 兼容名称的任何存储桶都可按以下方式引用：`http://[BucketName].s3.amazonaws.com/[Filename]`，例如 `http://images.johnsmith.net.s3.amazonaws.com/mydog.jpg`。通过使用别名记录，可将 `images.johnsmith.net` 映射到 Amazon S3 主机名，因此上述 URL 可以变为 `http://images.johnsmith.net/mydog.jpg`。

别名记录 DNS 记录应将您的域名别名设置为适当的虚拟托管式主机名。例如，如果存储桶名称和域名是 `images.johnsmith.net`，则别名记录应将别名设置为 `images.johnsmith.net.s3.amazonaws.com`。

```
images.johnsmith.net CNAME images.johnsmith.net.s3.amazonaws.com.
```

也可以针对 `s3.amazonaws.com` 设置别名，但可能导致额外的 HTTP 重定向。

Amazon S3 使用主机名来确定存储桶名称。例如，假定您配置了 `www.example.com` 作为 `www.example.com.s3.amazonaws.com` 的别名记录。当您访问 `http://www.example.com` 时，Amazon S3 会收到如下所示的请求：

### Example

```
GET / HTTP/1.1
Host: www.example.com
Date: date
Authorization: signatureValue
```

因为 Amazon S3 仅看到原始主机名 `www.example.com`，不知道用于解析请求的别名记录映射，所以别名记录和存储桶名称必须相同。

可在别名记录中使用任何 Amazon S3 终端节点。例如，`s3-ap-southeast-1.amazonaws.com` 可用于 CNAME 中。有关终端节点的更多信息，请参阅[请求终端节点 \(p. 10\)](#)。

### 使用别名记录将主机名与 Amazon S3 存储桶关联

1. 选择属于您控制的域的主机名。本示例使用 `images` 域的 `johnsmith.net` 子域。
2. 创建与主机名匹配的存储桶。在本示例中，主机名和存储桶名称是 `images.johnsmith.net`。

#### Note

存储桶名称必须与主机名精确匹配。

3. 创建一条 CNAME 记录，它定义主机名作为 Amazon S3 存储桶的别名。例如：

```
images.johnsmith.net CNAME images.johnsmith.net.s3.amazonaws.com
```

#### Important

出于请求路由原因，别名记录必须严格按照上述示例所示进行定义。否则，它可能显示为运行正常，但最终导致不可预测的行为。

#### Note

配置 DNS 的过程取决于您的 DNS 服务器或 DNS 提供商。有关特定信息，请参阅您的服务器文档或与您的供应商联系。

## 限制

对于非 SSL 请求以及在使用 REST API 时，支持使用 HTTP Host 标头为请求指定存储桶。您无法使用不同的终端节点在 SOAP 中指定存储桶。

#### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

## 向后兼容

Amazon S3 的早期版本错误地忽略了 HTTP Host 标头。必须更新依赖此未记录行为的应用程序才能正确设置 Host 标头。因为当 Host 存在时，Amazon S3 将据其确定存储桶名称，所以此问题最有可能的表现是收到意外的 NoSuchBucket 错误结果代码。

## 请求重定向和 REST API

### 主题

- [重定向和 HTTP 用户代理 \(p. 46\)](#)
- [重定向和 100-继续 \(p. 46\)](#)
- [重定向示例 \(p. 47\)](#)

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 Reference 中的 [请求重定向和 REST API \(p. 476\)](#)。

## 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-继续响应，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-继续

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-继续。当应用程序使用 100-继续时，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-继续的更多信息，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-继续。但是，如果您的请求包含了 `Expect: Continue`，Amazon S3 能够进行识别并且将使用临时的 100-继续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-继续执行客户端和服务器交互的示例。

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-继续，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
```

# 使用 Amazon S3 存储桶

Amazon S3 是一种 Internet 上的云存储服务。要上传数据（照片、视频、文档等），请首先在一个 AWS 区域中创建存储桶。然后，您可以将任何数量的对象上传到该存储桶。

对于实施，存储桶和对象是资源，而 Amazon S3 提供 API 供您管理资源。例如，您可以使用 Amazon S3 API 创建存储桶并上传对象。您还可以使用 Amazon S3 控制台执行这些操作。该控制台使用 Amazon S3 API 将请求发送到 Amazon S3。

本节介绍如何使用存储桶。有关使用对象的信息，请参阅[使用 Amazon S3 对象 \(p. 86\)](#)。

Amazon S3 存储桶名称是全局唯一的，无论在哪个 AWS 区域中创建存储桶都是如此。在创建存储桶时指定名称。有关存储桶命名指南，请参阅[存储桶限制 \(p. 53\)](#)。

Amazon S3 可在指定的区域中创建存储桶。为优化延迟、最大限度地降低成本或满足法规要求，请选择地理上靠近您的任何 AWS 区域。例如，如果您位于欧洲，您可能会发现在欧洲（爱尔兰）或欧洲（法兰克福）区域创建存储桶十分有利。有关 Amazon S3 区域的列表，请参阅 AWS 一般参考 中的[区域和终端节点](#)。

## Note

属于您在特定 AWS 区域中创建的存储桶的对象绝不会离开该区域，除非您显式将它们传输到其他区域。例如，在欧洲（爱尔兰）区域存储的对象将一直留在该区域。

## 主题

- [创建存储桶 \(p. 49\)](#)
- [访问存储桶 \(p. 51\)](#)
- [存储桶配置选项 \(p. 51\)](#)
- [存储桶限制 \(p. 53\)](#)
- [创建存储桶的示例 \(p. 54\)](#)
- [删除或清空存储桶 \(p. 57\)](#)
- [S3 存储桶的 Amazon S3 默认加密 \(p. 61\)](#)
- [管理存储桶网站配置 \(p. 63\)](#)
- [Amazon S3 Transfer Acceleration \(p. 67\)](#)
- [申请方付款存储桶 \(p. 73\)](#)
- [存储桶和访问控制 \(p. 77\)](#)
- [S3 存储桶的账单和使用率报告 \(p. 77\)](#)

## 创建存储桶

Amazon S3 提供 API 以创建和管理存储桶。默认情况下，您可以在每个 AWS 账户中创建多达 100 个存储桶。如果需要更多存储桶，可以通过提交服务限额提升来调高存储桶限制。要了解如何提交存储桶限额提升，请参阅 AWS 一般参考 中的[AWS 服务限制](#)。

创建存储桶时，您需提供名称以及要在其中创建存储桶的 AWS 区域。有关如何命名存储桶的信息，请参阅[存储桶命名规则 \(p. 53\)](#)。

一个存储桶中可以存储任意数量的对象。

可以使用以下任何方法创建存储桶：

- 使用控制台。
- 使用 AWS 开发工具包以编程方式管理。

**Note**

如果需要，也可以直接从代码中调用 Amazon S3 REST API。但是，这可能会比较繁琐，因为需要您编写代码对请求进行身份验证。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [PUT 存储桶](#)。

使用 AWS 开发工具包时，您应先创建一个客户端，然后使用该客户端发送创建存储桶的请求。创建客户端时，可以指定 AWS 区域。美国东部（弗吉尼亚北部）是默认区域。请注意以下几点：

- 如果通过指定美国东部（弗吉尼亚北部）区域创建客户端，则客户端将使用以下终端节点与 Amazon S3 通信：

`s3.amazonaws.com`

可使用此客户端在任何 AWS 区域中创建存储桶。在创建存储桶请求中：

- 如果您未指定区域，则 Amazon S3 将在 美国东部（弗吉尼亚北部）区域中创建存储桶。
- 如果指定 AWS 区域，则 Amazon S3 将在指定区域中创建存储桶。
- 如果通过指定其他 AWS 区域来创建客户端，则其中每个区域都映射到特定于区域的终端节点：

`s3-<region>.amazonaws.com`

例如，如果您通过指定 eu-west-1 区域创建客户端，则它将映射到以下特定于区域的终端节点：

`s3-eu-west-1.amazonaws.com`

在这种情况下，只能使用客户端在 eu-west-1 区域中创建存储桶。如果在创建存储桶的请求中指定任何其他区域，则 Amazon S3 将返回错误。

- 如果您创建客户端来访问双堆栈终端节点，则必须指定 AWS 区域。有关更多信息，请参阅 [双堆栈终端节点 \(p. 13\)](#)。

有关可用 AWS 区域的列表，请参阅 AWS General Reference 中的 [区域和终端节点](#)。

有关示例，请参阅 [创建存储桶的示例 \(p. 54\)](#)。

## 关于权限

您可以使用 AWS 账户根凭证创建存储桶并执行其他 Amazon S3 操作。但是，AWS 建议不要使用 AWS 账户的根凭证发出请求（如创建存储桶）。而是创建一个 IAM 用户，并向该用户授予完全访问权限（用户在默认情况下没有任何权限）。我们将这些用户称为管理员用户。您可以使用管理员用户凭证而不是您账户的根凭证来与 AWS 交互和执行任务，例如创建存储桶，创建用户和授予他们权限。

有关更多信息，请参阅 AWS 一般参考 中的 [根账户凭证与 IAM 用户凭证](#) 和 IAM 用户指南中的 [IAM 最佳实践](#)。

创建资源的 AWS 账户拥有该资源。例如，如果您在您的 AWS 账户中创建一个 IAM 用户并向该用户授予创建存储桶的权限，则该用户可以创建存储桶。但是用户不拥有存储桶；用户所属的 AWS 账户拥有存储桶。用户需要资源拥有者提供其他权限来执行任何其他存储桶操作。有关为您的 Amazon S3 资源管理权限的更多信息，请参阅 [管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)。

## 访问存储桶

可以使用 Amazon S3 控制台访问存储桶。可以使用控制台 UI 执行几乎所有存储桶操作，而不必编写任何代码。

如果您以编程方式访问存储桶，请注意 Amazon S3 支持 RESTful 架构，在该架构中，存储桶和对象是资源，各自具有唯一标识资源的资源 URI。

Amazon S3 支持虚拟托管类型和路径类型 URL 访问存储桶。

- 在虚拟托管类型 URL 中，存储桶名称是 URL 中域名的一部分。例如：

- `http://bucket.s3.amazonaws.com`
- `http://bucket.s3-aws-region.amazonaws.com`

在虚拟托管类型 URL 中，您可以使用其中任一终端节点。如果向 `http://bucket.s3.amazonaws.com` 终端节点发出请求，则 DNS 具有足够信息将请求直接路由到存储桶所位于的区域。

有关更多信息，请参阅 [存储桶的虚拟托管 \(p. 42\)](#)。

- 在路径类型 URL 中，存储桶名称不包含在域中（除非使用特定于区域的终端节点）。例如：

- 美国东部（弗吉尼亚北部）区域终端节点，`http://s3.amazonaws.com/bucket`
- 特定于区域的终端节点 `http://s3-aws-region.amazonaws.com/bucket`

在路径类型 URL 中，使用的终端节点必须与存储桶所在的区域匹配。例如，如果存储桶位于南美洲（圣保罗）区域中，则必须使用 `http://s3-sa-east-1.amazonaws.com/bucket` 终端节点。如果存储桶位于美国东部（弗吉尼亚北部）区域，则必须使用 `http://s3.amazonaws.com/bucket` 终端节点。

### Important

因为可以使用路径类型和虚拟托管类型 URL 访问存储桶，所以我们建议您使用符合 DNS 标准的存储桶名称创建存储桶。有关更多信息，请参阅 [存储桶限制 \(p. 53\)](#)。

### 通过 IPv6 访问 S3 存储桶

Amazon S3 有一组双堆栈终端节点，它们支持通过 Internet 协议版本 6 (IPv6) 和 IPv4 向 S3 存储桶发出请求。有关更多信息，请参阅 [通过 IPv6 发出请求 \(p. 10\)](#)。

## 存储桶配置选项

Amazon S3 支持各种供您用于配置存储桶的选项。例如，您可以配置存储桶用于网站托管、添加配置以管理存储桶中对象的生命周期以及配置存储桶以记录对存储桶的所有访问。Amazon S3 支持供您存储和管理存储桶配置信息的子资源。即，使用 Amazon S3 API 可以创建和管理这些子资源。还可以使用控制台或 AWS 开发工具包。

### Note

还有对象级配置。例如，可以通过特定于该对象的配置访问控制列表 (ACL) 来配置对象级权限。

这些称为子资源，因为它们存在于特定存储桶或对象的上下文中。下表列出使您可以管理特定于存储桶的配置的子资源。

子资源	说明
位置	创建存储桶时，请指定需要 Amazon S3 在其中创建存储桶的 AWS 区域。Amazon S3 将此信息存储在位置子资源中并提供 API 用于检索此信息。
策略 和 ACL (访问控制列表)	<p>所有资源 (如存储桶和对象) 在默认情况下为私有。Amazon S3 支持存储桶策略和访问控制列表 (ACL) 选项，供您用于授予和管理存储桶级权限。Amazon S3 将权限信息存储在策略 和 acl 子资源中。</p> <p>有关更多信息，请参阅 <a href="#">管理对 Amazon S3 资源的访问权限 (p. 242)</a>。</p>
cors (跨源资源共享)	<p>您可以配置存储桶以便允许跨源请求。</p> <p>有关更多信息，请参阅 <a href="#">启用跨源资源共享</a>。</p>
网站	<p>您可以配置存储桶以便用于静态网站托管。Amazon S3 通过创建网站 子资源来存储此配置。</p> <p>有关更多信息，请参阅 <a href="#">在 Amazon S3 上托管静态网站</a>。</p>
日志记录	<p>日志记录使您可以跟踪针对存储桶的访问请求。每个访问日志记录都提供有关单个访问请求的详细信息，如请求者、存储桶名称、请求时间、请求操作、响应状态和错误代码 (如果有)。访问日志信息可能在安全和访问审核方面十分有用。它还可以帮助您了解您的客户群并了解您的 Amazon S3 账单。</p> <p>有关更多信息，请参阅 <a href="#">Amazon S3 服务器访问日志记录 (p. 506)</a>。</p>
事件通知	<p>您可以使存储桶向您发送特定存储桶事件的通知。</p> <p>有关更多信息，请参阅 <a href="#">配置 Amazon S3 事件通知 (p. 425)</a>。</p>
版本控制	<p>版本控制可帮助恢复意外覆盖和删除。</p> <p>我们建议将版本控制作为从错误删除或覆盖恢复对象的最佳实践。</p> <p>有关更多信息，请参阅 <a href="#">使用版本控制 (p. 380)</a>。</p>
生命周期	<p>您可以为存储桶中明确定义了生命周期的对象定义生命周期规则。例如，您可以定义一个规则以在创建一年之后存档对象，或在创建 10 年之后删除对象。</p> <p>有关更多信息，请参阅 <a href="#">对象生命周期管理</a>。</p>
跨区域复制	跨区域复制是跨不同 AWS 区域中的存储桶自动、异步地复制对象。有关更多信息，请参阅 <a href="#">跨区域复制 (CRR) (p. 442)</a> 。
标记	<p>您可以向存储桶添加成本分配标签以便对 AWS 成本进行分类和跟踪。Amazon S3 提供标记 子资源以对存储桶上的标签进行存储和管理。通过使用应用于您存储桶的标签，AWS 可生成成本分配报告，其中按标签汇总了使用和成本。</p> <p>有关更多信息，请参阅 <a href="#">S3 存储桶的账单和使用率报告 (p. 77)</a>。</p>
requestPayment	<p>默认情况下，创建存储桶的 AWS 账户 (存储桶拥有者) 支付从存储桶进行的下载。存储桶拥有者可以使用此子资源指定对请求下载的人员收取下载费用。Amazon S3 提供了一个 API，用于管理此子资源。</p> <p>有关更多信息，请参阅 <a href="#">申请方付款存储桶 (p. 73)</a>。</p>
传输加速	Transfer Acceleration 可在客户端与 S3 存储桶之间实现快速、轻松、安全的远距离文件传输。Transfer Acceleration 利用 Amazon CloudFront 的全球分布式边缘站点。

子资源	说明
	有关更多信息，请参阅 <a href="#">Amazon S3 Transfer Acceleration (p. 67)</a> 。

## 存储桶限制

存储桶归创建它的 AWS 账户所有。默认情况下，您可以在每个 AWS 账户中创建多达 100 个存储桶。如果您需要更多存储桶，则可以通过提交服务限额提升来调高存储桶限制。有关如何调高存储桶限制的信息，请参阅 AWS 一般参考 中的 [AWS 服务限制](#)。

存储桶所有权不可转让；但是，如果存储桶为空，您可以删除它。删除存储桶后，该名称可供重用，但是出于各种原因，您可能无法重新使用该名称。例如，其他账户可能使用该名称创建存储桶。另请注意，该名称可能需要过一段时间之后才能重用。因此，如果您想要使用相同的存储桶名称，请不要删除该存储桶。

存储在存储桶中的对象无数量限制，且无论您是使用很多存储桶，还是仅使用少量存储桶，性能方面都不会有差别。您可以在单个存储桶中存储所有对象，也可以在多个存储桶中组织它们。

创建存储桶之后，您无法更改其区域。

如果在创建存储桶请求中显式指定了与创建客户端时指定的区域不同的 AWS 区域，您可能会收到错误。

您无法在其他存储桶中创建存储桶。

Amazon S3 的高可用性设计主要关注获取、放置、列出和删除操作。由于存储桶操作针对集中的全球资源空间工作，因此不适合在应用程序的高可用性代码路径上创建或删除存储桶。最好是在单独的初始化或设置不常运行的例程时创建或删除存储桶。

**Note**

如果您的应用程序自动创建了存储桶，请选择不会导致命名冲突的存储桶命名方案。确保存储桶名称已被使用时，您的应用程序逻辑会选择其他存储桶名称。

## 存储桶命名规则

创建 S3 存储桶后，您将无法更改存储桶名称，因此，请明智地选择名称。

**Important**

2018 年 3 月 1 日，我们更新了美国东部（弗吉尼亚北部）区域中 S3 存储桶的命名约定，从而与所有其他全球 AWS 区域使用的命名约定保持一致。Amazon S3 不再支持包含大写字母或下划线的存储桶名称。这一变化可确保每个存储桶都可以使用虚拟托管类型的寻址找到，如 <https://myawsbucket.s3.amazonaws.com>。我们强烈建议您检查现有的存储桶创建过程，确保遵守这些符合 DNS 标准的命名约定。

以下是在所有 AWS 区域中命名 S3 存储桶的规则：

- 存储桶名称在 Amazon S3 中的所有现有存储桶名称中必须唯一。
- 存储桶名称必须符合 DNS 命名约定。
- 存储桶名称的长度必须为至少 3 个字符，且不能超过 63 个字符。
- 存储桶名称不能包含大写字符或下划线。
- 存储桶名称必须以小写字母或数字开头。
- 存储桶名称必须是一系列的一个或多个标签。相邻标签通过单个句点 (.) 分隔。存储桶名称可以包含小写字母、数字和连字符。每个标签都必须以小写字母或数字开头和结尾。
- 存储桶名称不得采用 IP 地址格式（例如，192.168.5.4）。
- 当通过安全套接字 (SSL) 使用虚拟托管式存储桶时，SSL 通配符证书仅匹配不包含句点的存储桶。要解决此问题，请使用 HTTP 或编写自己的证书验证逻辑。使用虚拟托管式存储桶时，建议您不在存储桶名称中使用 (".")。

## 不符合 DNS 标准的传统存储桶名称

从 2018 年 3 月 1 日起，我们更新了美国东部（弗吉尼亚北部）区域中 S3 存储桶的命名约定，要求使用符合 DNS 标准的名称。

美国东部（弗吉尼亚北部）区域以前允许使用更为宽松的存储桶命名标准，这可能导致存储桶名称不符合 DNS 标准。例如，`MyAWSbucket` 虽然包含大写字母，但仍属于有效的存储桶名称。如果您要尝试使用虚拟托管类型请求 (`http://MyAWSbucket.s3.amazonaws.com/yourobject`) 访问此存储桶，该 URL 将解析为存储桶 `myawsbucket`，而不是存储桶 `MyAWSbucket`。Amazon S3 将返回“未找到存储桶”错误作为响应。有关使用虚拟托管类型访问您的存储桶的更多信息，请参阅存储桶的虚拟托管 (p. 42)。

美国东部（弗吉尼亚北部）区域的传统存储桶名称规则允许存储桶名称长达 255 个字符，存储桶名称可以包含大写字母、小写字母、数字、句点 (.)、连字符 (-) 和下划线 (\_) 的任意组合。

用于 Amazon S3 Transfer Acceleration 的存储桶的名称必须符合 DNS 标准，且不得包含句点 (".")。有关传输加速的更多信息，请参阅 Amazon S3 Transfer Acceleration (p. 67)。

## 创建存储桶的示例

### 主题

- 使用 Amazon S3 控制台 (p. 54)
- 使用 AWS SDK for Java (p. 55)
- 使用适用于 .NET 的 AWS 开发工具包 (p. 55)
- 使用适用于 Ruby 的 AWS 开发工具包版本 3 (p. 57)
- 使用其他 AWS 开发工具包 (p. 57)

以下代码示例使用适用于 Java、.NET 和 Ruby 的 AWS 开发工具包以编程方式创建存储桶。这些代码示例执行以下任务：

- 创建存储桶（如果存储桶尚不存在）— 示例将通过执行以下任务来创建存储桶：
  - 通过显式指定 AWS 区域来创建客户端（示例使用 `s3-eu-west-1` 区域）。因此，客户端使用 `s3-eu-west-1.amazonaws.com` 终端节点与 Amazon S3 进行通信。您可以指定任何其他 AWS 区域。有关 AWS 区域的列表，请参阅 AWS 一般参考 中的区域和终端节点。
  - 通过仅指定存储桶名称来发送创建存储桶请求。创建存储桶请求未指定其他 AWS 区域。客户端将请求发送到 Amazon S3 以便在创建客户端时所指定的区域中创建存储桶。创建存储桶之后，您无法更改其区域。

### Note

如果在创建存储桶请求中显式指定了与创建客户端时指定的区域不同的 AWS 区域，您可能会收到错误。有关更多信息，请参阅 创建存储桶 (p. 49)。

开发工具包会将 PUT 存储桶请求发送到 Amazon S3 以创建存储桶。有关更多信息，请参阅 PUT Bucket。

- 检索有关存储桶位置的信息 — Amazon S3 将存储桶位置信息存储在与存储桶关联的位置子资源中。开发工具包发送 GET Bucket location 请求（请参阅 GET Bucket 位置）以检索此信息。

## 使用 Amazon S3 控制台

要使用 Amazon S3 控制台创建存储桶，请参阅如何创建 S3 存储桶？（在 Amazon Simple Storage Service 控制台用户指南 中）。

## 使用 AWS SDK for Java

### Example

此示例演示如何使用AWS SDK for Java创建 Amazon S3 存储桶。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.GetBucketLocationRequest;

public class CreateBucket {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            if (!s3Client.doesBucketExistV2(bucketName)) {
                // Because the CreateBucketRequest object doesn't specify a region, the
                // bucket is created in the region specified in the client.
                s3Client.createBucket(new CreateBucketRequest(bucketName));

                // Verify that the bucket was created by retrieving it and checking its
                // location.
                String bucketLocation = s3Client.getBucketLocation(new
                    GetBucketLocationRequest(bucketName));
                System.out.println("Bucket location: " + bucketLocation);
            }
        } catch(AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and 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();
        }
    }
}
```

## 使用适用于 .NET 的 AWS 开发工具包

有关如何创建和测试有效示例的信息，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

### Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using Amazon.S3;  
using Amazon.S3.Model;  
using Amazon.S3.Util;  
using System;  
using System.Threading.Tasks;  
  
namespace Amazon.DocSamples.S3  
{  
    class CreateBucketTest  
    {  
        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()  
        {  
            s3Client = new AmazonS3Client(bucketRegion);  
            CreateBucketAsync().Wait();  
        }  
  
        static async Task CreateBucketAsync()  
        {  
            try  
            {  
                if (!(await AmazonS3Util.DoesS3BucketExistAsync(s3Client, bucketName)))  
                {  
                    var putBucketRequest = new PutBucketRequest  
                    {  
                        BucketName = bucketName,  
                        UseClientRegion = true  
                    };  
  
                    PutBucketResponse putBucketResponse = await  
s3Client.PutBucketAsync(putBucketRequest);  
                }  
                // Retrieve the bucket location.  
                string bucketLocation = await FindBucketLocationAsync(s3Client);  
            }  
            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);  
            }  
        }  
        static async Task<string> FindBucketLocationAsync(IAmazonS3 client)  
        {  
            string bucketLocation;  
            var request = new GetBucketLocationRequest()  
            {  
                BucketName = bucketName  
            };  
            GetBucketLocationResponse response = await  
client.GetBucketLocationAsync(request);  
            bucketLocation = response.Location.ToString();  
            return bucketLocation;  
        }  
    }  
}
```

}

## 使用适用于 Ruby 的 AWS 开发工具包版本 3

有关如何创建和测试有效示例的信息，请参阅 [使用适用于 Ruby 的 AWS 开发工具包 - 版本 3 \(p. 524\)](#)。

### Example

```
require 'aws-sdk-s3'

s3 = Aws::S3::Client.new(region: 'us-west-2')
s3.create_bucket(bucket: 'bucket-name')
```

## 使用其他 AWS 开发工具包

有关使用其他 AWS 开发工具包的信息，请参阅[示例代码和库](#)。

## 删除或清空存储桶

可以轻松删除空的存储桶。但是，在某些情况下，您可能需要删除或清空包含对象的存储桶。在此部分中，我们将介绍如何从不受版本控制的存储桶中删除对象，以及如何从已启用版本控制的存储桶中删除对象版本和删除标记。有关版本控制的更多信息，请参阅[使用版本控制 \(p. 380\)](#)。在某些情况下，您可以选择清空存储桶而不是删除存储桶。本部分将介绍可供您用来删除或清空包含对象的存储桶的各种选项。

### 主题

- [删除存储桶 \(p. 57\)](#)
- [清空存储桶 \(p. 60\)](#)

## 删除存储桶

您可以使用 AWS 开发工具包以编程方式删除存储桶及其内容。您还可以对某个存储桶使用生命周期配置来清空其内容，然后删除该存储桶。还提供了其他选项，例如使用 Amazon S3 控制台和 AWS CLI，但这些方法受到有关存储桶中的对象数和存储桶的版本控制状态的限制。

### 删除存储桶：使用 Amazon S3 控制台

Amazon S3 控制台支持删除存储桶（不论存储桶是否为空）。有关使用 Amazon S3 控制台删除存储桶的信息，请参阅[如何删除 S3 存储桶？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

### 删除存储桶：使用 AWS CLI

仅在包含对象的存储桶未启用版本控制时可使用 AWS CLI 删除该存储桶。如果您的存储桶未启用版本控制，则可将 rb(删除存储桶) AWS CLI 命令与 --force 参数结合使用来删除非空存储桶。此命令将先删除所有对象，然后再删除存储桶。

```
$ aws s3 rb s3://bucket-name --force
```

有关更多信息，请参阅 AWS Command Line Interface 用户指南中的[通过 AWS 命令行界面使用高级别 S3 命令](#)。

要删除未启用版本控制的非空存储桶，可使用以下选项：

- 使用 AWS 开发工具包以编程方式删除存储桶。
- 使用存储桶的生命周期配置删除所有对象，然后使用 Amazon S3 控制台删除空存储桶。

## 删除存储桶：使用生命周期配置

您可以配置存储桶的生命周期以使对象过期，Amazon S3 随后将删除过期对象。您可以添加生命周期配置规则，以使所有或部分带特定键名称前缀的对象过期。例如，要删除存储桶中的所有对象，您可以将生命周期规则设置为使对象在创建一天后过期。

如果您的存储桶已启用版本控制，则也配置规则来使非当前对象过期。

在 Amazon S3 删除存储桶中的所有对象之后，您可以删除存储桶或保留存储桶。

### Important

如果您只想清空而非删除存储桶，请确保删除为清空存储桶而添加的生命周期配置规则，以使在存储桶中创建的任何新对象将保留在存储桶中。

有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#) 和 [配置对象过期 \(p. 109\)](#)。

## 删除存储桶：使用 AWS 开发工具包

可以使用 AWS 开发工具包删除存储桶。以下部分提供了有关如何使用适用于 Java 和 .NET 的 AWS 开发工具包删除存储桶的示例。首先，该代码将删除存储桶中的对象，然后删除存储桶。有关其他 AWS 开发工具包的信息，请参阅[用于 Amazon Web Services 的工具](#)。

### 使用AWS SDK for Java删除存储桶

以下 Java 示例删除包含对象的存储桶。该示例将删除所有对象，然后删除存储桶。该示例适用于已启用版本控制或未启用版本控制的存储桶。

#### Note

对于未启用版本控制的存储桶，您可以直接删除所有对象，然后删除存储桶。对于启用了版本控制的存储桶，您必须先删除所有对象版本，然后再删除存储桶。

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.util.Iterator;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectSummary;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;

public class DeleteBucket {

    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
```

```
try {
    AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(clientRegion)
        .build();

    // Delete all objects from the bucket. This is sufficient
    // for unversioned buckets. For versioned buckets, when you attempt to delete
    objects, Amazon S3 inserts
    // delete markers for all objects, but doesn't delete the object versions.
    // To delete objects from versioned buckets, delete all of the object versions
    before deleting
    // the bucket (see below for an example).
    ObjectListing objectListing = s3Client.listObjects(bucketName);
    while (true) {
        Iterator<S3ObjectSummary> objIter =
objectListing.getObjectSummaries().iterator();
        while (objIter.hasNext()) {
            s3Client.deleteObject(bucketName, objIter.next().getKey());
        }

        // If the bucket contains many objects, the listObjects() call
        // might not return all of the objects in the first listing. Check to
        // see whether the listing was truncated. If so, retrieve the next page of
objects
        // and delete them.
        if (objectListing.isTruncated()) {
            objectListing = s3Client.listNextBatchOfObjects(objectListing);
        } else {
            break;
        }
    }

    // Delete all object versions (required for versioned buckets).
    VersionListing versionList = s3Client.listVersions(new
ListVersionsRequest().withBucketName(bucketName));
    while (true) {
        Iterator<S3VersionSummary> versionIter =
versionList.getVersionSummaries().iterator();
        while (versionIter.hasNext()) {
            S3VersionSummary vs = versionIter.next();
            s3Client.deleteVersion(bucketName, vs.getKey(), vs.getVersionId());
        }

        if (versionList.isTruncated()) {
            versionList = s3Client.listNextBatchOfVersions(versionList);
        } else {
            break;
        }
    }

    // After all objects and object versions are deleted, delete the bucket.
    s3Client.deleteBucket(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();
}
}
```

}

## 清空存储桶

您可以使用 AWS 开发工具包以编程方式清空存储桶的内容 (即，删除所有内容但保留存储桶)。您还可以指定存储桶的生命周期配置以使对象过期，以便 Amazon S3 能删除这些对象。还提供了其他选项，例如使用 Amazon S3 控制台和 AWS CLI，但此方法受到有关存储桶中的对象数和存储桶的版本控制状态的限制。

### 主题

- [清空存储桶：使用 Amazon S3 控制台 \(p. 60\)](#)
- [清空存储桶：使用 AWS CLI \(p. 60\)](#)
- [清空存储桶：使用生命周期配置 \(p. 60\)](#)
- [清空存储桶：使用 AWS 开发工具包 \(p. 61\)](#)

## 清空存储桶：使用 Amazon S3 控制台

有关使用 Amazon S3 控制台清空存储桶的信息，请参阅[如何清空 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

## 清空存储桶：使用 AWS CLI

仅在存储桶未启用版本控制时可使用 AWS CLI 清空存储桶。如果您的存储桶未启用版本控制，您可以将 `rm`(删除) AWS CLI 命令与 `--recursive` 参数结合使用来清空存储桶 (或删除部分带特定键名称前缀的对象)。

以下 `rm` 命令将删除带键名称前缀 `doc` 的对象，例如，`doc/doc1` 和 `doc/doc2`。

```
$ aws s3 rm s3://bucket-name/doc --recursive
```

使用以下命令删除所有对象，而无需指定前缀。

```
$ aws s3 rm s3://bucket-name --recursive
```

有关更多信息，请参阅 AWS Command Line Interface 用户指南中的[通过 AWS 命令行界面使用高级别 S3 命令](#)。

### Note

您无法删除已启用版本控制的存储桶中的对象。当您删除一个对象时 (此命令将执行的操作)，Amazon S3 将添加一个删除标记。有关版本控制的更多信息，请参阅[使用版本控制 \(p. 380\)](#)。

要清空已启用版本控制的存储桶，您可以使用以下选项：

- 使用 AWS 开发工具包以编程方式删除存储桶。
- 使用存储桶的生命周期配置请求 Amazon S3 删除对象。
- 使用 Amazon S3 控制台 (请参阅[如何清空 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南 中))。

## 清空存储桶：使用生命周期配置

您可以配置存储桶的生命周期，以使对象过期，并请求 Amazon S3 删除过期的对象。您可以添加生命周期配置规则，以使所有或部分带特定键名称前缀的对象过期。例如，要删除存储桶中的所有对象，您可以将生命周期规则设置为使对象在创建一天后过期。

如果您的存储桶已启用版本控制，则也配置规则来使非当前对象过期。

Warning

在您的对象过期后，Amazon S3 将删除过期的对象。如果您只想清空而非删除存储桶，请确保删除为清空存储桶而添加的生命周期配置规则，以使在存储桶中创建的任何新对象将保留在存储桶中。

有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#) 和 [配置对象过期 \(p. 109\)](#)。

## 清空存储桶：使用 AWS 开发工具包

您可以使用 AWS 开发工具包清空存储桶或删除部分带特定键名称前缀的对象。

有关如何使用 AWS SDK for Java 清空存储桶的示例，请参阅[使用 AWS SDK for Java 删除存储桶 \(p. 58\)](#)。该代码将删除所有对象（不论存储桶是否启用了版本控制），然后删除存储桶。要只清空存储桶，请确保删除用来删除存储桶的语句。

有关使用其他 AWS 开发工具包的更多信息，请参阅[用于 Amazon Web Services 的工具](#)。

## S3 存储桶的 Amazon S3 默认加密

Amazon S3 默认加密提供了一种方法来设置 S3 存储桶的默认加密行为。您可以对存储桶设置默认加密，以便在存储桶中存储所有对象时对这些对象进行加密。这些对象使用 Amazon S3 托管密钥 (SSE-S3) 或 AWS KMS 托管密钥 (SSE-KMS) 通过服务器端加密进行加密。

在使用服务器端加密时，Amazon S3 在将对象保存到其数据中心的磁盘上之前对其进行加密，并在下载对象时对其进行解密。有关使用服务器端加密和加密密钥管理来保护数据的更多信息，请参阅[使用加密保护数据 \(p. 345\)](#)。

默认加密适用于所有现有的和新的 S3 存储桶。如果没有默认加密，要对存储在存储桶中的所有对象进行加密，您必须包括加密信息与每个对象存储请求。您还必须设置 S3 存储桶策略以拒绝不包含加密信息的存储请求。

对 S3 存储桶使用默认加密不会产生新的费用。请求配置默认加密功能会产生标准 Amazon S3 请求费用。有关定价的信息，请参阅 [Amazon S3 定价](#)。对于 SSE-KMS 加密密钥存储，将会产生 AWS Key Management Service 费用，这些费用在 [AWS KMS 定价](#) 中列出。

主题

- [如何为 S3 存储桶设置 Amazon S3 默认加密？\(p. 61\)](#)
- [从使用存储桶策略执行加密转至默认加密 \(p. 62\)](#)
- [将默认加密用于跨区域复制 \(p. 62\)](#)
- [使用 CloudTrail 和 CloudWatch 监控默认加密 \(p. 63\)](#)
- [更多信息 \(p. 63\)](#)

## 如何为 S3 存储桶设置 Amazon S3 默认加密？

此部分介绍如何设置 Amazon S3 默认加密。您可以使用 AWS 开发工具包、Amazon S3 REST API、AWS Command Line Interface (AWS CLI) 或 Amazon S3 控制台启用默认加密。为 S3 存储桶设置默认加密的最简单方式是使用 AWS 管理控制台。

您可以使用以下任一方方法来对存储桶设置默认加密：

- 使用 Amazon S3 控制台。有关更多信息，请参阅[如何为 S3 存储桶启用默认加密？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

- 使用以下 REST API：
  - 使用 REST API [PUT Bucket 加密](#)操作可启用默认加密并设置要使用的服务器端加密类型 – SSE-S3 或 SSE-KMS。
  - 使用 REST API [DELETE Bucket 加密](#)可禁用对象的默认加密。在禁用默认加密后，Amazon S3 仅在 PUT 请求包含加密信息时加密对象。有关更多信息，请参阅 [PUT 对象](#) 和 [PUT 对象 – 复制](#)。
  - 使用 REST API [GET Bucket 加密](#)检查当前默认加密配置。
- 使用 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 [使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)。

在为存储桶启用默认加密后，将会应用以下加密行为：

- 在启用默认加密之前，存储桶中已存在的对象的加密没有变化。
- 在启用默认加密后上传对象时：
  - 如果您的 PUT 请求标头不包含加密信息，则 Amazon S3 将使用存储桶的默认加密设置来加密对象。
  - 如果您的 PUT 请求标头包含加密信息，则 Amazon S3 将使用 PUT 请求中的加密信息加密对象，然后再将对象存储在 Amazon S3 中。如果 PUT 成功，则响应为 HTTP/1.1 200 OK，并且响应标头中包含加密信息。有关更多信息，请参阅 [PUT Object](#)。
- 如果您将 SSE-KMS 选项用于默认加密配置，则您将受到 AWS KMS 的 RPS (每秒请求数) 限制。有关 AWS KMS 限制以及如何请求提高限制的更多信息，请参阅 [AWS KMS 限制](#)。

## 从使用存储桶策略执行加密转至默认加密

如果您当前使用存储桶策略拒绝不带加密标头的 PUT 请求来对 S3 存储桶实施对象加密，建议您使用以下过程以开始使用默认加密。

从使用存储桶策略拒绝不带加密标头的 **PUT** 请求变为使用默认加密

1. 如果您计划指定使用 SSE-KMS 的默认加密，请确保使用签名版本 4 对所有 PUT 和 GET 对象请求进行签名并通过 SSL 连接发送给 Amazon S3。有关使用 AWS KMS 的信息，请参阅 [使用具有 AWS KMS 托管密钥的服务器端加密 \(SSE-KMS\) 保护数据 \(p. 346\)](#)。

### Note

默认情况下，对于 Amazon S3 控制台、AWS CLI 版本 1.11.108 及更高版本以及 2016 年 5 月之后发布的所有 AWS 开发工具包，将使用通过 SSL 连接发送给 Amazon S3 的由签名版本 4 签名的请求。

2. 删除拒绝不带加密标头的 PUT 请求的存储桶策略声明。(建议您保存要替换的存储桶策略的备份副本。)
3. 要确保将加密行为设置为所需行为，可测试多个 PUT 请求以准确模拟实际工作负载。
4. 如果您将默认加密用于 SSE-KMS，则监视您的客户端是否有在您更改前未失败但现在失败的 PUT 和 GET 请求。最可能的情况是，有一些您未按照步骤 1 更新的请求。将失败的 PUT 或 GET 请求更改为使用 AWS 签名版本 4 进行签名并通过 SSL 发送。

在为 S3 存储桶启用默认加密后，对于通过任何不带加密标头的 PUT 请求存储在 Amazon S3 中的对象，将使用存储桶级别的默认加密设置进行加密。

## 将默认加密用于跨区域复制

在为跨区域复制目标存储桶启用默认加密后，将应用以下加密行为：

- 如果未对源存储桶中的对象进行加密，则将使用目标存储桶的默认加密设置对目标存储桶中的副本对象进行加密。这将导致源对象的 ETag 与副本对象的 ETag 不同。您必须更新使用 ETag 的应用程序以应对这种差异。

- 如果使用 SSE-S3 或 SSE-KMS 对源存储桶中的对象进行加密，则目标存储桶中的副本对象将使用与源对象相同的加密。不会使用目标存储桶的默认加密设置。

## 使用 CloudTrail 和 CloudWatch 监控默认加密

您可以通过 AWS CloudTrail 事件跟踪默认加密配置请求。CloudTrail 日志中使用的 API 事件名称为 PutBucketEncryption、GetBucketEncryption 和 DeleteBucketEncryption。您也可以使用 S3 存储桶级别操作作为事件类型来创建 Amazon CloudWatch Events。有关 CloudTrail 事件的更多信息，请参阅[如何使用 CloudWatch 数据事件为 S3 存储桶启用对象级别日志记录](#)？

您可以为对象级别的 Amazon S3 操作使用 CloudTrail 日志，来跟踪向 Amazon S3 发出的 PUT 和 POST 请求，以验证在传入 PUT 请求不包含加密标头时是否使用默认加密来加密对象。

在 Amazon S3 使用默认加密设置来加密对象时，日志将包含以下字段作为名称/值对：“SSEApplied”：“Default\_SSE\_S3” or “SSEApplied”：“Default\_SSE\_KMS”。

在 Amazon S3 使用 PUT 加密标头来加密对象时，日志将包含以下字段作为名称/值对：“SSEApplied”：“SSE\_S3”，“SSEApplied”：“SSE\_KMS”或“SSEApplied”：“SSE\_C”。对于分段上传，该信息包含在 InitiateMultipartUpload API 请求中。有关使用 CloudTrail 和 CloudWatch 的更多信息，请参阅[监控 Amazon S3 \(p. 481\)](#)。

## 更多信息

- [PUT Bucket 加密](#)
- [DELETE Bucket 加密](#)
- [GET Bucket 加密](#)

## 管理存储桶网站配置

### 主题

- [使用 AWS 管理控制台管理网站 \(p. 63\)](#)
- [使用 AWS SDK for Java 管理网站 \(p. 63\)](#)
- [使用适用于 .NET 的 AWS 开发工具包管理网站 \(p. 65\)](#)
- [使用适用于 PHP 的 AWS 开发工具包管理网站 \(p. 66\)](#)
- [使用 REST API 管理网站 \(p. 67\)](#)

通过为网站托管配置存储桶，可以在 Amazon S3 中托管静态网站。有关更多信息，请参阅[在 Amazon S3 上托管静态网站 \(p. 401\)](#)。有多种方法可以用来管理存储桶的网站配置。您无需编写任何代码，就可以使用 AWS 管理控制台管理配置。您可以使用 AWS 开发工具包，以编程方式创建、更新和删除网站配置。开发工具包围绕 Amazon S3 REST API 提供封装类。如果应用程序需要它，可以直接从应用程序发送 REST API 请求。

## 使用 AWS 管理控制台管理网站

有关更多信息，请参阅[为网站托管配置存储桶 \(p. 403\)](#)。

## 使用 AWS SDK for Java 管理网站

以下示例说明如何使用 AWS SDK for Java 管理存储桶的网站配置。要将网站配置添加到存储桶，请提供存储桶名称和网站配置。网站配置必须包含索引文档，并且可包含可选的错误文档。这些文档必须已存在于存储

桶中。有关更多信息，请参阅 [PUT 存储桶网站](#)。有关 Amazon S3 网站功能的更多信息，请参阅在 [Amazon S3 上托管静态网站 \(p. 401\)](#)。

### Example

以下示例使用AWS SDK for Java将网站配置添加到存储桶，检索并输出配置，然后删除配置并确认删除操作。有关如何创建和测试有效示例的说明，请参阅 [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;

public class WebsiteConfiguration {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String indexDocName = "**** Index document name ****";
        String errorDocName = "**** Error document name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();

            // Print the existing website configuration, if it exists.
            printWebsiteConfig(s3Client, bucketName);

            // Set the new website configuration.
            s3Client.setBucketWebsiteConfiguration(bucketName, new
                BucketWebsiteConfiguration(indexDocName, errorDocName));

            // Verify that the configuration was set properly by printing it.
            printWebsiteConfig(s3Client, bucketName);

            // Delete the website configuration.
            s3Client.deleteBucketWebsiteConfiguration(bucketName);

            // Verify that the website configuration was deleted by printing it.
            printWebsiteConfig(s3Client, 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();
        }
    }

    private static void printWebsiteConfig(AmazonS3 s3Client, String bucketName) {
        System.out.println("Website configuration: ");
        BucketWebsiteConfiguration bucketWebsiteConfig =
            s3Client.getBucketWebsiteConfiguration(bucketName);
        if (bucketWebsiteConfig == null) {
            System.out.println("No website config.");
        }
    }
}
```

```
        } else {
            System.out.println("Index doc: " +
bucketWebsiteConfig.getIndexDocumentSuffix());
            System.out.println("Error doc: " + bucketWebsiteConfig.getErrorDocument());
        }
    }
}
```

## 使用适用于 .NET 的 AWS 开发工具包管理网站

以下示例说明如何使用适用于 .NET 的 AWS 开发工具包管理存储桶的网站配置。要将网站配置添加到存储桶，请提供存储桶名称和网站配置。网站配置必须包含索引文档，并且可包含可选的错误文档。这些文档必须存储在存储桶中。有关更多信息，请参阅 [PUT 存储桶网站](#)。有关 Amazon S3 网站功能的更多信息，请参阅在 [Amazon S3 上托管静态网站 \(p. 401\)](#)。

### Example

以下 C# 代码示例将网站配置添加到指定的存储桶。该配置指定索引文档和错误文档名称。有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class WebsiteConfigTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object key ***"; // For
example, index.html.
        private const string errorDocument = "*** error object key ***"; // For example,
error.html.
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName, indexDocumentSuffix,
errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string bucketName,
                                                       string indexDocumentSuffix,
                                                       string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
            }
        }
    }
}
```

```
        }
    };
    PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);

    // 2. Get the website configuration.
    GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
    {
        BucketName = bucketName
    };
    GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
    Console.WriteLine("Index document: {0}",
getResponse.WebsiteConfiguration.IndexDocumentSuffix);
    Console.WriteLine("Error document: {0}",
getResponse.WebsiteConfiguration.ErrorDocument);
}
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);
}
}
}
```

## 使用适用于 PHP 的 AWS 开发工具包管理网站

本主题将说明如何使用适用于 PHP 的 AWS 开发工具包中的类来配置和管理 Amazon S3 存储桶以进行网站托管。此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并且已正确安装适用于 PHP 的 AWS 开发工具包。有关 Amazon S3 网站功能的更多信息，请参阅[在 Amazon S3 上托管静态网站 \(p. 401\)](#)。

以下 PHP 示例将网站配置添加到指定的存储桶。`create_website_config` 方法显式提供索引文档和错误文档名称。该示例还检索网站配置并输出响应。有关 Amazon S3 网站功能的更多信息，请参阅[在 Amazon S3 上托管静态网站 \(p. 401\)](#)。

有关创建和测试有效示例的说明，请参阅[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

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

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


// Add the website configuration.
$s3->putBucketWebsite([
    'Bucket'           => $bucket,
    'WebsiteConfiguration' => [
        'IndexDocument' => ['Suffix' => 'index.html'],
        'ErrorDocument' => ['Suffix' => 'error.html']
    ]
]);
```

```
        'ErrorDocument' => ['Key' => 'error.html']
    ]);
}

// Retrieve the website configuration.
$result = $s3->getBucketWebsite([
    'Bucket' => $bucket
]);
echo $result->getPath('IndexDocument/Suffix');

// Delete the website configuration.
$s3->deleteBucketWebsite([
    'Bucket' => $bucket
]);
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 Aws\S3\S3Client 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用 REST API 管理网站

您可以使用 AWS 管理控制台或 AWS 开发工具包将存储桶配置为网站。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。有关更多信息，请参阅“Amazon Simple Storage Service API Reference”中的以下章节。

- [PUT Bucket website](#)
- [GET Bucket website](#)
- [DELETE Bucket website](#)

## Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration 可在客户与 S3 存储桶之间实现快速、轻松、安全的远距离文件传输。Transfer Acceleration 利用 Amazon CloudFront 的全球分布式边缘站点。当数据到达某个边缘站点时，会被经过优化的网络路径路由至 Amazon S3。

在使用 Transfer Acceleration 时，可能会收取额外的数据传输费。有关定价的更多信息，请参阅 [Amazon S3 定价](#)。

### 主题

- [为什么要使用 Amazon S3 Transfer Acceleration ? \(p. 67\)](#)
- [Amazon S3 Transfer Acceleration 入门 \(p. 68\)](#)
- [使用 Amazon S3 Transfer Acceleration 的要求 \(p. 69\)](#)
- [Amazon S3 Transfer Acceleration 示例 \(p. 69\)](#)

## 为什么要使用 Amazon S3 Transfer Acceleration ?

您可能出于各种原因需要对存储桶使用 Transfer Acceleration，这些原因包括：

- 您位于全球各地的客户需要上传到集中式存储桶。
- 您定期跨大洲传输数 GB 至数 TB 数据。

- 您在上传到 Amazon S3 时无法充分利用 Internet 上的所有可用带宽。

有关何时使用 Transfer Acceleration 的更多信息，请参阅 [Amazon S3 常见问题](#)。

## 使用 Amazon S3 Transfer Acceleration 速度比较工具

您可以使用 [Amazon S3 Transfer Acceleration 速度比较工具](#) 来比较各个 Amazon S3 区域内加快的上传速度和未加快的上传速度。此速度比较工具使用分段上传来将文件从浏览器传输到各种使用和未使用 Transfer Acceleration 的 Amazon S3 区域。

可使用以下任一方法访问此速度比较工具：

- 将以下 URL 复制到浏览器窗口中，并分别将 `region` 和 `yourBucketName` 替换为使用的区域（例如 `us-west-2`）和要评估的存储桶的名称：

`http://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html?region=region&origBucketName=yourBucketName`

有关 Amazon S3 支持的区域的列表，请参阅 Amazon Web Services 一般参考 中的 [区域和终端节点](#)。

- 使用 Amazon S3 控制台。有关详细信息，请参阅 Amazon Simple Storage Service 控制台用户指南 中的 [启用 Transfer Acceleration](#)。

## Amazon S3 Transfer Acceleration 入门

要开始使用 Amazon S3 Transfer Acceleration，请执行以下步骤：

1. 对存储桶启用 Transfer Acceleration – 对于要使用传输加速的存储桶，存储桶名称必须符合 DNS 命名要求，且不得包含句点（“.”）。

可通过以下任一方式对存储桶启用 Transfer Acceleration：

- 使用 Amazon S3 控制台。有关更多信息，请参阅 Amazon Simple Storage Service 控制台用户指南 中的 [启用 Transfer Acceleration](#)。
  - 使用 REST API [PUT Bucket 加速](#) 操作。
  - 使用 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 [使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)。
2. 使用以下 s3-accelerate 终端节点域名的任何一种，向启用加速功能的存储桶传送数据或从存储桶中传出数据：
    - `bucketname.s3-accelerate.amazonaws.com` - 访问启用加速功能的存储桶。
    - `bucketname.s3-accelerate.dualstack.amazonaws.com` - 通过 IPv6 访问启用加速功能的存储桶。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。Transfer Acceleration 双堆栈终端节点仅可使用终端节点名称的虚拟托管类型。有关更多信息，请参阅 [开始通过 IPv6 发出请求 \(p. 10\)](#) 和 [使用 Amazon S3 双堆栈终端节点 \(p. 13\)](#)。

### Important

当前，仅 AWS Java 开发工具包支持双堆栈加速终端节点。即将推出对 AWS CLI 和其他 AWS 开发工具包的支持。

### Note

除了加速终端节点之外，您还可以继续使用常规终端节点。

在启用 Transfer Acceleration 功能后，您可以将 Amazon S3 放置对象和获取对象请求指向 s3-accelerate 终端节点域名。例如，假定您当前有一个使用 [PUT 对象](#)（该对象在 PUT 请求中使用主机

名 mybucket.s3.amazonaws.com) 的 REST API 应用程序。要加速 PUT 操作，您只需将请求中的主机名更改为 mybucket.s3-accelerate.amazonaws.com。要重新使用标准上传速度，只需将名称更改回 mybucket.s3.amazonaws.com。

启用 Transfer Acceleration 后，最多 20 分钟后即可实现性能提升。但是，一旦启用 Transfer Acceleration，加速终端节点将随即可用。

您可以在 AWS CLI、AWS 开发工具包和其他向传入数据和从 Amazon S3 传出数据的工具中使用加速终端节点。如果您使用 AWS 开发工具包，则某些受支持的语言会使用加速终端节点客户端配置标记，这样一来，您便无需显式将 Transfer Acceleration 的终端节点设置为 `bucketname.s3-accelerate.amazonaws.com`。有关如何使用加速终端节点客户端配置标记的示例，请参阅[Amazon S3 Transfer Acceleration 示例 \(p. 69\)](#)。

您可以借助事务加速终端节点执行所有 Amazon S3 操作，但以下操作除外：[获取服务 \(列出存储桶\)](#)、[放置存储桶 \(创建存储桶\)](#) 和 [删除存储桶](#)。此外，Amazon S3 Transfer Acceleration 不支持使用 `PUT Object-Copy` 进行跨区域复制。

## 使用 Amazon S3 Transfer Acceleration 的要求

以下是对 S3 存储桶使用 Transfer Acceleration 的要求：

- 仅虚拟样式请求支持 Transfer Acceleration。有关虚拟样式请求的更多信息，请参阅[使用 REST API 创建请求 \(p. 41\)](#)。
- 用于 Transfer Acceleration 的存储桶的名称必须符合 DNS 标准，且不得包含句点 (“.”)。
- 必须对存储桶启用 Transfer Acceleration。在对存储桶启用 Transfer Acceleration 后，可能需要最多 30 分钟的时间才能加快向存储桶传输数据的速度。
- 要访问已启用 Transfer Acceleration 的存储桶，您必须使用终端节点 `bucketname.s3-accelerate.amazonaws.com` 或双堆栈终端节点 `bucketname.s3-accelerate.dualstack.amazonaws.com`，通过 IPv6 连接至启用的存储桶。
- 您必须是存储桶拥有者才能设置传输加速状态。存储桶拥有者可以向其他用户分配权限，使他们能够对存储桶设置加速状态。`s3:PutAccelerateConfiguration` 权限允许用户对存储桶启用或禁用 Transfer Acceleration。`s3:GetAccelerateConfiguration` 权限允许用户返回存储桶的 Transfer Acceleration 状态，即 `Enabled` 或 `Suspended`。有关这些权限的更多信息，请参阅[与存储桶子资源操作相关的权限 \(p. 284\)](#) 和 [管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)。

## 更多信息

- [GET Bucket 加速](#)
- [PUT Bucket 加速](#)

## Amazon S3 Transfer Acceleration 示例

本节提供了有关如何对存储桶启用 Amazon S3 Transfer Acceleration 和对启用的存储桶使用加速终端节点的示例。一些 AWS 开发工具包支持的语言（例如，Java 和 .NET）使用加速终端节点客户端配置标记，这样一来，您便无需显式将 Transfer Acceleration 的终端节点设置为 `bucketname.s3-accelerate.amazonaws.com`。有关 Transfer Acceleration 的更多信息，请参阅[Amazon S3 Transfer Acceleration \(p. 67\)](#)。

### 主题

- [使用 Amazon S3 控制台 \(p. 70\)](#)
- [从 AWS Command Line Interface \(AWS CLI\) 使用 Transfer Acceleration \(p. 70\)](#)
- [将 Transfer Acceleration 与 AWS SDK for Java 结合使用 \(p. 71\)](#)

- 从适用于 .NET 的 AWS 开发工具包使用 Transfer Acceleration (p. 72)
- 通过适用于 JavaScript 的 AWS 开发工具包使用 Transfer Acceleration (p. 73)
- 通过 AWS SDK for Python (Boto) 使用 Transfer Acceleration (p. 73)
- 使用其他 AWS 开发工具包 (p. 73)

## 使用 Amazon S3 控制台

有关使用 Amazon S3 控制台对存储桶启用 Transfer Acceleration 的信息，请参阅 Amazon Simple Storage Service 控制台用户指南 中的[启用 Transfer Acceleration](#)。

## 从 AWS Command Line Interface (AWS CLI) 使用 Transfer Acceleration

本节提供了用于 Transfer Acceleration 的 AWS CLI 命令的示例。有关设置 AWS CLI 的说明，请参阅[设置 AWS CLI \(p. 520\)](#)。

### 使用 AWS CLI 对存储桶启用 Transfer Acceleration

使用 AWS CLI `put-bucket-accelerate-configuration` 命令对存储桶启用或暂停 Transfer Acceleration。以下示例设置 `Status=Enabled` 以对存储桶启用 Transfer Acceleration。可使用 `Status=Suspended` 暂停 Transfer Acceleration。

Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket bucketname --accelerate-configuration Status=Enabled
```

### 从 AWS CLI 使用 Transfer Acceleration

在 AWS Config 文件的配置文件中将配置值 `use_accelerate_endpoint` 设置为 `true` 会将通过 s3 和 s3api AWS CLI 命令发起的所有 Amazon S3 请求定向到加速终端节点：`s3-accelerate.amazonaws.com`。必须对存储桶启用 Transfer Acceleration 才能使用加速终端节点。

使用存储桶寻址的虚拟样式发送所有请求：`my-bucket.s3-accelerate.amazonaws.com`。不会将任何 `ListBuckets`、`CreateBucket` 和 `DeleteBucket` 请求发送到加速终端节点，因为此终端节点不支持这些操作。有关 `use_accelerate_endpoint` 的更多信息，请参阅 [AWS CLI S3 配置](#)。

以下示例在默认配置文件中将 `use_accelerate_endpoint` 设置为 `true`。

Example

```
$ aws configure set default.s3.use_accelerate_endpoint true
```

如果您需要对某些 AWS CLI 命令使用加速终端节点，但不对其他此类命令使用加速终端节点，则可使用以下两种方法中的任一方法：

- 您可以通过将任何 s3 或 s3api 命令的 `--endpoint-url` 参数设置为 `https://s3-accelerate.amazonaws.com` 或 `http://s3-accelerate.amazonaws.com` 来对每条命令使用加速终端节点。
- 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_accelerate_endpoint` 设置为 `true` 的配置文件和一个不设置 `use_accelerate_endpoint` 的配置文件。在执行一条命令时，根据是否需要使用加速终端节点来指定要使用的配置文件。

## 将对象上传到已启用 Transfer Acceleration 的存储桶的 AWS CLI 示例

以下示例通过使用已配置为使用加速终端节点的默认配置文件来将文件上传到已启用 Transfer Acceleration 的存储桶。

### Example

```
$ aws s3 cp file.txt s3://bucketname/keyname --region region
```

以下示例通过使用 `--endpoint-url` 参数指定加速终端节点来将文件上传到已启用 Transfer Acceleration 的存储桶。

### Example

```
$ aws configure set s3.addressing_style virtual
$ aws s3 cp file.txt s3://bucketname/keyname --region region --endpoint-url http://s3-accelerate.amazonaws.com
```

## 将Transfer Acceleration与AWS SDK for Java结合使用

### Example

以下示例演示如何使用加速终端节点将对象上传到 Amazon S3。本示例执行以下操作：

- 创建配置为使用加速终端节点的 `AmazonS3Client`。客户端访问的所有存储桶都必须已启用传输加速。
- 对指定存储桶启用传输加速。仅当您指定的存储桶尚未启用传输加速时，此步骤才是必需的。
- 验证是否为指定存储桶启用了传输加速。
- 使用存储桶的加速终端节点将新对象上传到指定存储桶。

有关使用 Transfer Acceleration 的更多信息，请参阅 [Amazon S3 Transfer Acceleration 入门 \(p. 68\)](#)。有关创建和测试有效示例的说明，请参阅 [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketAccelerateConfiguration;
import com.amazonaws.services.s3.model.BucketAccelerateStatus;
import com.amazonaws.services.s3.model.GetBucketAccelerateConfigurationRequest;
import com.amazonaws.services.s3.model.SetBucketAccelerateConfigurationRequest;

public class TransferAcceleration {
    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";

        try {
            // Create an Amazon S3 client that is configured to use the accelerate endpoint.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .enableAccelerateMode()
                .build();

            // Enable Transfer Acceleration for the specified bucket.
            s3Client.setBucketAccelerateConfiguration(

```

```
        new SetBucketAccelerateConfigurationRequest(bucketName,
                                                 new
                                                 BucketAccelerateConfiguration(
                                                 BucketAccelerateStatus.Enabled)));

        // Verify that transfer acceleration is enabled for the bucket.
        String accelerateStatus = s3Client.getBucketAccelerateConfiguration(
            new
            GetBucketAccelerateConfigurationRequest(bucketName))
                .getStatus();
        System.out.println("Bucket accelerate status: " + accelerateStatus);

        // Upload a new object using the accelerate endpoint.
        s3Client.putObject(bucketName, keyName, "Test object for transfer
acceleration");
        System.out.println("Object \\" + keyName + "\\" uploaded with transfer
acceleration.");
    }
    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();
    }
}
}
```

## 从适用于 .NET 的 AWS 开发工具包使用 Transfer Acceleration

以下示例演示如何使用适用于 .NET 的 AWS 开发工具包对存储桶启用 Transfer Acceleration。有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

### Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class TransferAccelerationTest
    {
        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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            EnableAccelerationAsync().Wait();
        }

        static async Task EnableAccelerationAsync()
        {
```

```
try
{
    var putRequest = new PutBucketAccelerateConfigurationRequest
    {
        BucketName = bucketName,
        AccelerateConfiguration = new AccelerateConfiguration
        {
            Status = BucketAccelerateStatus.Enabled
        }
    };
    await s3Client.PutBucketAccelerateConfigurationAsync(putRequest);

    var getRequest = new GetBucketAccelerateConfigurationRequest
    {
        BucketName = bucketName
    };
    var response = await
s3Client.GetBucketAccelerateConfigurationAsync(getRequest);

    Console.WriteLine("Acceleration state = '{0}' ", response.Status);
}
catch (AmazonS3Exception amazonS3Exception)
{
    Console.WriteLine(
        "Error occurred. Message:{0} when setting transfer acceleration",
        amazonS3Exception.Message);
}
}
```

在将对象上传到启用了 Transfer Acceleration 的存储桶时，可指定在创建客户端时使用加速终端节点，如下所示：

```
var client = new AmazonS3Client(new AmazonS3Config
{
    RegionEndpoint = TestRegionEndpoint,
    UseAccelerateEndpoint = true
})
```

## 通过适用于 JavaScript 的 AWS 开发工具包使用 Transfer Acceleration

有关通过使用适用于 JavaScript 的 AWS 开发工具包来启用 Transfer Acceleration 的示例，请参阅适用于 JavaScript 的 AWS 开发工具包 API 参考中的[调用 putBucketAccelerateConfiguration 操作](#)。

## 通过 AWS SDK for Python (Boto) 使用 Transfer Acceleration

有关通过使用 SDK for Python 来启用 Transfer Acceleration 的示例，请参阅 AWS SDK for Python (Boto 3) API Reference 中的[put\\_bucket\\_accelerate\\_configuration](#)。

## 使用其他 AWS 开发工具包

有关使用其他 AWS 开发工具包的信息，请参阅[示例代码和库](#)。

# 申请方付款存储桶

主题

- 使用 Amazon S3 控制台配置申请方付款 (p. 74)
- 使用 REST API 配置申请方付款 (p. 75)
- 费用详细信息 (p. 77)

通常，存储桶拥有者将支付与他们的存储桶相关联的所有 Amazon S3 存储和数据转移费用。但是，存储桶拥有者可以将存储桶配置为申请方付款存储桶。使用申请方付款存储桶时，申请方（而不是存储桶拥有者）将支付请求和从存储桶下载数据的费用。存储桶拥有者将始终支付存储数据的费用。

通常情况下，当您想共享数据，而又不会产生与访问数据等其他操作相关联的费用时，您可以将存储桶配置为申请方付款。例如，当使大型数据集（如邮政编码目录、参考数据、地理空间信息或网络爬取数据）可用时，您可能会使用申请方付款存储桶。

#### Important

如果您在存储桶上启用了申请方付款，则不允许匿名访问该存储桶。

您必须对涉及申请方付款存储桶的所有请求进行身份验证。请求身份验证使 Amazon S3 能够识别申请方对申请方付款存储桶的使用并对其收费。

当请求者在做出其请求前担任 AWS Identity and Access Management (IAM) 角色时，该角色所属的账户将负责处理此请求。有关 IAM 角色的更多信息，请参阅 IAM 用户指南 中的 [IAM 角色](#)。

将存储桶配置为申请方付款存储桶后，申请方必须在其请求中包含 `x-amz-request-payer`（在 POST、GET 和 HEAD 请求的标头中，或在 REST 请求中作为参数），以显示他们知道请求和数据下载将产生费用。

申请方付款存储桶不支持以下内容。

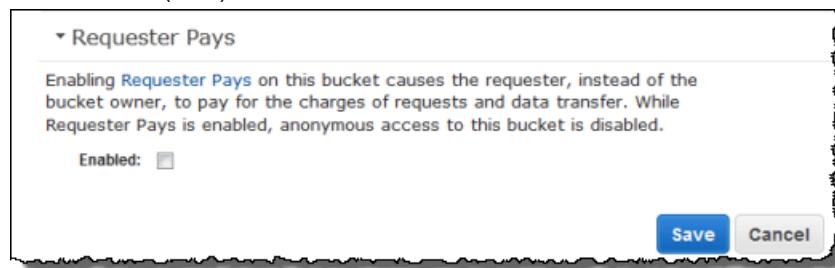
- 匿名申请
- BitTorrent
- SOAP 请求
- 您不能将申请方付款存储桶用作进行最终用户日志记录的目标存储桶，反之亦然；但是，您可以在目标存储桶不是申请方付款存储桶的申请方付款存储桶上开启最终用户日志记录。

## 使用 Amazon S3 控制台配置申请方付款

您可以使用 Amazon S3 控制台为申请方付款配置存储桶。

### 为申请方付款配置存储桶的步骤

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 在“Buckets”列表中，单击存储桶名左侧的详细信息图标，再单击“Properties”即可显示存储桶属性。
3. 在“Properties”窗格中，单击“Requester Pays”。
4. 选中 Enabled (启用) 复选框。



## 使用 REST API 配置申请方付款

### 主题

- [设置 requestPayment 存储桶配置 \(p. 75\)](#)
- [检索 requestPayment 配置 \(p. 75\)](#)
- [在申请方付款存储桶中下载数据元 \(p. 76\)](#)

## 设置 requestPayment 存储桶配置

只有存储桶拥有者才能将存储桶的 RequestPaymentConfiguration.payer 配置值设置为 BucketOwner (默认值) 或 Requester。设置 requestPayment 资源是可选的。默认情况下，存储桶不是申请方付款存储桶。

要将申请方付款存储桶恢复为常规存储桶，请使用值 BucketOwner。通常情况下，在将数据上传到 Amazon S3 存储桶时，您将使用 BucketOwner，然后将值设置为 Requester，才能在该存储桶中发布对象。

### 设置 requestPayment 的步骤

- 使用 PUT 请求在指定存储桶上将 Payer 值设置为 Requester。

```
PUT ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Content-Length: 173
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

如果请求成功，Amazon S3 将返回类似于以下内容的响应。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
x-amz-request-charged:requester
```

您只能在存储桶级别上设置申请方付款；您不能为存储桶内的特定对象设置申请方付款。

您可以随时将存储桶配置为 BucketOwner 或 Requester。但是，需要注意的是可能会出现一些短暂延迟（大约数分钟），新配置值才会生效。

### Note

在将存储桶配置为申请方付款之前，分发预签名 URL 的存储桶拥有者应当再三考虑，尤其是在 URL 的生命周期非常长时更应如此。在每次申请方使用预签名 URL (使用存储桶拥有者的凭证) 时，会向存储桶拥有者收取费用。

## 检索 requestPayment 配置

您可以通过请求资源 Payer 来确定在存储桶上设置的 requestPayment 值。

### 返回 requestPayment 资源的步骤

- 使用 GET 请求来获取 requestPayment 资源，如以下请求所示。

```
GET ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

如果请求成功，Amazon S3 将返回类似于以下内容的响应。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: [length]
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

此响应显示 payer 值已设置为 Requester。

## 在申请方付款存储桶中下载数据元

因为会向申请方收取从申请方付款存储桶下载数据的费用，因此请求必须包含特殊参数 x-amz-request-payer，该参数确认申请方了解将向他们收取下载费用。要在申请方付款存储桶中访问对象，请求必须包含以下内容之一。

- 对于 GET、HEAD 和 POST 请求，标头中应包含 x-amz-request-payer : requester
- 对于已签名的 URL，需在请求中包括 x-amz-request-payer=requester

如果请求成功且已向申请方收取费用，则响应将包括标头 x-amz-request-charged:requester。如果请求中没有 x-amz-request-payer，Amazon S3 将返回 403 错误并向存储桶拥有者收取请求费用。

#### Note

存储桶拥有者无需将 x-amz-request-payer 添加到他们的请求。

确保在您的签名计算中包含 x-amz-request-payer 及其值。有关详细信息，请参阅[构建 canonicalizedAmzHeaders 元素 \(p. 535\)](#)。

### 从申请方付款存储桶下载对象的步骤

- 使用 GET 请求从申请方付款存储桶下载对象，如以下请求所示。

```
GET / [destinationObject] HTTP/1.1
Host: [BucketName].s3.amazonaws.com
x-amz-request-payer : requester
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

如果 GET 请求成功且已向申请方收取费用，则响应将包括 x-amz-request-charged:requester。

Amazon S3 可以为尝试从申请方付款存储桶获取对象的请求返回 Access Denied 错误。有关更多信息，请参阅[错误响应](#)。

## 费用详细信息

成功的申请方付款请求费用简单明了：申请方支付数据传输和请求方面的费用；存储桶拥有者支付数据存储方面的费用。但是，在以下条件下会对存储桶拥有者收取请求费用：

- 申请方未在标头中 (GET、HEAD 或 POST) 包含参数 `x-amz-request-payer`，或未在请求中将其作为参数 (REST) (HTTP 代码 403)。
- 请求身份验证失败 (HTTP 代码 403)。
- 请求是匿名的 (HTTP 代码 403)。
- 请求是 SOAP 请求。

## 存储桶和访问控制

每个存储桶都具有关联的访问控制策略。此策略将管理对象在存储桶中的创建、删除和枚举。有关更多信息，请参阅[管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)。

## S3 存储桶的账单和使用率报告

当使用 Amazon Simple Storage Service (Amazon S3) 时，您不必支付任何前期费用或承诺您会存储多少内容。与其他 Amazon Web Services (AWS) 服务一样，您可以即付即用，并且只为使用的内容付费。

AWS 提供 Amazon S3 的以下报告：

- 账单报告 – 多个报告，提供您正在使用的 AWS 服务 (包括 Amazon S3) 的所有活动的高级视图。AWS 始终把 Amazon S3 费用账单开具给 S3 存储桶的所有者，除非存储桶创建为“请求者付费”存储桶。有关请求者付费的更多信息，请参阅[申请方付款存储桶 \(p. 73\)](#)。有关账单报告的更多信息，请参阅[Amazon S3 的 AWS 账单报告 \(p. 77\)](#)。
- 使用率报告 – 概括了特定服务按小时、天或月聚合的活动情况。可以选择要包括的使用类型和操作。您还可以选择数据聚合的方式。有关更多信息，请参阅[Amazon S3 的 AWS 使用率报告 \(p. 79\)](#)。

以下主题提供有关 Amazon S3 账单和使用率报告的信息。

### 主题

- [Amazon S3 的 AWS 账单报告 \(p. 77\)](#)
- [Amazon S3 的 AWS 使用率报告 \(p. 79\)](#)
- [了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)
- [使用成本分配 S3 存储桶标签 \(p. 84\)](#)

## Amazon S3 的 AWS 账单报告

AWS 为您提供按 AWS 服务和功能分隔的使用率信息和成本。有多种 AWS 账单报告可用：月度报告、成本分配报告和详细账单报告。有关如何查看您的账单报告的信息，请参阅 AWS Billing and Cost Management 用户指南 中的[查看您的账单](#)。

还可以下载使用率报告，它会比账单报告给出更多有关 Amazon S3 存储使用量的详细信息。有关更多信息，请参阅[Amazon S3 的 AWS 使用率报告 \(p. 79\)](#)。

下表列出了与 Amazon S3 使用量相关联的费用。

#### Amazon S3 使用费

费用	注释
存储	您要在您的 S3 存储桶中存储对象付费。您需支付的费率取决于对象的大小、在该月内将对象存储了多长时间，以及存储类 - STANDARD、STANDARD_IA (适用于不常访问的 IA)、ONEZONE_IA、GLACIER 或低冗余存储 (RRS)。有关存储类别的更多信息，请参阅 <a href="#">存储类别 (p. 90)</a> 。
请求	您要为请求付费，例如，针对您的 S3 存储桶和对象进行的 GET 请求。这包括生命周期请求。请求的费率取决于您提出的请求类型。有关请求定价的信息，请参阅 <a href="#">Amazon S3 定价</a> 。
检索	您要为检索在 STANDARD_IA、ONEZONE_IA 和 Glacier 存储中保存的对象付费。
提前删除	如果在最小存储承诺过去之前删除在 STANDARD_IA、ONEZONE_IA 或 Glacier 存储中保存的对象，您需要为该对象支付提前删除费用。
存储管理	您要在您账户的存储桶上启用的存储管理功能 (Amazon S3 清单、分析和对象标签) 付费。
带宽	您要为所有出入 Amazon S3 的带宽付费，以下内容除外： <ul style="list-style-type: none"><li>• 从 Internet 传入的数据</li><li>• 当 Amazon Elastic Compute Cloud (Amazon EC2) 实例位于 S3 存储桶所在的 AWS 区域时传出到实例的数据</li><li>• 传出到 Amazon CloudFront (CloudFront) 的数据</li></ul> 您还要为使用 Amazon S3 传输加速所传输的全部数据付费。

有关存储、数据传输和服务的 Amazon S3 使用费的详细信息，请参阅 [Amazon S3 定价](#) 和 [Amazon S3 常见问题](#)。

有关了解在 Amazon S3 账单和使用率报告中使用的代码和缩写的信息，请参阅 [了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)。

## 更多信息

- [Amazon S3 的 AWS 使用率报告 \(p. 79\)](#)
- [使用成本分配 S3 存储桶标签 \(p. 84\)](#)
- [AWS 账单和成本管理](#)
- [Amazon S3 定价](#)
- [Amazon S3 常见问题](#)
- [Amazon Glacier 定价](#)

## Amazon S3 的 AWS 使用率报告

有关您的 Amazon S3 存储使用量的更多详细信息，请下载动态生成的 AWS 使用率报告。可以选择要包括的使用类型、操作和时间段。您还可以选择数据聚合的方式。

在下载使用率报告时，您可以选择按小时、天或月聚合使用量数据。Amazon S3 使用率报告按使用类型和 AWS 地区列出操作，例如，从亚太区域（悉尼）传出的数据量。

Amazon S3 使用率报告包含以下信息：

- Service – Amazon Simple Storage Service
- Operation – 对您的存储桶或对象执行的操作。有关 Amazon S3 操作的详细说明，请参阅[跟踪使用率报告中的操作 \(p. 84\)](#)。
- UsageType – 下列值之一：
  - 标识存储类型的代码
  - 标识请求类型的代码
  - 标识检索类型的代码
  - 标识数据传输类型的代码
  - 标识从 STANDARD\_IA、ONEZONE\_IA 或 GLACIER 存储中进行的提前删除的代码
- StorageObjectCount – 在给定存储桶中存储的对象计数

有关 Amazon S3 使用类型的详细说明，请参阅[了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)。

- Resource – 与列出的使用量相关联的存储桶的名称。
- StartTime – 以协调世界时 (UTC) 表示的使用开始时间。
- EndTime – 以协调世界时 (UTC) 表示的使用结束时间。
- UsageValue – 下列容量值之一：
  - 指定时间段内的请求数
  - 传输的数据量（以字节为单位）
  - 存储的数据量（以字节为单位），即在给定小时内存储的字节数
  - 与从 GLACIER、STANDARD\_IA 或 ONEZONE\_IA 存储中进行的还原相关联的数据量（以字节为单位）

### Tip

有关 Amazon S3 收到的针对您对象的每个请求的详细信息，请打开您的存储桶的服务器访问日志记录。有关更多信息，请参阅[Amazon S3 服务端访问日志记录 \(p. 506\)](#)。

您可以下载 XML 或逗号分隔值 (CSV) 文件格式的使用率报告。下面是在电子表格应用程序中打开的示例 CSV 使用率报告。

Service	Operation	UsageType	Resource	StartTime	EndTime	UsageValue
AmazonS3	HeadBucket	USW2-C3DataTransfer-Out-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	15309
AmazonS3	PutObject	USW2-C3DataTransfer-In-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	19062
AmazonS3	HeadBucket	USW2-Requests-Tier2	admin-created3	6/1/2017 0:00	7/1/2017 0:00	68
AmazonS3	PutObjectForRepl	USW1-Requests-SIA-Tier1	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	178294
AmazonS3	PutObjectForRepl	USW1-USW2-AWS-In-Bytes	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	387929083
AmazonS3	GetObjectForRepl	USW2-Requests-NoCharge	admin-created3	6/1/2017 0:00	7/1/2017 0:00	108
AmazonS3	GetObjectForRepl	USW2-USW1-AWS-Out-Bytes	my-test-bucket-bash	6/1/2017 0:00	7/1/2017 0:00	387910021

有关了解使用率报告的信息，请参阅[了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)。

## 下载 AWS 使用率报告

您可以下载 .xml 或 .csv 文件格式的使用率报告。

## 下载使用率报告

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 在标题栏中，选择您的 AWS Identity and Access Management (IAM) 用户名，然后选择 My Billing Dashboard。
3. 在导航窗格中，选择 Reports。
4. 在 Other Reports 部分中，选择 AWS Usage Report。
5. 对于 Services：，选择 Amazon Simple Storage Service。
6. 对于 Download Usage Report，选择以下设置：
  - Usage Types – 有关 Amazon S3 使用类型的详细说明，请参阅[了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)。
  - Operation – 有关 Amazon S3 操作的详细说明，请参阅[跟踪使用率报告中的操作 \(p. 84\)](#)。
  - Time Period – 您希望报告涵盖的时间段。
  - Report Granularity – 您是否要让报告包含按小时、天或月计算的小计。
7. 要选择报告的格式，请选择该格式对应的 Download，然后按照提示进行操作以查看或保存报告。

## 更多信息

- [了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)
- [Amazon S3 的 AWS 账单报告 \(p. 77\)](#)

## 了解 Amazon S3 的 AWS 账单和使用率报告

Amazon S3 账单和使用率报告使用代码和缩写。例如，对于在下表中定义的使用类型，*region* 将被替换为以下缩写之一：

- APN1: 亚太区域 ( 东京 )
- APN2: 亚太区域 ( 首尔 )
- APS1: 亚太区域 ( 新加坡 )
- APS2: 亚太区域 ( 悉尼 )
- APS3: 亚太地区 ( 孟买 )
- CAN1: 加拿大 ( 中部 )
- EUC1: 欧洲 ( 法兰克福 )
- EU: 欧洲 ( 爱尔兰 )
- EUW2: 欧洲 ( 伦敦 )
- SAE1: 南美洲 ( 圣保罗 )
- UGW1: AWS GovCloud (US-West)
- USE1 (or no prefix): 美国东部 ( 弗吉尼亚北部 )
- USE2: 美国东部 ( 俄亥俄州 )
- USW1: 美国西部 ( 加利福尼亚北部 )
- USW2: 美国西部 ( 俄勒冈 )

有关各 AWS 区域的定价信息，请参阅 [Amazon S3 定价](#)。

下表中的第一列列出了您的账单和使用率报告中显示的使用类型。

### 使用类型

Usage Type	单位	粒度	描述
<i>region1-region2-AWS-In-Bytes</i>	字节	每小时	从 AWS 区域 2 传输到 AWS 区域 1 的数据量
<i>region1-region2-AWS-Out-Bytes</i>	字节	每小时	从 AWS 区域 1 传输到 AWS 区域 2 的数据量
<i>region-C3DataTransfer-Out-Bytes</i>	字节	每小时	在同一 AWS 区域内从 Amazon S3 传输到 Amazon EC2 的数据量
<i>region-C3DataTransfer-In-Bytes</i>	字节	每小时	在同一 AWS 区域内从 Amazon EC2 传输到 Amazon S3 的数据量
<i>region-S3G-DataTransfer-Out-Bytes</i>	字节	每小时	为了将对象转换为 GLACIER 存储而从 Amazon S3 传出的数据量
<i>region-S3G-DataTransfer-In-Bytes</i>	字节	每小时	为了从 GLACIER 存储中还原对象而传入 Amazon S3 的数据量
<i>region-DataTransfer-Regional-Bytes</i>	字节	每小时	在同一 AWS 区域内从 Amazon S3 传输到 AWS 资源的数据量
StorageObjectCount	Count	每天	在给定存储桶中存储的对象数
<i>region-CloudFront-In-Bytes</i>	字节	每小时	从 CloudFront 分配传输到 AWS 区域的数据量
<i>region-CloudFront-Out-Bytes</i>	字节	每小时	从 AWS 区域传输到 CloudFront 分配的数据量
<i>region-EarlyDelete-ByteHrs</i>	字节小时数 <sup>1</sup>	每小时	在 90 天最低承诺结束之前，从 GLACIER 存储中删除的对象的按比例存储使用量 <sup>2</sup>
<i>region-EarlyDelete-SIA</i>	字节小时数	每小时	在 30 天最低承诺结束之前，从 STANDARD_IA 中删除的对象的按比例存储使用量 <sup>3</sup>
<i>region-EarlyDelete-ZIA</i>	字节小时数	每小时	在 30 天最低承诺结束之前，从 ONEZONE_IA 中删除的对象的按比例存储使用量 <sup>3</sup>
<i>region-EarlyDelete-SIA-SmObjects</i>	字节小时数	每小时	在 30 天最低承诺结束之前，从 STANDARD_IA 中删除的小对象(小于 128KB) 的按比例存储使用量 <sup>4</sup>
<i>region-EarlyDelete-ZIA-SmObjects</i>	字节小时数	每小时	在 30 天最低承诺结束之前，从 ONEZONE_IA 中删除的小对象(小于 128KB) 的按比例存储使用量 <sup>4</sup>

Usage Type	单位	粒度	描述
<i>region</i> -Inventory-ObjectsListed	对象	每小时	为采用清单列表的对象组 (对象按存储桶或前缀分组) 列出的对象数
<i>region</i> -Requests-SIA-Tier1	Count	每小时	STANDARD_IA 对象上的 PUT、COPY、POST 或 LIST 请求的数量
<i>region</i> -Requests-ZIA-Tier1	Count	每小时	ONEZONE_IA 对象上的 PUT、COPY、POST 或 LIST 请求的数量
<i>region</i> -Requests-SIA-Tier2	Count	每小时	STANDARD_IA 对象上的 GET 和所有其他非 SIA-Tier1 请求的数量
<i>region</i> -Requests-ZIA-Tier2	Count	每小时	ONEZONE_IA 对象上的 GET 和所有其他非 ZIA-Tier1 请求的数量
<i>region</i> -Requests-Tier1	Count	每小时	STANDARD、RRS 和标签的 PUT、COPY、POST 或 LIST 请求的数量
<i>region</i> -Requests-Tier2	Count	每小时	GET 和其他非 Tier1 请求的数量
<i>region</i> -Requests-Tier3	Count	每小时	GLACIER 存档请求和标准还原请求的数量
<i>region</i> -Requests-Tier4	Count	每小时	转换到 STANDARD_IA 或 ONEZONE_IA 存储的生命周期转换数
<i>region</i> -Requests-Tier5	Count	每小时	批量 GLACIER 还原请求的数量
<i>region</i> -Requests-Tier6	Count	每小时	加急 GLACIER 还原请求的数量
<i>region</i> -Bulk-Retrieval-Bytes	字节	每小时	使用批量 GLACIER 请求检索的数据的字节数
<i>region</i> -Expedited-Retrieval-Bytes	字节	每小时	使用加急 GLACIER 请求检索的数据的字节数
<i>region</i> -Standard-Retrieval-Bytes	字节	每小时	使用标准 GLACIER 请求检索的数据的字节数
<i>region</i> -Retrieval-SIA	字节	每小时	从 STANDARD_IA 存储检索的数据的字节数
<i>region</i> -Retrieval-ZIA	字节	每小时	从 ONEZONE_IA 存储检索的数据的字节数
<i>region</i> -StorageAnalytics-ObjCount	对象	每小时	存储分析跟踪的每个对象组 (其中对象按存储桶或前缀分组) 中的唯一对象数

Usage Type	单位	粒度	描述
<i>region</i> -Select-Scanned-Bytes	字节	每小时	使用 Select 请求从 STANDARD 存储扫描的数据的字节数
<i>region</i> -Select-Scanned-SIA-Bytes	字节	每小时	使用 Select 请求从 STANDARD_IA 存储扫描的数据的字节数
<i>region</i> -Select-Scanned-ZIA-Bytes	字节	每小时	使用 Select 请求从 ONEZONE_IA 存储扫描的数据的字节数
<i>region</i> -Select-Returned-Bytes	字节	每小时	使用 Select 请求从 STANDARD 存储返回的数据的字节数
<i>region</i> -Select-Returned-SIA-Bytes	字节	每小时	使用 Select 请求从 STANDARD_IA 存储返回的数据的字节数
<i>region</i> -Select-Returned-ZIA-Bytes	字节	每小时	使用 Select 请求从 ONEZONE_IA 存储返回的数据的字节数
<i>region</i> -TagStorage-TagHrs	标签小时数	每天	按小时报告的存储桶中所有对象上的标签总数
<i>region</i> -TimedStorage-ByteHrs	字节小时数	每天	在 STANDARD 存储中保存的字节小时数
<i>region</i> -TimedStorage-GLACIERByteHrs	字节小时数	每天	在 GLACIER 存储中保存的字节小时数
<i>region</i> -TimedStorage-RRS-ByteHrs	字节小时数	每天	在低冗余存储 (RRS) 存储中保存的字节小时数
<i>region</i> -TimedStorage-SIA-ByteHrs	字节小时数	每天	在 STANDARD_IA 存储中保存的字节小时数
<i>region</i> -TimedStorage-ZIA-ByteHrs	字节小时数	每天	在 ONEZONE_IA 存储中保存的字节小时数
<i>region</i> -TimedStorage-SIA-SmObjects	字节小时数	每天	在 STANDARD_IA 存储中保存的小对象 (小于 128KB) 的字节小时数
<i>region</i> -TimedStorage-ZIA-SmObjects	字节小时数	每天	在 ONEZONE_IA 存储中保存的小对象 (小于 128KB) 的字节小时数

备注:

- 有关字节小时数单位的更多信息，请参阅[将使用率字节小时数转换为计费 GB 月数 \(p. 84\)](#)。
- 对于存档到 GLACIER 存储类的对象，当它们在 90 天之前被删除时，对剩余的天数会以 GB 为单位按比例收费。

3. 对于位于 STANDARD\_IA 或 ONEZONE\_IA 存储中的对象，当它们在 30 天之前被删除、覆盖或转换到其他存储类时，对剩余的天数会以 GB 为单位按比例收费。
4. 对于位于 STANDARD\_IA 或 ONEZONE\_IA 存储中的小对象（小于 128 KB），当它们在 30 天之前被删除、覆盖或转换到其他存储类时，对剩余的天数会以 GB 为单位按比例收费。

## 跟踪使用率报告中的操作

操作描述指定的使用类型对您的 AWS 对象或存储桶所采取的操作。操作由不言自明的代码表示，如 `PutObject` 或 `ListBucket`。要查看存储桶上的哪些操作生成了特定类型的用途，请使用这些代码。当您创建使用率报告时，可以选择包括 All Operations 或特定操作（例如 `GetObject`）以进行报告。

## 将使用率字节小时数转换为计费 GB 月数

我们每月给您开具账单的存储量以您整个月使用的平均存储量为基础。您需为在您的 AWS 账户下创建的存储桶中存储的所有对象数据和元数据付费。有关元数据的更多信息，请参阅[对象键和元数据 \(p. 86\)](#)。

我们按 `TimedStorage-ByteHrs` 衡量您的存储使用量，每月底合计该值，得出您的月度费用。使用率报告以字节小时数为单位报告您的存储使用量，账单报告以 GB 月数为单位报告存储使用量。要将您的使用率报告与账单报告相关联，您需要将字节小时数转换为 GB 月数。

例如，如果您在 3 月份的前 15 天在存储桶中存储 100 GB (107,374,182,400 字节) 的标准 Amazon S3 存储数据，在 3 月份的后 16 天在存储桶中存储 100 TB (109,951,162,777,600 字节) 的标准 Amazon S3 存储数据，则您使用了 42,259,901,212,262,400 字节小时。

首先，计算字节小时使用量总计：

```
[107,374,182,400 ## x 15 # x (24 ##/#)] + [109,951,162,777,600 ## x 16 # x (24 ##/#)] =  
42,259,901,212,262,400 #####
```

然后将字节小时数转换为 GB 月数：

```
42,259,901,212,262,400 #####/# GB 1,073,741,824 ##/## 24 ##/3 ## 31 # = 52,900 GB #
```

## 更多信息

- [Amazon S3 的 AWS 使用率报告 \(p. 79\)](#)
- [Amazon S3 的 AWS 账单报告 \(p. 77\)](#)
- [Amazon S3 定价](#)
- [Amazon S3 常见问题](#)
- [Amazon Glacier 定价](#)
- [Amazon Glacier 常见问题](#)

## 使用成本分配 S3 存储桶标签

要跟踪单个项目或项目组的存储成本或其他标准，请使用成本分配标签标记您的 Amazon S3 存储桶。成本分配标签是与 S3 存储桶相关联的键-值对。在激活成本分配标签后，AWS 将使用这些标签在成本分配报告上组织您的资源成本。成本分配标签只能用于标记存储桶。有关用于标记对象的标签的信息，请参阅[对象标签 \(p. 96\)](#)。

成本分配报告会按产品类别和 AWS Identity and Access Management (IAM) 用户列出您的账户的 AWS 使用量。该报告包含与详细账单报告相同的行项目（请参阅[了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)）和用于您的标签键的附加列。

AWS 提供了两种类型的成本分配标签：AWS 生成的标签和用户定义的标签。在 Amazon S3 CreateBucket 事件后，AWS 会为您定义、创建和应用 AWS 生成的 `createdBy` 标签。您定义、创建用户定义的 标签并将其应用到 S3 存储桶。

必须在账单和成本管理控制台中分别激活这两种类型的标签，然后它们才能出现在账单报告中。有关 AWS 生成的标签的更多信息，请参阅 [AWS 生成的成本分配标签](#)。有关激活标签的更多信息，请参阅 AWS Billing and Cost Management 用户指南 中的[使用成本分配标签](#)。

用户定义的成本分配标签有以下组成部分：

- 标签键。标签键是标签的名称。例如，在标签 `project/Trinity` 中，`project` 是键。标记键是一个区分大小写的字符串，它可以包含 1 到 128 个 Unicode 字符。
- 标签值。标签值是必需的字符串。例如，在标签 `project/Trinity` 中，`Trinity` 是值。标记值是一个区分大小写的字符串，它可以包含 0 到 256 个 Unicode 字符。

有关用户定义的标签的允许字符以及其他限制的详细信息，请参阅 AWS Billing and Cost Management 用户指南 中的[用户定义的标签限制](#)。

每个 S3 存储桶都有一个标签集。标签集包含了分配到该存储桶的所有标签。一个标签集可以包含多达 10 个标签，也可以为空。键在标签集中必须是唯一的，但标签集中的值不必是唯一的。例如，您可以在名为 `project/Trinity` 和 `cost-center/Trinity` 的标签集中使用相同的值。

在存储桶内，如果您要添加的标签与现有标签的键相同，则新值会覆盖旧值。

AWS 不会对您的标签应用任何语义意义。我们严格按字符串解释标签。

您可以使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI) 或 Amazon S3 API 来添加、列出、编辑或删除标签。

有关创建标签的更多信息，请参阅相应的主题：

- 要在控制台中创建标签，请参阅[如何查看 S3 存储桶的属性？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。
- 要使用 Amazon S3 API 创建标签，请参阅 Amazon Simple Storage Service API Reference 中的[PUT 存储桶标签](#)。
- 要使用 AWS CLI 创建标签，请参阅 AWS CLI Command Reference 中的[put-bucket-tagging](#)。

有关用户定义的标签的更多信息，请参阅 AWS Billing and Cost Management 用户指南 中的[用户定义的成本分配标签](#)。

## 更多信息

- AWS Billing and Cost Management 用户指南 中的[使用成本分配标签](#)
- [了解 Amazon S3 的 AWS 账单和使用率报告 \(p. 80\)](#)
- [Amazon S3 的 AWS 账单报告 \(p. 77\)](#)

# 使用 Amazon S3 对象

Amazon S3 是一种简单的键值存储，专门用于存储任意多的对象。您可以在一个或多个存储桶中存储这些对象。对象由以下内容组成：

- 键 – 分配给对象的名称。您可以使用对象键检索该对象。

有关详细信息，请参阅[对象键和元数据 \(p. 86\)](#)

- 版本 ID – 在存储桶中，键和版本 ID 唯一地标识对象。

版本 ID 是 Amazon S3 在对象添加到存储桶时生成的字符串。有关更多信息，请参阅[对象版本控制 \(p. 94\)](#)。

- 值 – 您正在存储的内容。

对象值可以是任意序列的字节。对象的大小范围是 0 到 5 TB。有关更多信息，请参阅[上传对象 \(p. 150\)](#)。

- 元数据 – 一组名称值对，可用于存储有关对象的信息。

您可以将元数据（称为用户定义的元数据）分配给 Amazon S3 中的对象。Amazon S3 还将系统元数据分配给这些对象，用于管理对象。有关详细信息，请参阅[对象键和元数据 \(p. 86\)](#)。

- 子资源 – Amazon S3 使用子资源机制存储特定于对象的其他信息。

因为子资源从属于对象，因此它们始终与某些其他实体（如对象或存储桶）相关联。有关更多信息，请参阅[对象子资源 \(p. 94\)](#)。

- 访问控制信息 – 您可以控制对您在 Amazon S3 中存储的对象的访问。

Amazon S3 支持基于资源的访问控制（如访问控制列表（ACL）和存储桶策略）和基于用户的访问控制。有关更多信息，请参阅[管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)。

有关使用对象的更多信息，请参阅以下部分。您的 Amazon S3 资源（例如存储桶和对象）在默认情况下是私有的。您需要显式授予权限，才能允许其他人访问这些资源。例如，您可能希望在网站上共享在 Amazon S3 存储桶中存储的视频或照片。只有在网站上将对象设为公开或使用预签名 URL 时，这才有效。有关共享对象的更多信息，请参阅[与其他用户共享数据元 \(p. 147\)](#)。

## 主题

- [对象键和元数据 \(p. 86\)](#)
- [存储类别 \(p. 90\)](#)
- [对象子资源 \(p. 94\)](#)
- [对象版本控制 \(p. 94\)](#)
- [对象标签 \(p. 96\)](#)
- [对象生命周期管理 \(p. 104\)](#)
- [跨源资源共享 \(CORS\) \(p. 133\)](#)
- [在对象上的操作 \(p. 141\)](#)

## 对象键和元数据

每个 Amazon S3 对象都有数据、键和元数据。对象键（或键名称）在存储桶中唯一地标识对象。对象元数据是一组名称值对。您可以在上传对象元数据时对其进行设置。上传对象后，您将无法修改对象元数据。修改对象元数据的唯一方式是创建对象的副本并设置元数据。

## 主题

- [对象键 \(p. 87\)](#)
- [对象元数据 \(p. 89\)](#)

# 对象键

创建对象时，要指定键名称，它在存储桶中唯一地标识该对象。例如，在 Amazon S3 控制台中（参阅 [AWS 管理控制台](#)），在突出显示存储桶时，将显示存储桶中的对象的列表。这些名称是对象键。键的名称是一系列的 Unicode 字符，它的 UTF-8 编码长度最大为 1024 个字节。

## 对象键命名指导原则

虽然您可以在对象键名称中使用任何 UTF-8 字符，但是以下键命名最佳做法有助于确保与其他应用程序的最大兼容性。每个应用程序对特殊字符的分析方式可能不同。以下指导原则有助于最大程度符合 DNS、Web 安全字符、XML 分析器和其他 API 的要求。

### 安全字符

以下字符集通常可安全地用于键名称：

- 字母数字字符 [0-9a-zA-Z]
- 特殊字符 !、-、\_、.、\*、'、( 和 )

以下是有效对象键名称的示例：

- 4my-organization
- my.great\_photos-2014/jan/myvacation.jpg
- videos/2014/birthday/video1.wmv

Amazon S3 数据模型是一种扁平结构：您创建存储桶，存储桶存储对象。不存在子存储桶或子文件夹层次结构；但您可以使用键名称前缀和分隔符推断逻辑层次结构（如同 Amazon S3 控制台一样）。Amazon S3 控制台支持文件夹的概念。假设您的存储桶（admin-created）包含具有以下对象键的四个对象：

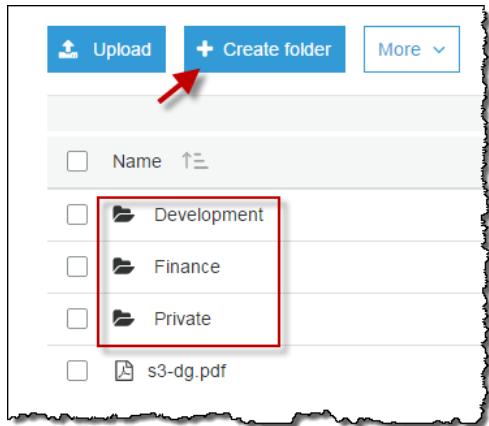
Development/Projects1.xls

Finance/statement1.pdf

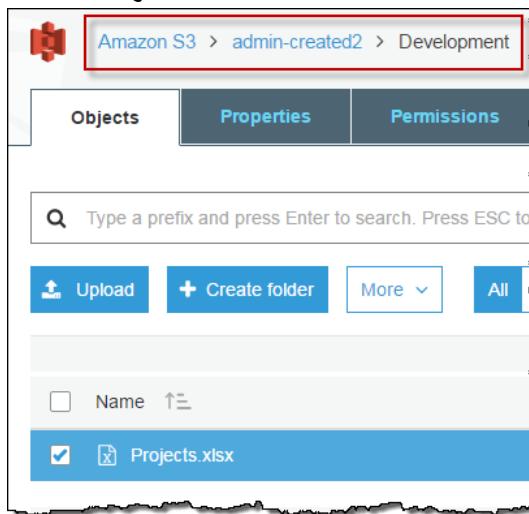
Private/taxdocument.pdf

s3-dg.pdf

控制台使用键名称前缀（Development/、Finance/ 和 Private/）和分隔符（“/”）呈现如下所示的文件夹结构：



s3-dg.pdf 键没有前缀，因此其对象直接在存储桶的根级别出现。如果您打开 Development/ 文件夹，可以看到 Projects.xlsx 对象。



#### Note

Amazon S3 支持存储桶和对象，且 Amazon S3 中没有层次结构。但是，通过对对象键名称中的前缀和分隔符，Amazon S3 控制台和 AWS 开发工具包可以推断层次结构并引入文件夹的概念。

#### 可能需要特殊处理的字符

键名称中的以下字符可能需要另外进行代码处理，并且可能需要以十六进制形式在 URL 中编码或引用。其中部分字符是不可打印的字符，浏览器可能无法处理它们，这也需要特殊处理：

表示和的符号 ("&")	美元 ("\$")	ASCII 字符范围 00–1F 十六进制 (0–31 十进制) 和 7F (127 十进制)
"At" 符号 ("@")	等于 ("=")	分号 (";")
冒号 (":")	加号 ("+")	空格 – 大量连续空格可能会在某些使用情形中丢失 (特别是多个空格)
逗号 (",")	问号 ("?")	

## 要避免的字符

避免在键名称中使用以下字符，因为这些字符需要进行大量的特殊处理，才能在所有应用程序间保持一致性。

反斜杠 ("\"")	左大括号 ("{"")	不可打印的 ASCII 字符 (128–255 十进制字符)
插入符号 ("^")	右大括号 ("}")	百分比字符 ("%")
重音符/反勾号 ("`")	右方括号 ("]")	引号
“大于”符号 (>")	左方括号 (["")	波浪字符 (~")
“小于”符号 (<")	“井号”字符 (#")	竖线 (" ")

## 对象元数据

有两种元数据：系统元数据和用户定义的元数据。

### 系统定义的元数据

对于存储桶中存储的每个对象，Amazon S3 均保留一组系统元数据。Amazon S3 根据需要处理这些系统元数据。例如，Amazon S3 将保留对象创建日期和大小元数据，并将这些信息用作对象管理的一部分。

有两种类别的系统元数据：

- 元数据（如对象创建日期）受系统控制，仅 Amazon S3 可以修改其值。
- 其他系统元数据（例如为对象配置的存储类别以及指示对象是否已启用服务器端加密的数据）是您可以控制其值的系统元数据的示例。如果您的存储桶配置为网站，有时您可能想要将页面请求重定向到其他页面或外部 URL。在这种情况下，网页就是您的存储桶中的对象。Amazon S3 将页面重定向值存储为您可以控制其值的系统元数据。

创建对象时，您可以配置这些系统元数据项目的值或在需要时更新这些值。有关存储类别的更多信息，请参阅 [存储类别 \(p. 90\)](#)。有关服务器端加密的更多信息，请参阅 [使用加密保护数据 \(p. 345\)](#)。

下表提供了系统定义的元数据列表以及您是否可以更新它。

名称	说明	用户是否可以修改该值？
日期	当前日期和时间。	否
内容长度	对象大小以字节为单位。	否
上次修改日期	对象创建日期或上次修改日期（以较晚者为准）。	否
Content-MD5	对象的 base64 编码的 128 位 MD5 摘要。	否
x-amz-server-side-encryption	指示是否为对象启用了服务器端加密，以及加密是来自 AWS Key Management Service (SSE-KMS) 还是来自 AWS 托管加密 (SSE-S3)。有关更多信息，请参阅 <a href="#">使用服务器端加密保护数据 (p. 345)</a> 。	是

名称	说明	用户是否可以修改该值？
x-amz-version-id	对象版本。在存储桶上启用版本控制时，Amazon S3 会为添加到存储桶的对象指定版本号。有关更多信息，请参阅 <a href="#">使用版本控制 (p. 380)</a> 。	否
x-amz-delete-marker	在启用版本控制的存储桶中，此布尔值标记指示对象是否为删除标记。	否
x-amz-storage-class	用于存储对象的存储类别。有关更多信息，请参阅 <a href="#">存储类别 (p. 90)</a> 。	是
x-amz-website-redirect-location	将相关联的对象的请求重定向到同一存储桶中的其他对象或外部 URL。有关更多信息，请参阅 <a href="#">(可选) 配置网页重定向 (p. 408)</a> 。	是
x-amz-server-side-encryption-aws-kms-key-id	如果 x-amz-server-side-encryption 存在并具有值 aws:kms，则它表示用于对象的 AWS Key Management Service (AWS KMS) 主加密密钥的 ID。	是
x-amz-server-side-encryption-customer-algorithm	指示是否启用了具有客户提供的加密密钥的服务器端加密 (SSE-C)。有关更多信息，请参阅 <a href="#">通过使用客户提供的加密密钥的服务器端加密 (SSE-C) 保护数据 (p. 359)</a> 。	是

## 用户定义的元数据

上传对象时，您也可以将元数据指定给该对象。发送 PUT 或 POST 请求创建对象时，您将以名称-值（键-值）对的形式提供此可选信息。如果使用 REST API 上传对象，可选的用户定义的元数据名称必须以“x-amz-meta-”开头，以与其他 HTTP 标头区分开来。使用 REST API 检索对象时，将返回此前缀。如果使用 SOAP API 上传对象则无需前缀。使用 SOAP API 检索对象时，无论您使用哪种 API 上传对象，都将删除前缀。

**Note**

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

通过 REST API 检索元数据时，Amazon S3 会将同名（忽略大小写）标头合并为逗号分隔的列表。如果某些元数据包含不可打印的字符，则不会返回它。但是，x-amz-missing-meta 标头将与不可打印的元数据条目的数量值一起返回。

用户定义的元数据是一组键值对。Amazon S3 使用小写存储用户定义的元数据键。使用 REST 时，每个键-值对都必须符合 US-ASCII；使用 SOAP，或通过 POST 进行基于浏览器的上传时，每个键-值对都必须符合 UTF-8。

**Note**

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

有关在上传对象后向其中添加元数据的信息，请参阅 [如何向 S3 对象添加元数据？](#)（在 Amazon Simple Storage Service 控制台用户指南 中）。

## 存储类别

**主题**

- 经常访问对象的存储类 (p. 91)
- 不经常访问对象的存储类 (p. 91)
- GLACIER 存储类 (p. 92)
- 存储类：比较持久性和可用性 (p. 92)
- 设置对象的存储类 (p. 93)

Amazon S3 中的每个对象都有与之关联的存储类别。例如，如果您列出存储桶中的所有对象，则控制台会在列表中显示所有对象的存储类别。

Name	Last modified	Size	Storage class
notice.pdf	Jul 13, 2016 7:19:13 PM GMT-0700	175.5 KB	Standard
screen-shot.png	Apr 2, 2018 6:47:22 PM GMT-0700	109.8 KB	Standard-IA
screen-shot3.png	Apr 2, 2018 7:08:32 PM GMT-0700	109.8 KB	Standard-IA

Amazon S3 将为您存储的对象提供以下存储类别。您根据使用案例场景和性能访问要求选择一个存储类别。所有这些存储类别都提供高持久性存储。

## 经常访问对象的存储类

对于性能敏感的使用案例 (需要毫秒级访问时间的用例) 和经常访问的数据，Amazon S3 提供以下存储类：

- STANDARD – 默认存储类。如果上传对象时未指定存储类，Amazon S3 会分配 STANDARD 存储类。
- REDUCED\_REDUNDANCY – 低冗余存储 (RRS) 存储类设计用于可使用低于 STANDARD 存储类的冗余级别存储的非关键性可再生数据。

### Important

我们建议您不要使用此存储类。STANDARD 存储类更经济高效。

为了实现持久性，RRS 对象的平均每年对象损失率为 0.01%。如果 RRS 对象丢失，则在对该对象发出请求时，Amazon S3 会返回 405 错误。

## 不经常访问对象的存储类

STANDARD\_IA 和 ONEZONE\_IA 存储类用于长时间运行且不经常访问的数据。(IA 表示不经常访问。)STANDARD\_IA 和 ONEZONE\_IA 对象可用于毫秒级访问 (类似于 STANDARD 存储类)。Amazon S3 会收取这些对象的检索费，因此，它们最适合不经常访问的数据。有关定价信息，请参阅 [Amazon S3 定价](#)。

例如，您可以选择 STANDARD\_IA 和 ONEZONE\_IA 存储类：

- 用于存储备份。

- 用于不经常访问但仍需要毫秒级访问的旧数据。例如，上传数据时，您可能会选择 STANDARD 存储类，然后使用生命周期配置指示 Amazon S3 将对象转换为 STANDARD\_IA 或 ONEZONE\_IA 类。有关生命周期管理的更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

#### Note

STANDARD\_IA 和 ONEZONE\_IA 存储类适用于您计划存储至少 30 天并且大小超过 128 KB 的对象。如果对象小于 128 KB，Amazon S3 会收取 128 KB 的费用。如果您在 30 天的最短使用期限前删除对象，则需支付 30 天的费用。有关定价信息，请参阅 [Amazon S3 定价](#)。

这些存储类在以下方面有所不同：

- STANDARD\_IA – Amazon S3 跨多个地理位置独立的可用区存储冗余对象数据（类似于 STANDARD 存储类）。STANDARD\_IA 对象可在出现可用区丢失时复原。此存储类可提供比 ONEZONE\_IA 类更好的可用性、持久性和弹性。
- ONEZONE\_IA – Amazon S3 只在一个可用区存储对象数据，因此比 STANDARD\_IA 更便宜。但是，数据无法灵活地应对由于地震和洪水灾害而造成可用区物理丢失的情况。ONEZONE\_IA 存储类和 STANDARD\_IA 一样具有持久性，但是可用性和弹性较差。有关存储类的持久性和可用性比较，请参阅此部分结尾的持久性和可用性表。有关定价信息，请参阅 [Amazon S3 定价](#)。

我们建议执行下列操作：

- STANDARD\_IA – 用于主数据或无法重新创建的数据副本。
- ONEZONE\_IA – 如果在可用区出现故障时可重新创建数据，可在设置跨区域复制 (CRR) 时用于对象副本。

## GLACIER 存储类

GLACIER 存储类适用于在不常访问数据的位置对数据存档。存档对象不可用于实时访问。您必须先还原对象，然后才可以访问它们。有关更多信息，请参阅 [恢复存档对象 \(p. 223\)](#)。此存储类提供与 STANDARD 存储类相同的持久性和弹性。

#### Important

如果选择 Glacier 存储类，Amazon S3 将使用低成本 Amazon Glacier 服务来存储对象。尽管对象存储在 Amazon Glacier 中，它们仍是 Amazon S3 对象，需在 Amazon S3 中管理；并且您无法直接通过 Amazon Glacier 访问它们。

请注意以下关于 GLACIER 存储类别的信息：

- 您无法在创建对象时指定 GLACIER 作为存储类别。您在创建 GLACIER 对象时先使用 STANDARD、RRS、STANDARD\_IA 或 ONEZONE\_IA 作为存储类别来上传这些对象。然后，使用生命周期管理将这些对象转换为 GLACIER 存储类别。有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。
- 您必须先将 GLACIER 对象还原才能访问它们 (STANDARD、RRS、STANDARD\_IA 和 ONEZONE\_IA 对象可随时访问)。有关更多信息，请参阅 [转换为 GLACIER 存储类 \(对象存档\) \(p. 107\)](#)。

要了解有关 Amazon Glacier 服务的更多信息，请参阅 [Amazon Glacier 开发人员指南](#)。

## 存储类：比较持久性和可用性

下表总结了每个存储类别提供的持久性和可用性。

存储类别	持久性 (设计目标)	可用性 (设计目标)	其他考虑因素		
STANDARD	99.999999999%	99.99%	无		
STANDARD_IA	99.999999999%	99.9%	STANDARD_IA 对象会收取检索费用。此类最适合不经常访问的数据。 有关定价信息，请参阅 <a href="#">Amazon S3 定价</a> 。		
ONEZONE_IA	99.999999999%	99.5%	无法灵活地应对可用区丢失的情况。		
GLACIER	99.999999999%	99.99% (在您还原对象之后)	GLACIER 对象不可用于实时访问。 您必须先还原存档对象，然后才可以访问它们。有关更多信息，请参阅 <a href="#">恢复存档对象 (p. 223)</a> 。		
RRS	99.99%	99.99%	无		

除 ONEZONE\_IA 之外的所有存储类均采用灵活的弹性设计，适用于单个可用区同时丢失完整数据以及另一个可用区丢弃部分数据的情况。

除了应用程序场景的性能要求之外，还考虑了价格问题。有关存储类定价，请参阅 [Amazon S3 定价](#)。

## 设置对象的存储类

Amazon S3 API 支持按照以下方式设置 (或更新) 对象的存储类：

- 创建新对象时，可以指定其存储类。例如，使用 [PUT 对象](#)、[POST 对象](#) 和 [启动分段上传](#) API 时，添加 `x-amz-storage-class` 请求标头以指定存储类。如果您未添加此标头，Amazon S3 将 STANDARD 存储类作为默认存储类。
- 您还可以通过使用 [PUT Object - Copy](#) API 复制对象来更改已存储在 Amazon S3 中的对象。您使用相同键名称复制相同存储桶中的对象并按照以下方式指定请求标头：
  - 将 `x-amz-metadata-directive` 标头设置为 COPY。
  - 将 `x-amz-storage-class` 设置为要使用的存储类。

在启用版本控制的存储桶中，您无法更改特定版本对象的存储类。当您复制对象时，Amazon S3 将为其指定新的版本 ID。

- 您可以通过向存储桶添加生命周期配置来指示 Amazon S3 更改对象的存储类。有关更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

要创建和更新对象存储类，您可以使用 Amazon S3 控制台、AWS 开发工具包或 AWS Command Line Interface (AWS CLI)。每个都使用 Amazon S3 API 向 Amazon S3 发送请求。

## 对象子资源

Amazon S3 定义一组与存储桶和对象相关联的子资源。子资源从属于对象；即，子资源不会自行存在，它们始终与某些其他实体（例如对象或存储桶）相关联。

下表列出了与 Amazon S3 对象相关联的子资源。

子资源	说明
acl	包含可以识别被授权者和所授予的许可的授权列表。创建对象时，acl 将识别可以完全控制对象的对象所有者。您可以检索对象 ACL 或将其替换为更新的授权列表。对 ACL 的任何更新都需要您替换现有 ACL。有关 ACL 的更多信息，请参阅 <a href="#">使用 ACL 管理访问 (p. 333)</a> 。
torrent	Amazon S3 支持 BitTorrent 协议。Amazon S3 使用 torrent 子资源返回与特定对象相关的 torrent 文件。要检索 torrent 文件，您需在 GET 请求中指定 torrent 子资源。Amazon S3 创建 torrent 文件并返回它。您只能检索 torrent 子资源，不能创建、更新或删除 torrent 子资源。有关更多信息，请参阅 <a href="#">将 BitTorrent 与 Amazon S3 配合使用 (p. 496)</a> 。

## 对象版本控制

使用版本控制可在存储桶中保留多个版本的对象。例如，可在存储桶中存储 my-image.jpg (版本 111111) 和 my-image.jpg (版本 222222)。版本控制可保护您免于造成意外覆盖或删除的后果。您还可以使用控制版本为对象存档，以便访问早期版本。

### Note

SOAP API 不支持版本控制。HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。

要自定义您的数据保留方法和控制存储成本，请将对象版本控制与[对象生命周期管理 \(p. 104\)](#)结合使用。有关使用 AWS 管理控制台创建生命周期策略的信息，请参阅[如何创建 S3 存储桶的生命周期策略？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

如果您在不受版本控制的存储桶中具有对象到期生命周期策略，并且希望在启用版本控制时保持相同的永久删除行为，则必须添加非当前版本到期策略。非当前版本到期生命周期策略将管理在受版本控制的存储桶中删除非当前对象版本的行为。（启用版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。）

您必须在您的存储桶上显式启用版本控制。默认情况下，版本控制处于禁用状态。无论您是否已启用版本控制，您的存储桶中的每个对象都具有版本 ID。如果未启用版本控制，Amazon S3 将版本 ID 值设置为空。如果已启用版本控制，则 Amazon S3 为对象指定唯一版本 ID 值。如果对存储桶启用版本控制，存储桶中已存储的对象不会发生更改。版本 ID (空)、内容和权限保持不变。

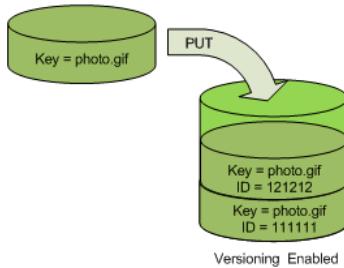
已在存储桶级别上完成启用和暂停版本控制。启用存储桶的版本控制时，向其添加的所有对象将具有唯一版本 ID。唯一版本 ID 可随机生成，包括 Unicode、UTF-8 编码、准备就绪的 URL、不透明

字符串，长度最大为 1024 个字节。示例版本 ID 是 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCx3v+jVBH40Nr8X8gdRQpUMLUo。仅 Amazon S3 可以生成版本 ID。无法对其进行编辑。

Note

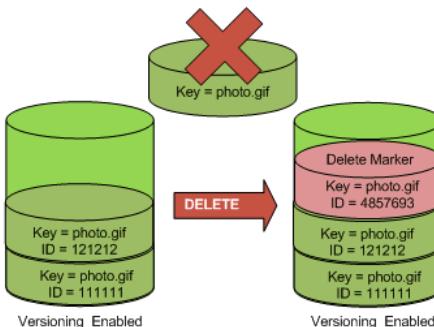
为了获得简易性，我们将在我们的所有示例中使用较为简短的 ID。

当您在启用版本控制的存储桶中通过 PUT 放入对象时，不会覆盖非当前版本。下图显示了当将新版本的 photo.gif PUT 在已包含具有相同名称的对象的存储桶中时，原始对象 (ID = 111111) 将保留在该存储桶中，Amazon S3 将生成新版本 ID (121212)，并将较新版本添加到该存储桶。

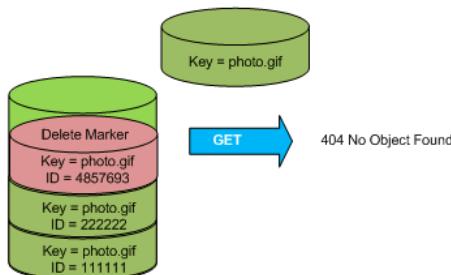


此功能将防止您意外覆盖或删除对象，并向您提供检索早期版本的对象的机会。

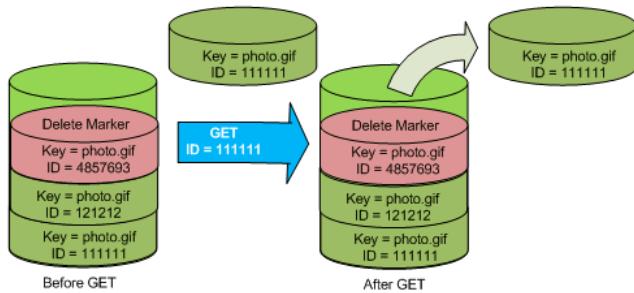
DELETE 对象时，所有版本都将保留在存储桶中，且 Amazon S3 将插入删除标记，如下图所示。



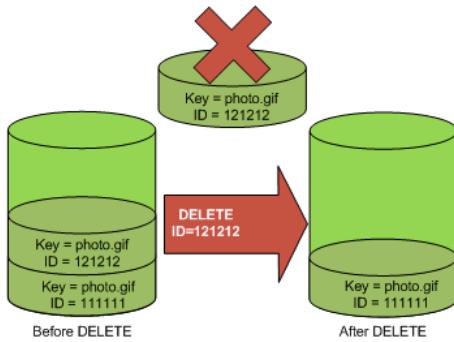
删除标记将成为对象的当前版本。默认情况下，GET 请求将检索最新存储的版本。在当前版本为删除标记时，执行简单 GET Object 请求将返回 404 Not Found 错误，如下图所示。



但是，您可以通过指定对象版本 ID，通过 GET 获取非当前版本的对象。在下图中，我们 GET 特定对象版本 111111。即使该对象版本不是当前版本，Amazon S3 也会返回它。



您可以通过指定要删除的版本来永久删除对象。只有 Amazon S3 存储桶的拥有者才能永久删除某个版本。下图显示了 `DELETE versionId` 如何永久删除存储桶中的对象以及 Amazon S3 不会插入删除标记。



您可以通过配置存储桶来启用 MFA (多因素认证) 删除，从而增加额外的安全性。执行此操作时，存储桶拥有者必须在任何请求中包含两种形式的身份验证，以删除版本或更改存储桶的版本控制状态。有关详细信息，请参阅 [MFA 删除 \(p. 381\)](#)。

#### Important

如果您注意到启用版本控制后，Amazon S3 对存储桶的 PUT 或 DELETE 对象请求的 HTTP 503 慢速响应数量显著增加，那么存储桶中可能有一个或多个对象有数以百万计的版本。有关更多信息，请参阅 [Amazon S3 疑难解答 \(p. 502\)](#)。

有关详细信息，请参阅 [使用版本控制 \(p. 380\)](#)。

## 对象标签

使用对象标签对存储进行分类。每个标签都是一个键-值对。请考虑以下标签示例：

- 假设某个对象包含受保护医疗信息 (PHI) 数据。您可以使用以下键-值对标记该对象，如下所示：

```
PHI=True
```

或者

```
Classification=PHI
```

- 假设您将项目文件存储在 S3 存储桶中。您可以使用一个名为 Project 的键和一个值标记这些对象，如下所示：

```
Project=Blue
```

- 您可以将多个标签添加到一个对象，如下所示：

```
Project=x
Classification=confidential
```

您可以将标签添加到新对象（当您上传新对象时），也可以将标签添加到现有对象。请注意以下几点：

- 您最多可以将 10 个标签与对象关联。与对象关联的标签必须具有唯一的标签键。
- 标签键的长度最大可以为 128 个 Unicode 字符，标签值的长度最大可以为 256 个 Unicode 字符。
- 键和值区分大小写。

对象键名称前缀还使您可以对存储进行分类，但基于前缀的分类是一维的。请考虑以下对象键名称：

```
photos/photo1.jpg
project/projectx/document.pdf
project/projecty/document2.pdf
```

这些键名称具有前缀 photos/、project/projectx/ 和 project/projecty/。这些前缀支持一维分类。即，一个前缀下的一切都属于一个类别。例如，前缀 project/projectx 可确定与项目 x 相关的所有文档。

利用标签，您现在获得了另一个维度。如果您希望 photo1 属于项目 x 类别，则可以相应地标记该对象。除了数据分类之外，标签还提供其他好处。例如，

- 对象标签支持权限的精细访问控制。例如，您可以向一个 IAM 用户授予仅读取带有特定标签的对象的权限。
- 对象标签支持精细的对象生命周期管理，在其中，除了在生命周期规则中指定键名称前缀之外，还可以指定基于标签的筛选条件。
- 使用 Amazon S3 分析时，您可以配置筛选条件，以便按对象标签、键名称前缀或前缀和标签的组合对对象进行分组以进行分析。
- 您还可以自定义 Amazon CloudWatch 指标以按特定标签筛选条件显示信息。以下各节提供了详细信息。

#### Important

尽管使用标签来标记包含机密数据（如个人身份信息 (PII) 或受保护医疗信息 (PHI)）的对象是可以接受的，但标签本身不应包含任何机密信息。

## 与对象标签相关的 API 操作

Amazon S3 支持特定于对象标签的以下 API 操作：

### 对象 API 操作

- [PUT Object 标签](#) - 替换对象上的标签。您可以在请求正文中指定标签。使用此 API 的对象标签管理有两个不同的情形。
  - 对象没有标签 - 利用此 API，您可以将一组标签添加到某个对象（该对象没有以前的标签）。
  - 对象有一组现有标签 - 要修改现有标签，您必须先检索现有标签集，在客户端侧修改它，然后使用此 API 替换它。如果您发送带有空标签集的此请求，S3 将删除对象上的现有标签集。
- [GET Object 标签](#) - 返回与某个对象关联的标签集。Amazon S3 将在响应正文中返回对象标签。
- [DELETE 对象标签](#) - 删除与某个对象关联的标签集。

### 支持标签的其他 API 操作

- [PUT 对象和开始分段上传](#) – 您可以在创建对象时指定标签。您将使用 `x-amz-tagging` 请求标头指定标签。
- [GET Object](#) - Amazon S3 不会返回标签集，而会返回 `x-amz-tag-count` 标头中的对象标签计数（仅当请求者有权读取标签时），因为标头响应大小限制为 8 K 个字节。如果要查看标签，您应该再提出一个 [GET Object 标签 API](#) 操作请求。
- [POST Object](#) - 您可以在 POST 请求中指定标签。  
只要请求中的标签不超过 8 K 个字节的 HTTP 请求标头大小限制，您就可以使用 `PUT Object` API 创建带标签的对象。如果您指定的标签超过了标头大小限制，您可以使用将标签包含在正文中的此 POST 方法。
- [PUT Object - Copy](#) - 您可以在请求中指定 `x-amz-tagging-directive` 以指示 Amazon S3 复制（默认行为）标签或将标签替换为请求中提供的一组新标签。

请注意以下几点：

- 标签采用了最终一致性模型。也就是说，在将标签添加到某个对象后，如果您很快尝试检索标签，您可能获得该对象上的旧标签（如果有）。但是，后续调用可能提供更新后的标签。

## 对象标签和其他信息

本节介绍对象标签如何与其他配置关联。

### 对象标签和生命周期管理

在存储桶生命周期配置中，您可以指定筛选条件以选择该规则适用的一部分对象。您可以基于键名称前缀、对象标签或两者指定筛选条件。

假设您将照片（原始格式和已完成格式）存储在 Amazon S3 存储桶中。您可以按下面所示标记这些对象：

```
phototype=raw
or
phototype=finished
```

您可能考虑有时候在创建原始照片后将其存档到 Amazon Glacier。您可以在生命周期规则中配置一个筛选条件，用于确定一部分包含特定标签（`phototype=raw`）且带有键名称前缀（`photos/`）的对象。

有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。

### 对象标签和跨区域复制 (CRR)

如果您在存储桶上配置了跨区域复制（CRR），Amazon S3 将复制标签，前提是您向 S3 授予了读取标签的权限。有关更多信息，请参阅 [设置跨区域复制 \(p. 445\)](#)。

### 对象标签和访问控制策略

您还可以使用权限策略（存储桶和用户策略）管理对象标签相关权限。有关策略操作，请参阅以下主题：

- [对象操作的权限 \(p. 282\)](#)

- 与存储桶操作相关的权限 (p. 284)

对象标签支持用于管理权限的精细访问控制。您可以基于对象标签授予条件权限。Amazon S3 支持以下条件键，这些键可用于授予基于对象标签的条件权限。

- s3:ExistingObjectTag/<tag-key> - 使用此条件键可验证现有对象标签是否有特定标签键和值。

Note

当授予 PUT Object 和 DELETE Object 操作的权限时，此条件键不受支持。也就是说，您无法创建这样一个策略：允许或拒绝用户基于现有对象的现有标签删除或覆盖该对象。

- s3:RequestObjectTagKeys - 使用此条件键可限制要在对象上允许的标签键。当使用 PutObjectTagging 和 PutObject 以及 POST 对象请求将标签添加到对象时，这很有用。
- s3:RequestObjectTag/<tag-key> - 使用此条件键可限制要在对象上允许的标签键和值。当使用 PutObjectTagging 和 PutObject 以及 POST 存储桶请求将标签添加到对象时，这很有用。

有关特定于 Amazon S3 服务的条件键的完整列表，请参阅[可用条件键 \(p. 287\)](#)。以下权限策略说明了对象标签如何支持精细访问权限管理。

Example 1: 允许用户仅读取具有特定标签的对象

以下权限策略将向用户授予读取对象的权限，但条件将读取权限限制为只有具有以下特定标签键和值的对象：

```
security : public
```

请注意，该策略使用 Amazon S3 条件键 s3:ExistingObjectTag/<tag-key> 来指定键和值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/security": "public"
                }
            }
        ]
    }
}
```

Example 2: 允许用户添加对允许的标签键有限制的对象标签

以下权限策略将向用户授予执行 s3:PutObjectTagging 操作的权限，这使用户可以将标签添加到现有对象。条件限制了用户可使用的标签键。条件使用 s3:RequestObjectTagKeys 条件键指定一组标签键。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObjectTagging"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket/*"  
            ],  
            "Condition": {  
                "ForAllValues:StringLike": {  
                    "s3:RequestObjectTagKeys": [  
                        "Owner",  
                        "CreationDate"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

该策略确保了标签集 (如果已在请求中指定) 包含指定的键。用户可以在 PutObjectTagging 中发送空标签集，这是该策略允许的 (请求中的空标签集将删除对象上的任何现有标签)。如果您要阻止用户删除标签集，则可以添加另一个条件来确保用户至少提供一个值。条件中的 ForAnyValue 确保了请求中必须至少存在一个值。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObjectTagging"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket/*"  
            ],  
            "Condition": {  
                "ForAllValues:StringLike": {  
                    "s3:RequestObjectTagKeys": [  
                        "Owner",  
                        "CreationDate"  
                    ]  
                },  
                "ForAnyValue:StringLike": {  
                    "s3:RequestObjectTagKeys": [  
                        "Owner",  
                        "CreationDate"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

有关更多信息，请参阅 IAM 用户指南 中的[创建测试多个键值的条件 \(集合运算\)](#)。

#### Example 3: 允许用户添加包含特定标签键和值的对象标签

以下用户策略将向用户授予执行 s3:PutObjectTagging 操作的权限，这使用户可以在现有对象上添加标签。条件要求用户包含值设置为 x 的特定标签 (Project)。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObjectTagging"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket/*"  
            ],  
            "Condition": {  
                "StringEquals": {  
                    "s3:RequestObjectTag/Project": "X"  
                }  
            }  
        }  
    ]  
}
```

#### 相关主题

[管理对象标签 \(p. 101\)](#)

## 管理对象标签

本节介绍如何使用适用于 Java 的 AWS 开发工具包或使用 Amazon S3 控制台以编程方式添加对象标签。

#### 主题

- [使用控制台管理对象标签 \(p. 101\)](#)
- [使用AWS SDK for Java管理标签 \(p. 101\)](#)
- [使用适用于 .NET 的 AWS 开发工具包管理标签 \(p. 102\)](#)

## 使用控制台管理对象标签

您可以使用 Amazon S3 控制台将标签添加到新对象（当您上传新对象时）或现有对象。有关如何使用 Amazon S3 控制台将标签添加到对象的说明，请参阅 Amazon Simple Storage Service 控制台用户指南 中的[添加对象标签](#)。

## 使用AWS SDK for Java管理标签

以下示例演示如何使用AWS SDK for Java为新对象设置标签并检索或替换现有对象的标签。有关对象标签的更多信息，请参阅[对象标签 \(p. 96\)](#)。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.File;  
import java.util.ArrayList;  
import java.util.List;  
  
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.SdkClientException;  
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.services.s3.AmazonS3;  
import com.amazonaws.services.s3.AmazonS3ClientBuilder;  
import com.amazonaws.services.s3.model.*;  
  
public class ManagingObjectTags {
```

```
public static void main(String[] args) {
    String clientRegion = "**** Client region ****";
    String bucketName = "**** Bucket name ****";
    String keyName = "**** Object key ****";
    String filePath = "**** File path ****";

    try {
        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
            .withCredentials(new ProfileCredentialsProvider())
            .withRegion(clientRegion)
            .build();

        // Create an object, add two new tags, and upload the object to Amazon S3.
        PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName, new
File(filePath));
        List<Tag> tags = new ArrayList<Tag>();
        tags.add(new Tag("Tag 1", "This is tag 1"));
        tags.add(new Tag("Tag 2", "This is tag 2"));
        putRequest.setTagging(new ObjectTagging(tags));
        PutObjectResult putResult = s3Client.putObject(putRequest);

        // Retrieve the object's tags.
        GetObjectTaggingRequest getTaggingRequest = new
GetObjectTaggingRequest(bucketName, keyName);
        GetObjectTaggingResult getTagsResult =
s3Client.getObjectTagging(getTaggingRequest);

        // Replace the object's tags with two new tags.
        List<Tag> newTags = new ArrayList<Tag>();
        newTags.add(new Tag("Tag 3", "This is tag 3"));
        newTags.add(new Tag("Tag 4", "This is tag 4"));
        s3Client.setObjectTagging(new SetObjectTaggingRequest(bucketName, keyName, new
ObjectTagging(newTags)));
    }
    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();
    }
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包管理标签

以下示例演示如何使用适用于 .NET 的 AWS 开发工具包为新对象设置标签并检索或替换现有对象的标签。有关对象标签的更多信息，请参阅[对象标签 \(p. 96\)](#)。

有关如何创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

```
namespace Amazon.DocSamples.S3
{
    public class ObjectTagsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for the new object ***";
        private const string filePath = @ "*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            PutObjectWithTagsTestAsync().Wait();
        }

        static async Task PutObjectWithTagsTestAsync()
        {
            try
            {
                // 1. Put 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);
                // 2. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest = new GetObjectTaggingRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                GetObjectTaggingResponse objectTags = await
                client.GetObjectTaggingAsync(getTagsRequest);
                for (int i = 0; i < objectTags.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}", objectTags.Tagging[i].Key,
objectTags.Tagging[0].Value);

                // 3. Replace the tagset.

                Tagging newTagSet = new Tagging();
                newTagSet.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);
            }
        }
    }
}
```

```
// 4. Retrieve the object's tags.  
GetObjectTaggingRequest getTagsRequest2 = new GetObjectTaggingRequest();  
getTagsRequest2.BucketName = bucketName;  
getTagsRequest2.Key = keyName;  
GetObjectTaggingResponse objectTags2 = await  
client.GetObjectTaggingAsync(getTagsRequest2);  
for (int i = 0; i < objectTags2.Tagging.Count; i++)  
    Console.WriteLine("Key: {0}, Value: {1}", objectTags2.Tagging[i].Key,  
objectTags2.Tagging[0].Value);  
  
}  
catch (AmazonS3Exception e)  
{  
    Console.WriteLine(  
        "Error encountered ***. Message:'{0}' when writing an object"  
        , e.Message);  
}  
catch (Exception e)  
{  
    Console.WriteLine(  
        "Encountered an error. Message:'{0}' when writing an object"  
        , e.Message);  
}  
}  
}  
}
```

## 对象生命周期管理

要管理您的对象以使其在整个生命周期内经济高效地存储，请配置其生命周期。生命周期配置是一组规则，用于定义 Amazon S3 对一组对象应用的操作。有两种类型的操作：

- 转换操作 – 定义对象转换为另一个[存储类](#)的时间。例如，您可以选择在对象创建 30 天后将其转换为 STANDARD\_IA 存储类，或在对象创建 1 年后将其存档到 GLACIER 存储类。

存在与生命周期转换请求关联的成本。有关定价信息，请参阅 [Amazon S3 定价](#)。

- 过期操作 – 定义对象的过期时间。Amazon S3 将代表您删除过期的对象。

生命周期过期成本取决于您选择过期对象的时间。有关更多信息，请参阅 [配置对象过期 \(p. 109\)](#)。

有关生命周期的更多信息，请参阅 [生命周期配置元素 \(p. 110\)](#)。

## 我应何时使用生命周期配置？

为明确定义了生命周期的对象定义生命周期配置规则。例如：

- 如果您将定期日志上传到一个存储桶，您的应用程序可能需要使用这些日志一个星期或一个月。之后，您可能需要删除这些日志。
- 在限定的时间段内可能需要经常访问某些文档。自此之后，这些文档很少被访问。有时，您可能不需要对这些文档进行实时访问，但是您的组织或法规可能要求您将它们存档一段特定的时间。之后，您可以删除这些文档。

- 您可以主要为了存档目的而将一些类型的数据上传到 Amazon S3。例如，您可以存档数字媒体、财务和健康记录、原始基因组序列数据、长期数据库备份，以及为遵从法规而必须保留的数据。

利用生命周期配置规则，您可以指示 Amazon S3 将对象转换为较低成本的存储类，或者存档或删除它们。

## 如何配置生命周期？

生命周期配置 (XML 文件) 由一组规则组成，这些规则预定义了您希望 Amazon S3 在对象的生命周期内对对象执行的操作。

Amazon S3 提供了一组用于在存储桶上管理生命周期配置的 API 操作。Amazon S3 将该配置存储为附加到存储桶的生命周期子资源。有关详细信息，请参阅：

[PUT Bucket lifecycle](#)

[GET Bucket lifecycle](#)

[DELETE Bucket lifecycle](#)

您还可以使用 Amazon S3 控制台配置生命周期，或是使用 AWS 开发工具包包装程序库以编程方式配置生命周期。如果需要，您还可以直接进行 REST API 调用。有关更多信息，请参阅 [在存储桶上设置生命周期配置 \(p. 124\)](#)。

有关更多信息，请参阅以下主题：

- [生命周期配置的其他注意事项 \(p. 105\)](#)
- [生命周期配置元素 \(p. 110\)](#)
- [生命周期配置的示例 \(p. 115\)](#)
- [在存储桶上设置生命周期配置 \(p. 124\)](#)

## 生命周期配置的其他注意事项

在配置对象的生命周期时，您需要了解以下有关转换对象、设置过期日期和其他对象配置的指南。

主题

- [转换对象 \(p. 105\)](#)
- [配置对象过期 \(p. 109\)](#)
- [生命周期和其他存储桶配置 \(p. 109\)](#)

## 转换对象

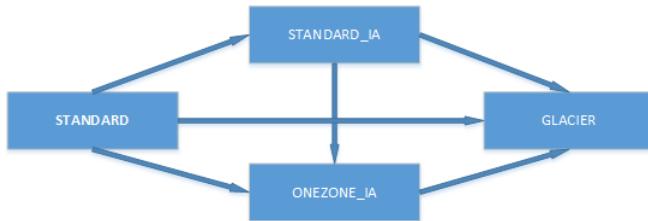
您可以在生命周期配置中添加规则以指示 Amazon S3 将对象转换为另一个 Amazon S3 存储类。例如：

- 当您知道对象不常访问时，您可能会将其转换为 STANDARD\_IA 存储类。
- 您可能想要将不需要实时访问的对象存档到 GLACIER 存储类。

以下各节介绍了受支持的转换、相关限制和到 GLACIER 存储类的转换。

### 受支持的转换和相关限制

在生命周期配置中，您可以定义将对象从一个存储类转换为另一个存储类以及保存在该存储类中的规则。下图显示了支持的存储类转换。



Amazon S3 支持用于带生命周期配置的存储类之间的转换的瀑布模型：

- 从 STANDARD 存储类到 STANDARD\_IA 或 ONEZONE\_IA。以下限制适用：
  - 对于较大的对象，转换到 STANDARD\_IA 或 ONEZONE\_IA 可获得成本效益。Amazon S3 不会将小于 128 KB 的对象转换为 STANDARD\_IA 或 ONEZONE\_IA 存储类，因为这样做不具有成本效益。
  - 必须在当前存储类中将对象存储至少 30 天，然后才能将对象转换为 STANDARD\_IA 或 ONEZONE\_IA。例如，您无法创建用于在对象创建 1 天后将其转换为 STANDARD\_IA 存储类的生命周期规则。

Amazon S3 不会在第一个 30 天内转换对象，因为较新的对象的访问频率或删除速度通常高于 STANDARD\_IA 或 ONEZONE\_IA 存储应有的值。

- 如果您在转换非当前对象（在受版本控制的存储桶中），则只能将至少在 30 天内是非当前版本的对象转换为 STANDARD\_IA 或 ONEZONE\_IA 存储。

**Note**

上图未提及 REDUCED\_REDUNDANCY 存储类，因为我们建议不使用该类别。有关存储类别的信息，请参阅[存储类别 \(p. 90\)](#)。

- 从 STANDARD\_IA 到 ONEZONE\_IA。以下限制适用：
  - 必须在 STANDARD\_IA 存储类中将对象存储至少 30 天，然后才能将对象转换为 ONEZONE\_IA 类别。
- 从任何存储类别到 GLACIER。

您可以组合这些生命周期操作来管理对象的完整生命周期。例如，假设您创建了具有明确定义的生命周期的对象。最初，这些对象在 30 天的周期内可能被经常访问。紧接着，对象在长达 90 天内不常访问。之后，不再需要对象，您可能选择存档或删除对象。在此方案中，您创建一个生命周期规则，用于指定到 STANDARD\_IA（或 ONEZONE\_IA）存储的初始转换操作，并指定到用于存档的 GLACIER 存储的另一个转换操作以及一个过期操作。在将对象从一个存储类移至另一个存储类时，可节省存储成本。有关成本考虑的更多信息，请参阅[Amazon S3 定价](#)。

**Note**

当 GLACIER 转换在 STANDARD\_IA（或 ONEZONE\_IA）转换发生后的不到 30 天内发生时，您无法同时为 STANDARD\_IA（或 ONEZONE\_IA）和 GLACIER 转换指定一个生命周期规则。这是因为，存在与 STANDARD\_IA 和 ONEZONE\_IA 存储类关联的最少 30 天存储费用。在指定从 STANDARD\_IA 存储到 ONEZONE\_IA 存储的转换时，此最少 30 天存储费用也将适用。您可以指定两个规则来实现这一点，而只需支付最少存储费用。有关成本考虑的更多信息，请参阅[Amazon S3 定价](#)。

以下转换不受支持：

- 您无法从 STANDARD\_IA (或 ONEZONE\_IA) 存储类转换为 STANDARD 或 REDUCED\_REDUNDANCY 类别。
- 您无法从 STANDARD\_IA 存储类转换为 STANDARD\_IA 存储类。
- 您无法从 GLACIER 存储类转换为任何其他存储类。
- 您无法从任何存储类转换为 REDUCED\_REDUNDANCY。

## 转换为 GLACIER 存储类 (对象存档)

通过生命周期配置，可以将对象转换为 GLACIER 存储类，也就是将数据存档到 Amazon Glacier (一个成本较低的存储解决方案)。

### Important

如果选择 Glacier 存储类，Amazon S3 将使用低成本 Amazon Glacier 服务来存储对象。尽管对象存储在 Amazon Glacier 中，它们仍是 Amazon S3 对象，需在 Amazon S3 中管理；并且您无法直接通过 Amazon Glacier 访问它们。

在存档对象之前，请查看以下章节中的相关注意事项。

### 一般注意事项

下面是存档对象之前的一般注意事项：

- 加密对象在整个存储类转换过程中保持加密状态。
- GLACIER 存储类中的对象无法实时提供。

存档的对象是 Amazon S3 对象，但在您可以访问某个存档的对象之前，您必须先还原它的临时副本。根据您在恢复请求内指定的持续时间，恢复的对象副本仅在该期间内可用。在此之后，Amazon S3 会删除临时副本，而对象仍在 Amazon Glacier 中存档。

您可以使用 Amazon S3 控制台还原对象，也可以在代码中使用 AWS 开发工具包包装程序库或 Amazon S3 REST API 以编程方式还原对象。有关更多信息，请参阅 [恢复存档对象 \(p. 223\)](#)。

- 对象到 GLACIER 存储类的转换是单向的。

您无法使用生命周期配置规则将对象的存储类从 GLACIER 转换为 STANDARD 或 REDUCED\_REDUNDANCY 存储类。如果要将已存档对象的存储类更改为 STANDARD 或 REDUCED\_REDUNDANCY，您必须首先使用还原操作制作一个临时副本。然后使用复制操作将对象覆盖为 STANDARD、STANDARD\_IA、ONEZONE\_IA 或 REDUCED\_REDUNDANCY 对象。

- GLACIER 存储类对象只能通过 Amazon S3 (而不能通过 Amazon Glacier) 查看和使用。

Amazon S3 会将已存档对象存储在 Amazon Glacier 中。但是，这些对象是 Amazon S3 对象，您只能通过使用 Amazon S3 控制台或 Amazon S3 API 访问它们。您无法通过 Amazon Glacier 控制台或 Amazon Glacier API 访问这些已存档对象。

## 与成本相关的注意事项

如果您计划在数月或数年的时间内存档不经常访问的数据，则 GLACIER 存储类通常可降低您的存储成本。但是，您应考虑以下事项以确保 GLACIER 存储类适合于您：

- **存储开销费用** - 将对象转换为 GLACIER 存储类时，需向每个对象添加固定存储量以容纳用于管理对象的元数据。
  - 对于存档到 Amazon Glacier 的每个对象，Amazon S3 将 8 KB 存储用于对象和其他元数据的名称。Amazon S3 将存储此元数据，以便您可以使用 Amazon S3 API 获取已存档对象的实时列表。有关更多信息，请参阅 [Get Bucket \(List Objects\)](#)。将按照标准 Amazon S3 费率对此附加存储收费。
  - 对于每个存档对象，Amazon Glacier 添加 32 KB 的存储用于索引及相关元数据。标识和还原对象需要此额外数据。将按照 Amazon Glacier 费率对此附加存储收费。

如果您打算存档小对象，请考虑这些存储费用。还可以考虑将许多小型对象合并为少量大型对象，以便减少开销成本。

- **计划存档对象的天数** - Amazon Glacier 是长期存档解决方案。如果删除的对象存档三个月或更长时间，则删除存档到 Amazon Glacier 的数据是免费的。如果在存档的三个月内删除或覆盖对象，则 Amazon S3 将收取按比例计算的提早删除费。
- **Glacier 存档请求费用** - 每个转换为 GLACIER 存储类的对象都构成一个存档请求。每个这类请求都有成本。如果您计划转换大量对象，则考虑这些请求费用。
- **Glacier 数据还原费用** - Amazon Glacier 旨在用于长期存档您不经常访问的数据。有关数据还原费用的信息，请参阅[从 Glacier 检索数据的成本是多少？](#) (在 Amazon S3 常见问题中)。有关如何从 Glacier 还原数据的信息，请参阅[恢复存档对象 \(p. 223\)](#)。

当通过使用对象生命周期管理将对象存档到 Amazon Glacier 时，Amazon S3 会异步转换这些对象。生命周期配置规则中的转换日期与实际转换日期之间可能存在延迟。收取的 Amazon Glacier 价格基于此规则中指定的转换日期。

Amazon S3 产品详细信息页面针对存档 Amazon S3 对象提供了定价信息和示例计算。有关更多信息，请参阅以下主题：

- 对于存档到 Amazon Glacier 的 Amazon S3 对象，[我的存储费用是如何计算的？](#)
- [删除存储在 Amazon Glacier 中不到 3 个月的对象时，如何收费？](#)
- [从 Glacier 检索数据的成本是多少？](#)
- 针对标准和存储类的存储成本的 [Amazon S3 定价 GLACIER](#)。

## 恢复存档对象

已存档对象无法实时访问。您必须首先启动恢复请求，然后耐心等待，直到对象的临时副本根据您在请求中指定的持续时间变为可用。在收到已还原对象的临时副本后，此对象的存储类仍保持为 GLACIER (GET 或 HEAD 请求将返回 GLACIER 作为存储类)。

#### Note

还原某个存档时，您需要同时为存档 (GLACIER 费率) 和临时还原的副本 (REDUCED\_REDUNDANCY 存储费率) 付费。有关定价的信息，请参阅 [Amazon S3 定价](#)。

您可以采用编程方式或使用 Amazon S3 控制台还原对象副本。Amazon S3 针对每个对象每次仅处理一个还原请求。有关更多信息，请参阅 [恢复存档对象 \(p. 223\)](#)。

## 配置对象过期

在对象的生存期结束后，Amazon S3 会将该对象加入删除队列并异步删除它。过期日期和 Amazon S3 删除对象的日期之间可能会有一段延迟。对象过期后，不会再向您收取相关的存储时间费用。

要找出对象计划过期的时间，可使用 [HEAD Object](#) 或 [GET Object](#) API 操作。这些 API 操作将返回可提供此信息的响应头。

如果您创建的生命周期过期规则将促使存储在 STANDARD\_IA (或 ONEZONE\_IA) 存储中不到 30 天的对象过期，则您需要支付 30 天的费用。如果您创建的生命周期过期规则将促使存储在 GLACIER 存储中不到 90 天的对象过期，则您需要支付 90 天的费用。有关更多信息，请参阅 [Amazon S3 定价](#)。

## 生命周期和其他存储桶配置

除了生命周期配置之外，您还可以将其他配置与存储桶关联。本部分解释了生命周期配置如何与其他存储桶配置相关。

### 生命周期和版本控制

您可以向不受版本控制的存储桶和启用了版本控制的存储桶添加生命周期配置。有关更多信息，请参阅 [对象版本控制 \(p. 94\)](#)。

启用了版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。您可以为当前和非当前对象版本定义单独的生命周期规则。

有关更多信息，请参阅 [生命周期配置元素 \(p. 110\)](#)。有关版本控制的信息，请参阅 [对象版本控制 \(p. 94\)](#)。

### 启用了 MFA 的存储桶上的生命周期配置

启用了 MFA 的存储桶上不支持生命周期配置。

### 生命周期和日志记录

如果在您的存储桶上启用了日志记录，Amazon S3 将报告过期操作的结果，如下所示：

- 如果生命周期过期操作导致 Amazon S3 永久删除对象，Amazon S3 会在日志记录中将生命周期过期操作报告为 S3.EXPIRE.OBJECT 操作。
- 对于启用了版本控制的存储桶，如果生命周期过期操作导致当前版本 (其中 Amazon S3 添加了删除标记) 被逻辑删除，Amazon S3 会在日志记录中将逻辑删除报告为 S3.CREATE.DELETEMARKER 操作。有关更多信息，请参阅 [对象版本控制 \(p. 94\)](#)。
- 当 Amazon S3 将一个对象转换为 GLACIER 存储类时，它会在日志记录中将该转换报告为操作 S3.TRANSITION.OBJECT，以指示它发起了此操作。在将对象转换为 STANDARD\_IA (或 ONEZONE\_IA) 存储类时，会将它报告为 S3.TRANSITION\_SIA.OBJECT (或 S3.TRANSITION\_ZIA.OBJECT) 操作。

### 更多信息

- [生命周期配置元素 \(p. 110\)](#)
- [转换为 GLACIER 存储类 \(对象存档\) \(p. 107\)](#)
- [在存储桶上设置生命周期配置 \(p. 124\)](#)

## 生命周期配置元素

### 主题

- [ID 元素 \(p. 110\)](#)
- [Status 元素 \(p. 110\)](#)
- [Filter 元素 \(p. 110\)](#)
- [用于描述生命周期操作的元素 \(p. 112\)](#)

您可以将生命周期配置指定为 XML，该配置包含一个或多个生命周期规则。

```
<LifecycleConfiguration>
  <Rule>
    ...
  </Rule>
  <Rule>
    ...
  </Rule>
</LifecycleConfiguration>
```

每个规则由以下内容组成：

- 规则元数据，包含规则 ID 以及用于指示规则是已启用还是已禁用的状态。如果规则处于禁用状态，则 Amazon S3 不会执行规则中指定的任何操作。
- 筛选条件，用于标识规则将应用于的对象。您可以通过使用一个对象键前缀和/或一个或多个对象标签来指定筛选条件。
- 您希望 Amazon S3 执行指定操作时的一个或多个转换或过期操作，带有位于对象的生命周期内的日期或时间段。

以下部分介绍了生命周期配置中的 XML 元素。有关示例生命周期配置，请参阅[生命周期配置的示例 \(p. 115\)](#)。

### ID 元素

一个生命周期配置最多可以有 1000 个规则。`<ID>` 元素唯一地标识规则。ID 长度最多为 255 个字符。

### Status 元素

`<Status>` 元素值可以是 Enabled 或 Disabled。如果规则处于禁用状态，则 Amazon S3 不会执行规则中定义的任何操作。

### Filter 元素

生命周期规则可基于您在该规则中指定的 `<Filter>` 元素应用于存储桶中的所有对象或一部分对象。

您可以按键前缀、对象标签或二者的组合（在此情况下，Amazon S3 使用逻辑 AND 组合筛选条件）筛选对象。考虑以下示例：

- 使用键前缀指定筛选条件 - 此示例显示一个生命周期规则，此规则基于键名前缀（`logs/`）应用于一部分对象。例如，此生命周期规则应用于对象 `logs/mylog.txt`、`logs/temp1.txt` 和 `logs/test.txt`。此规则不应用于对象 `example.jpg`。

```
<LifecycleConfiguration>
  <Rule>
```

```
<Filter>
    <Prefix>logs/</Prefix>
</Filter>
transition/expiration actions.
...
</Rule>
...
</LifecycleConfiguration>
```

如果您要基于不同的键名称前缀将生命周期操作应用于一部分对象，可指定单独的规则。在每个规则中，指定基于前缀的筛选条件。例如，要描述具有键前缀 projectA/ 和 projectB/ 的对象的生命周期操作，可指定两个规则，如下所示：

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions.
    ...
    </Rule>

    <Rule>
        <Filter>
            <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions.
    ...
    </Rule>
</LifecycleConfiguration>
```

有关对象键的更多信息，请参阅[对象键 \(p. 87\)](#)。

- 指定基于对象标签的筛选条件 - 在以下示例中，此生命周期规则指定基于标签 (**key**) 和值 (**value**) 的筛选条件。随后，此规则仅应用于具有特定标签的一部分对象。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <Tag>
                <Key>key</Key>
                <Value>value</Value>
            </Tag>
        </Filter>
        transition/expiration actions.
    ...
    </Rule>
</LifecycleConfiguration>
```

您可以指定基于多个标签的筛选条件。您必须在 **<AND>** 元素中包含标签，如以下示例所示。此规则指示 Amazon S3 对具有两个标签 (带特定的标签键和值) 的对象执行生命周期操作。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Tag>
                    <Key>key1</Key>
                    <Value>value1</Value>
                </Tag>
                <Tag>
                    <Key>key2</Key>
```

```
<Value>value2</Value>
</Tag>
...
</And>
</Filter>
transition/expiration actions.
</Rule>
</Lifecycle>
```

此生命周期规则应用于具有指定的两个标签的对象。Amazon S3 执行逻辑 AND。请注意以下几点：

- 每个标签的键和值必须完全匹配。
- 此规则应用于具有规则中指定的一个或多个标签的一部分对象。如果为对象指定了其他标签，也没关系。

#### Note

当您在筛选条件中指定多个标签时，每个标签键必须是唯一的。

- 指定基于前缀和一个或多个标签的筛选条件 - 在生命周期规则中，您可以指定基于键前缀和一个或多个标签的筛选条件。同样，您也必须在 `<And>` 元素中包含所有这些内容，如下所示：

```
<LifecycleConfiguration>
  <Rule>
    <Filter>
      <And>
        <Prefix>key-prefix</Prefix>
        <Tag>
          <Key>key1</Key>
          <Value>value1</Value>
        </Tag>
        <Tag>
          <Key>key2</Key>
          <Value>value2</Value>
        </Tag>
        ...
      </And>
    </Filter>
    <Status>Enabled</Status>
    transition/expiration actions.
  </Rule>
</LifecycleConfiguration>
```

Amazon S3 使用逻辑 AND 组合这些筛选条件。即，此规则应用于具有特定键前缀和特定标签的一部分对象。一个筛选条件只能有一个前缀以及零个或多个标签。

- 您可以指定空筛选条件，在此情况下，此规则应用于存储桶中的所有对象。

```
<LifecycleConfiguration>
  <Rule>
    <Filter>
      </Filter>
    <Status>Enabled</Status>
    transition/expiration actions.
  </Rule>
</LifecycleConfiguration>
```

## 用于描述生命周期操作的元素

通过在生命周期规则中指定一个或多个以下预定义操作，您可以指示 Amazon S3 在对象的生命周期内执行特定操作。这些操作的效果取决于存储桶的版本控制状态。

- Transition 操作元素 - 您指定 Transition 操作以将对象从一个存储类别转换为另一个存储类别。有关转换对象的更多信息，请参阅[受支持的转换和相关限制 \(p. 105\)](#)。达到对象生命周期内指定的日期或时间段后，Amazon S3 将执行该转换。

对于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），Transition 操作适用于当前对象版本。为了管理非当前版本，Amazon S3 定义了 NoncurrentVersionTransition 操作（描述如下）。

- Expiration 操作元素 - Expiration 操作使规则中标识的对象过期，并且适用于任意 Amazon S3 存储类中符合条件的对象。有关存储类别的更多信息，请参阅[存储类别 \(p. 90\)](#)。Amazon S3 使所有过期的对象不可用。是否永久删除对象取决于存储桶的版本控制状态。

#### Important

对象过期生命周期策略不会删除未完成的分段上传。要删除未完成的分段上传，您必须使用 AbortIncompleteMultipartUpload 生命周期配置操作，如本节后面所述。

- 不受版本控制的存储桶 - Expiration 操作导致 Amazon S3 永久删除该对象。
- 受版本控制的存储桶 - 对于受版本控制的存储桶（即，已启用版本控制或已暂停版本控制），有一些指导 Amazon S3 如何处理 expiration 操作的注意事项。有关更多信息，请参阅[使用版本控制 \(p. 380\)](#)。无论版本控制状态如何，以下规则都适用：
  - Expiration 操作仅适用于当前版本（对非当前对象版本没有影响）。
  - 如果有一个或多个对象版本，并且删除标记是当前版本，则 Amazon S3 不会执行任何操作。
  - 如果当前对象版本是唯一的对象版本并且它还是删除标记（也称为过期对象删除标记，在这种情况下，所有对象版本都已删除，您只剩下一个删除标记），则 Amazon S3 将删除过期对象删除标记。您还可以使用过期操作来指示 Amazon S3 移除所有过期对象删除标记。有关示例，请参阅[示例 7：移除过期对象删除标记 \(p. 122\)](#)。

设置 Amazon S3 以管理过期时，还应该考虑以下各项：

- 已启用版本控制的存储桶

如果当前对象版本不是删除标记，Amazon S3 将添加具有唯一的版本 ID 的删除标记。这会使当前对象版本变为非当前版本，并使删除标记变为当前版本。

- 已暂停版本控制的存储桶

在已暂停版本控制的存储桶中，过期操作将使 Amazon S3 创建版本 ID 为 null 的删除标记。此删除标记会在版本层次结构中将任何对象版本替换为 null 版本 ID，从而实际上删除对象。

此外，Amazon S3 还提供可用于管理受版本控制的存储桶（即，启用了版本控制和暂停了版本控制的存储桶）中的非当前对象版本的以下操作。

- NoncurrentVersionTransition 操作元素 - 使用此操作可指定您希望在 Amazon S3 将对象转换为指定存储类别之前在当前存储类别中保留对象的时间（从对象变为非当前时开始）。有关转换对象的更多信息，请参阅[受支持的转换和相关限制 \(p. 105\)](#)。
- NoncurrentVersionExpiration 操作元素 - 使用此操作可指定您希望在 Amazon S3 永久删除非当前对象版本之前保留它们的时间（从对象变为非当前时开始）。删除的对象无法恢复。

当您需要更正任何意外删除或覆盖时，非当前对象的延时删除可能很有帮助。例如，您可以配置一个到期规则，以便在对象变为非当前版本五天后删除非当前版本。例如，假设在 2014 年 1 月 1 日上午 10:30 UTC，您创建了名为 photo.gif 的对象（版本 ID 111111）。在 2014 年 2 月 1 日上午 11:30 UTC，您意外删除了 photo.gif（版本 ID 111111），这将导致使用新版本 ID（如版本 ID 4857693）创建一个删除标记。您现在有五天时间可以在永久删除之前，恢复原始版本的 photo.gif（版本 ID 111111）。在 2014 年 1 月 8 日 00:00 UTC，过期生命周期规则执行并永久删除 photo.gif（版本 ID 111111）（在它成为非当前版本五天之后）。

### Important

对象过期生命周期策略不会删除未完成的分段上传。要删除未完成的分段上传，您必须使用 AbortIncompleteMultipartUpload 生命周期配置操作，如本节后面所述。

除了转换和过期操作之外，您还可以使用以下生命周期配置操作来指示 Amazon S3 中止未完成的分段上传。

- AbortIncompleteMultipartUpload 操作元素 – 使用此元素可设置您希望允许分段上传保持运行的最长时间（天）。如果适用的分段上传（由生命周期规则中指定的键名称 prefix 确定）未在预定义的时间段内成功完成，Amazon S3 将中止未完成的分段上传。有关更多信息，请参阅 [使用存储桶生命周期策略中止未完成的分段上传 \(p. 156\)](#)。

### Note

您无法在指定了基于对象标签的筛选条件的规则中指定此生命周期操作。

- ExpiredObjectDeleteMarker 操作元素 - 在启用了版本控制的存储桶中，包含零个非当前版本的删除标记称为“过期对象删除标记”。您可以使用此生命周期操作来指示 S3 删除过期对象删除标记。有关示例，请参阅 [示例 7：移除过期对象删除标记 \(p. 122\)](#)。

### Note

您无法在指定了基于对象标签的筛选条件的规则中指定此生命周期操作。

## Amazon S3 如何计算对象已成为非当前版本的时间长度

在启用版本控制的存储桶中，您可以有一个对象的多个版本，始终有一个当前版本和零个或零个以上非当前版本。每次上传对象时，当前版本都保留为非当前版本，新添加的版本（后继者）会成为当前版本。为了确定对象成为非当前版本的天数，Amazon S3 会查看其后继者的创建时间。Amazon S3 使用自后继者创建以来的天数作为对象成为非当前版本的天数。

### 在使用生命周期配置时还原对象的以前版本

按照主题 [还原早期版本 \(p. 396\)](#) 中的详细说明，您可以使用以下两种方法中的任一方法来检索对象的以前版本：

- 通过将对象的非当前版本复制到相同存储桶中。复制的对象将成为该对象的当前版本，且所有对象版本都保留。
- 通过永久删除当前版本的对象。当您删除当前对象版本时，实际上是将非当前版本转换为该对象的当前版本。

将生命周期配置规则用于启用版本控制的存储桶时，我们建议的最佳实践是使用第一种方法。

由于 Amazon S3 的最终一致性语义，在更改传播之前，永久删除的当前版本可能不会消失（Amazon S3 可能不知道此删除操作）。同时，您配置来使非当前对象过期的生命周期规则可能会永久删除非当前对象，包括您要还原的对象。因此，复制旧版本（按照第一种方法中的建议）是更安全的替代方法。

## 生命周期规则：基于对象的期限

您可以指定 Amazon S3 能够执行操作的时间段（自对象创建（或修改）以来的天数）。

当您在生命周期配置中的 Transition 和 Expiration 操作中指定天数时，请注意以下事项：

- 它是自对象创建以来发生操作的天数。
- Amazon S3 按以下方式计算时间：将在规则中指定的天数与对象创建时间相加，然后将得出的时间舍入至下一日的午夜 UTC。例如，如果对象的创建时间是 2014 年 1 月 15 日上午 10:30 UTC，并且您在转换规则中指定了 3 天，则对象的转换日期将计算为 2014 年 1 月 19 日 00:00 UTC。

### Note

Amazon S3 仅为每个对象保持上次修改日期。例如，Amazon S3 控制台在对象 Properties (属性) 窗格中显示 Last Modified (上次修改日期) 日期。最初创建新对象时，此日期反映对象的创建日期。如果您替换对象，此日期会相应地更改。因此，在我们使用术语创建日期时，它与上次修改日期术语是同义词。

当在生命周期配置中的 NoncurrentVersionTransition 和 NoncurrentVersionExpiration 操作中指定天数时，请注意以下几点：

- 它是从对象版本变为非当前版本 (即，自对象被覆盖或删除起) 以来的天数，作为 Amazon S3 将对指定对象执行操作的时间。
- Amazon S3 按以下方式计算时间：将规则中指定的天数与创建对象新后继者版本的时间相加，然后将得出的时间舍入至下一日的午夜 UTC。例如，在您的存储桶中，某个对象的当前版本的创建时间是 2014 年 1 月 1 日上午 10:30 UTC，如果替换当前版本的对象新后继者版本的创建时间是 2014 年 1 月 15 日上午 10:30 UTC，并且您在转换规则中指定了 3 天，则对象的转换日期计算为 2014 年 1 月 19 日 00:00 UTC。

## 生命周期规则：基于特定日期

当在生命周期规则中指定操作时，您可以指定希望 Amazon S3 执行此操作的日期。到达特定日期时，S3 会向所有合格对象应用该操作 (基于筛选条件)。

如果为生命周期操作指定一个过去的日期，所有合格对象会立即符合该生命周期操作的条件。

### Important

基于日期的操作并非一次性操作。即使过了该日期后，只要规则状态为“已启用”，S3 仍会继续应用该基于日期的操作。

例如，假设您指定一个基于日期的过期操作来删除所有对象 (假设规则中未指定任何筛选条件)。在指定日期，S3 会使存储桶中的所有对象过期。此外，S3 还会继续使你在存储桶中创建的所有新对象过期。要终止生命周期操作，您必须从生命周期配置中删除操作，禁用规则或从生命周期配置中删除规则。

此日期值必须符合 ISO 8601 格式。时间始终为午夜 UTC。

### Note

您无法使用 Amazon S3 控制台创建基于日期的生命周期规则，但是可以查看、禁用或删除这类规则。

## 生命周期配置的示例

此部分提供生命周期配置的示例。每个示例演示如何在各个示例方案中指定 XML。

### 主题

- [示例 1：指定筛选条件 \(p. 116\)](#)
- [示例 2：禁用生命周期规则 \(p. 117\)](#)
- [示例 3：在对象的生命周期内逐步将存储类降级 \(p. 118\)](#)
- [示例 4：指定多个规则 \(p. 118\)](#)
- [示例 5：重叠的筛选条件、冲突的生命周期操作以及 Amazon S3 的功能 \(p. 119\)](#)
- [示例 6：为启用了版本控制的存储桶指定生命周期规则 \(p. 122\)](#)
- [示例 7：移除过期对象删除标记 \(p. 122\)](#)

- [示例 8：用于中止分段上传的生命周期配置 \(p. 124\)](#)

## 示例 1：指定筛选条件

每个生命周期规则都包含一个筛选条件，该筛选条件可用于确定存储桶中适用生命周期规则的一部分对象。以下生命周期配置显示了如何指定筛选条件的示例。

- 在此生命周期配置规则中，筛选条件指定了一个键前缀 (tax/)。因此，此规则应用于带键名称前缀 tax/ 的对象，例如 tax/doc1.txt 和 tax/doc2.txt

此规则指定两个引导 Amazon S3 完成以下任务的操作：

- 在对象创建 365 天 (一年) 后将其转换为 GLACIER 存储类。
- 在对象创建 3650 天 (10 年) 后将其删除 (Expiration 操作)。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition and Expiration Rule</ID>
    <Filter>
      <Prefix>tax/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

您可以为每个操作指定一个日期，而不用以创建后的天数的形式指定对象期限。但是，您不能在同一规则中同时使用 Days 和 Date。

- 如果希望生命周期规则应用于存储桶中的所有对象，请指定一个空前缀。在以下配置中，该规则指定了一个 Transition 操作，该操作指示 Amazon S3 在对象创建 0 天后将其转换为 GLACIER 存储类，在这种情况下，对象都有资格在创建后的午夜 UTC 存档到 Amazon Glacier。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- 您可以在筛选条件中指定零或一个键名称前缀以及零或多个对象标签。以下示例代码将生命周期规则应用于带 tax/ 键前缀的一部分对象以及包含具有特定键和值的两个标签的对象。请注意，在指定多个筛选条件时，您必须按下面所示包含 AND (Amazon S3 会应用逻辑 AND 以将指定筛选条件组合起来)。

```
...
<Filter>
  <And>
    <Prefix>tax/</Prefix>
```

```
<Tag>
  <Key>key1</Key>
  <Value>value1</Value>
</Tag>
<Tag>
  <Key>key2</Key>
  <Value>value2</Value>
</Tag>
</And>
</Filter>
...
```

- 您可以仅基于标签筛选对象。例如，以下生命周期规则应用于具有两个指定标签的对象（该规则未指定任何前缀）：

```
...
<Filter>
<And>
<Tag>
  <Key>key1</Key>
  <Value>value1</Value>
</Tag>
<Tag>
  <Key>key2</Key>
  <Value>value2</Value>
</Tag>
</And>
</Filter>
...
```

### Important

当您在生命周期配置中有多个规则时，对象可能变得有资格执行多个生命周期操作。Amazon S3 在这些情况下遵循的一般规则是：

- 永久删除优先于转换。
- 转换优先于删除标记的创建。
- 当对象有资格进行 GLACIER 和 STANDARD\_IA (或 ONEZONE\_IA) 转换时，Amazon S3 将选择 GLACIER 转换。

有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作以及 Amazon S3 的功能 (p. 119)。

## 示例 2：禁用生命周期规则

您可以临时禁用生命周期规则。以下生命周期配置指定了两个规则：

- 规则 1 指示 Amazon S3 在带 logs/ 前缀的对象创建后立即将其转换为 GLACIER 存储类。
- 规则 2 指示 Amazon S3 在带 documents/ 前缀的对象创建后立即将其转换为 GLACIER 存储类。

在策略中，启用了规则 1 并禁用了规则 2。Amazon S3 不会对已禁用规则执行任何操作。

```
<LifecycleConfiguration>
<Rule>
  <ID>Rule1</ID>
  <Filter>
    <Prefix>logs/</Prefix>
```

```
</Filter>
<Status>Enabled</Status>
<Transition>
  <Days>0</Days>
  <StorageClass>GLACIER</StorageClass>
</Transition>
</Rule>
<Rule>
  <ID>Rule2</ID>
  <Prefix>documents/<Prefix>
  <Status>Disabled</Status>
  <Transition>
    <Days>0</Days>
    <StorageClass>GLACIER</StorageClass>
  </Transition>
</Rule>
</LifecycleConfiguration>
```

## 示例 3：在对象的生命周期内逐步将存储类降级

在本示例中，您需要利用生命周期配置在对象的生命周期内逐步将存储类降级。级别降低可帮助减少存储成本。有关定价的更多信息，请参阅 [Amazon S3 定价](#)。

以下生命周期配置指定了应用于带键名称前缀 logs/ 的对象的规则。该规则指定了以下操作：

- 两个转换操作：
  - 在对象创建 30 天后将其转换为 STANDARD\_IA 存储类。
  - 在对象创建 90 天后将其转换为 GLACIER 存储类。
- 一个过期操作，指示 Amazon S3 在对象创建一年后将其删除。

```
<LifecycleConfiguration>
<Rule>
  <ID>example-id</ID>
  <Filter>
    <Prefix>logs/<Prefix>
  </Filter>
  <Status>Enabled</Status>
  <Transition>
    <Days>30</Days>
    <StorageClass>STANDARD_IA</StorageClass>
  </Transition>
  <Transition>
    <Days>90</Days>
    <StorageClass>GLACIER</StorageClass>
  </Transition>
  <Expiration>
    <Days>365</Days>
  </Expiration>
</Rule>
</LifecycleConfiguration>
```

### Note

如果所有生命周期操作都应用于同一组对象（由筛选条件标识），您可以使用一个规则来描述这些操作。或者，您也可以添加多个规则，每个规则指定一个不同的筛选条件。

## 示例 4：指定多个规则

如果您希望不同的对象有不同的生命周期操作，则可以指定多个规则。以下生命周期配置有两个规则：

- 规则 1 应用于带键名称前缀 classA/ 的对象。它指示 Amazon S3 在对象创建一年后将其转换为 GLACIER 存储类，并在对象创建 10 年后使它们过期。
- 规则 2 应用于带键名称前缀 classB/ 的对象。它指示 Amazon S3 在对象创建 90 天后将其转换为 STANDARD\_IA 存储类，并在对象创建 1 年后将其删除。

```
<LifecycleConfiguration>
  <Rule>
    <ID>ClassADocRule</ID>
    <Filter>
      <Prefix>classA/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>ClassBDocRule</ID>
    <Filter>
      <Prefix>classB/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>90</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

## 示例 5：重叠的筛选条件、冲突的生命周期操作以及 Amazon S3 的功能

您可能会指定一个在其中指定了重叠的前缀或操作的生命周期配置。以下示例显示了 Amazon S3 如何选择解决潜在冲突。

### Example 1: 重叠的前缀 (无冲突)

以下示例配置包含两个规则，它们指定了如下所示的重叠前缀：

- 第一个规则指定了一个空筛选条件，指示存储桶中的所有对象。
- 第二个规则指定了一个键名称前缀 logs/，指示仅一部分对象。

规则 1 请求 Amazon S3 在对象创建一年后将其全部删除，规则 2 请求 Amazon S3 在对象创建 30 天后将其中一部分转换为 STANDARD\_IA 存储类。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule_1</ID>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
```

```
<Expiration>
  <Days>365</Days>
</Expiration>
</Rule>
<Rule>
  <ID>Rule 2</ID>
  <Filter>
    <Prefix>logs/</Prefix>
  </Filter>
  <Status>Enabled</Status>
  <Transition>
    <StorageClass>STANDARD_IA<StorageClass>
    <Days>30</Days>
  </Transition>
</Rule>
</LifecycleConfiguration>
```

### Example 2: 冲突的生命周期操作

本示例配置中有两个规则，它们指示 Amazon S3 在对象的生命周期中同时对同一组对象执行两个不同的操作：

- 两个规则指定了相同的键名称前缀，因此两个规则都应用于同一组对象。
- 当应用两个规则时，它们指定了相同的“对象创建后的 365 天”。
- 一个规则指示 Amazon S3 将对象转换为 STANDARD\_IA 存储类，另一个规则希望 Amazon S3 使对象同时过期。

```
<LifecycleConfiguration>
<Rule>
  <ID>Rule 1</ID>
  <Filter>
    <Prefix>logs/</Prefix>
  </Filter>
  <Status>Enabled</Status>
  <Expiration>
    <Days>365</Days>
  </Expiration>
</Rule>
<Rule>
  <ID>Rule 2</ID>
  <Filter>
    <Prefix>logs/</Prefix>
  </Filter>
  <Status>Enabled</Status>
  <Transition>
    <StorageClass>STANDARD_IA<StorageClass>
    <Days>365</Days>
  </Transition>
</Rule>
</LifecycleConfiguration>
```

在这种情况下，由于您希望对象过期（已删除），更改存储类没有意义，且 Amazon S3 只选择针对这些对象的过期操作。

### Example 3: 导致冲突的生命周期操作的重叠的前缀

在本示例中，配置包含两个指定重叠前缀的规则，如下所示：

- 规则 1 指定了一个空前缀（指示所有对象）。
- 规则 2 指定了一个键名称前缀（logs/），用于确定所有对象中的一部分。

对于带 logs/ 键名称前缀的一部分对象，两个规则中的生命周期操作都适用。一个规则指示 Amazon S3 在对象创建 10 天后对其进行转换，另一个规则指示 Amazon S3 在对象创建 365 天后对其进行转换。

```
<LifecycleConfiguration>
<Rule>
  <ID>Rule 1</ID>
  <Filter>
    <Prefix></Prefix>
  </Filter>
  <Status>Enabled</Status>
  <Transition>
    <StorageClass>STANDARD_IA<StorageClass>
    <Days>10</Days>
  </Transition>
</Rule>
<Rule>
  <ID>Rule 2</ID>
  <Filter>
    <Prefix>logs/</Prefix>
  </Filter>
  <Status>Enabled</Status>
  <Transition>
    <StorageClass>STANDARD_IA<StorageClass>
    <Days>365</Days>
  </Transition>
</Rule>
</LifecycleConfiguration>
```

在这种情况下，Amazon S3 将选择在对象创建 10 天后转换它们。

#### Example 4: 基于标签的筛选和随之出现的冲突的生命周期操作

假设您有包含两个规则 (每个规则各指定一个标签筛选条件) 的以下生命周期策略：

- 规则 1 指定了基于标签的筛选条件 (tag1/value1)。此规则指示 Amazon S3 在对象创建 365 天后将其转换为 GLACIER 存储类。
- 规则 2 指定了基于标签的筛选条件 (tag2/value2)。此规则指示 Amazon S3 在对象创建 14 天后使其过期。

生命周期配置如下所示：

```
<LifecycleConfiguration>
<Rule>
  <ID>Rule 1</ID>
  <Filter>
    <Tag>
      <Key>tag1</Key>
      <Value>value1</Value>
    </Tag>
  </Filter>
  <Status>Enabled</Status>
  <Transition>
    <StorageClass>GLACIER<StorageClass>
    <Days>365</Days>
  </Transition>
</Rule>
<Rule>
  <ID>Rule 2</ID>
  <Filter>
    <Tag>
      <Key>tag2</Key>
      <Value>value1</Value>
    </Tag>
  </Filter>
  <Status>Enabled</Status>
  <Transition>
    <StorageClass>DELETED<StorageClass>
    <Days>14</Days>
  </Transition>
</Rule>
</LifecycleConfiguration>
```

```
</Tag>
</Filter>
<Status>Enabled</Status>
<Expiration>
    <Days>14</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>
```

该策略很有用，但如果存在带有这两个标签的对象，则 S3 必须决定如何操作。也就是说，两个规则都将应用于对象，您实际上在指示 Amazon S3 执行冲突的操作。在这种情况下，Amazon S3 将在对象创建 14 天后使其过期。该对象将被删除，因此转换操作不会起作用。

## 示例 6：为启用了版本控制的存储桶指定生命周期规则

假设您有一个启用了版本控制的存储桶，这意味着对于每个对象，您都有一个当前版本以及零个或零个以上的非当前版本。您想要保留一年的历史记录，然后删除非当前版本。有关版本控制的更多信息，请参阅[对象版本控制 \(p. 94\)](#)。

您还希望在对象变为非当前版本 30 天后通过将非当前版本移动到 GLACIER 来节省存储成本（担任您不需要其实时访问权限的冷数据）。此外，您还期望在对象创建 90 天后降低当前版本的访问频率，因此您可能会选择将这些对象移动到 STANDARD\_IA 存储类。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
            <Prefix></Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
            <Days>90</Days>
            <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
        <NoncurrentVersionExpiration>
            <NoncurrentDays>365</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

## 示例 7：移除过期对象删除标记

已启用版本控制的存储桶的每个对象有一个当前版本，以及一个或多个非当前版本。在删除对象时，请注意以下几点：

- 如果您在删除请求中未指定版本 ID，Amazon S3 将添加一个删除标记，而不是删除对象。当前对象版本将变为非当前版本，然后删除标记将变为当前版本。
- 如果您在删除请求中指定了版本 ID，Amazon S3 将永久删除对象版本（不会创建删除标记）。
- 包含零个非当前版本的删除标记称为过期对象删除标记。

本示例演示了可在存储桶中创建过期的对象删除标记的场景，以及如何使用生命周期配置指示 Amazon S3 移除过期的对象删除标记。

假设您编写了一个生命周期策略，该策略指定 NoncurrentVersionExpiration 操作以在对象变为非当前版本 30 天之后删除这些非当前版本，如下所示：

```
<LifecycleConfiguration>
  <Rule>
    ...
    <NoncurrentVersionExpiration>
      <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

`NoncurrentVersionExpiration` 操作不会应用于当前对象版本，而只会删除非当前版本。

对于当前对象版本，您可以根据当前对象版本是否遵循了明确定义的生命周期，通过以下方式管理其生命周期：

- 当前对象版本遵循明确定义的生命周期。

在这种情况下，您可以将生命周期策略与 `Expiration` 操作一起使用以指示 Amazon S3 移除当前版本，如以下示例所示：

```
<LifecycleConfiguration>
  <Rule>
    ...
    <Expiration>
      <Days>60</Days>
    </Expiration>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

Amazon S3 将在每个当前对象版本创建 60 天之后通过为其添加删除标记来删除它们。这会使当前对象版本变为非当前版本，并使删除标记变为当前版本。有关更多信息，请参阅 [使用版本控制 \(p. 380\)](#)。

同一生命周期配置中的 `NoncurrentVersionExpiration` 操作将在对象变为非当前对象 30 天后删除它们。这样，所有对象版本都会删除，您将获得过期对象删除标记，但 Amazon S3 会为您检测并移除过期对象删除标记。

- 当前对象版本没有明确定义的生命周期。

在这种情况下，您可以在不需要对象时手动删除它们，从而创建具有一个或多个非当前版本的删除标记。如果使用 `NoncurrentVersionExpiration` 操作的生命周期配置删除了所有非当前版本，则您现在会获得过期对象删除标记。

特别针对这种情况，Amazon S3 生命周期配置提供了 `Expiration` 操作，让您可以请求 Amazon S3 移除过期对象删除标记：

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
    </Expiration>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
```

```
</LifecycleConfiguration>
```

通过在 `Expiration` 操作中将 `ExpiredObjectDeleteMarker` 元素设为 `true`，您可以指示 Amazon S3 移除过期对象删除标记。

**Note**

当指定 `ExpiredObjectDeleteMarker` 生命周期操作时，该规则无法指定基于标签的筛选条件。

## 示例 8：用于中止分段上传的生命周期配置

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[分段上传概述 \(p. 155\)](#)。

利用生命周期配置，当分段上传在指定的启动后的天数内未完成时，您可以指示 Amazon S3 中止未完成的分段上传（通过在规则中指定的键名称前缀确定）。当 Amazon S3 中止分段上传时，它将删除与分段上传关联的所有分段。这将确保您没有包含存储在 Amazon S3 中的分段的未完成分段上传，从而不必为这些分段支付任何存储费用。

**Note**

当指定 `AbortIncompleteMultipartUpload` 生命周期操作时，该规则无法指定基于标签的筛选条件。

下面是使用 `AbortIncompleteMultipartUpload` 操作指定规则的示例生命周期配置。此操作将请求 Amazon S3 在分段上传启动 7 天后中止未完成的分段上传。

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Filter>
      <Prefix>SomeKeyPrefix</Prefix>
    </Filter>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

## 在存储桶上设置生命周期配置

### 主题

- 使用 Amazon S3 控制台管理对象的生命周期 (p. 125)
- 使用 AWS CLI 设置生命周期配置 (p. 125)
- 使用 AWS SDK for Java 管理对象生命周期 (p. 127)
- 使用适用于 .NET 的 AWS 开发工具包管理对象的生命周期 (p. 129)
- 使用适用于 Ruby 的 AWS 开发工具包管理对象的生命周期 (p. 132)
- 使用 REST API 管理对象的生命周期 (p. 132)

本部分介绍如何使用 AWS 开发工具包、Amazon S3 控制台或 AWS CLI 以编程方式在存储桶上设置生命周期配置。请注意以下几点：

- 将生命周期配置添加到存储桶时，在将新的或更新的生命周期配置完全传播到所有 Amazon S3 系统前通常存在一些滞后。在生命周期配置完全生效之前，预计会有几分钟延迟。在删除生命周期配置时也可能出现这种延迟。

- 禁用或删除生命周期规则时，在短暂延迟后 Amazon S3 会停止安排新对象的删除或转换。已安排的任何对象都将被取消安排，并且不会被删除或转换。
- 当您向存储桶添加生命周期配置时，配置规则将应用到现有对象以及您在以后添加的对象。例如，如果您在今天添加带有过期操作的生命周期配置规则，而该规则使带有特定前缀的对象在创建 30 天后过期，Amazon S3 会将任何超过 30 天的现有对象加入删除队列。
- 当满足生命周期配置规则时，可能经过一些滞后才会触发相应的操作。但一满足生命周期配置规则，账单就会立即发生变化，无论是否执行操作。例如，在对象到期后，即使没有立即删除，也不会向您收取存储费。另一个示例是对象转换时间一过就会立即按照 Amazon Glacier 存储费率向您收费，即使没有立即将该对象转换到 Amazon Glacier 也是如此。

有关生命周期配置的信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

## 使用 Amazon S3 控制台管理对象的生命周期

您可以使用 Amazon S3 控制台为某个存储桶指定生命周期规则。

有关如何使用 AWS 管理控制台设置生命周期规则的说明，请参阅[如何为 S3 存储桶创建生命周期策略？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

## 使用 AWS CLI 设置生命周期配置

您可以使用以下 AWS CLI 命令管理生命周期配置：

- put-bucket-lifecycle-configuration
- get-bucket-lifecycle-configuration
- delete-bucket-lifecycle

有关设置 AWS CLI 的说明，请参阅[设置 AWS CLI \(p. 520\)](#)。

请注意，Amazon S3 生命周期配置是一个 XML 文件。但在使用 CLI 时，您无法指定 XML，而是必须指定 JSON。以下是您可在 AWS CLI 命令中指定的示例 XML 生命周期配置和等效 JSON：

- 请考虑以下示例生命周期配置：

```
<LifecycleConfiguration>
  <Rule>
    <ID>ExampleRule</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

将显示等效的 JSON：

```
{
  "Rules": [
    {
      "Filter": {
```

```
        "Prefix": "documents/"
    },
    "Status": "Enabled",
    "Transitions": [
        {
            "Days": 365,
            "StorageClass": "GLACIER"
        }
    ],
    "Expiration": {
        "Days": 3650
    },
    "ID": "ExampleRule"
}
]
```

- 请考虑以下示例生命周期配置：

```
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
        <ID>id-1</ID>
        <Expiration>
            <Days>1</Days>
        </Expiration>
        <Filter>
            <And>
                <Prefix>myprefix</Prefix>
                <Tag>
                    <Key>mytagkey1</Key>
                    <Value>mytagvalue1</Value>
                </Tag>
                <Tag>
                    <Key>mytagkey2</Key>
                    <Value>mytagvalue2</Value>
                </Tag>
            </And>
        </Filter>
        <Status>Enabled</Status>
    </Rule>
</LifecycleConfiguration>
```

将显示等效的 JSON：

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": [
                    {
                        "Prefix": "myprefix",
                        "Tags": [
                            {
                                "Value": "mytagvalue1",
                                "Key": "mytagkey1"
                            },
                            {
                                "Value": "mytagvalue2",
                                "Key": "mytagkey2"
                            }
                        ]
                    }
                ],
                "Status": "Enabled",
            }
        }
    ]
}
```

```
        "Expiration": {
            "Days": 1
        }
    ]
}
```

您可以测试 `put-bucket-lifecycle-configuration`，如下所示：

1. 将 JSON 生命周期配置保存在一个文件 (`lifecycle.json`) 中。
2. 运行以下 AWS CLI 命令以在存储桶上设置生命周期配置：

```
$ aws s3api put-bucket-lifecycle-configuration \
--bucket bucketname \
--lifecycle-configuration file://lifecycle.json
```

3. 要进行验证，可使用 `get-bucket-lifecycle-configuration` AWS CLI 命令检索生命周期配置，如下所示：

```
$ aws s3api get-bucket-lifecycle-configuration \
--bucket bucketname
```

4. 要删除生命周期配置，可使用 `delete-bucket-lifecycle` AWS CLI 命令，如下所示：

```
aws s3api delete-bucket-lifecycle \
--bucket bucketname
```

## 使用AWS SDK for Java管理对象生命周期

您可以使用AWS SDK for Java管理存储桶的生命周期配置。有关管理生命周期配置的更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

### Note

在将生命周期配置添加到存储桶时，Amazon S3 会替换存储桶的当前生命周期配置（如果有）。要更新一个配置，请检索它，进行所需的更改，然后向存储桶添加已修订的生命周期配置。

### Example

以下示例说明如何使用AWS SDK for Java添加、更新和删除存储桶的生命周期配置。本示例执行以下操作：

- 向存储桶添加生命周期配置。
- 检索生命周期配置并通过添加其他规则来更新该配置。
- 向存储桶添加已修改的生命周期配置。Amazon S3 将替换现有配置。
- 再次检索配置并通过输出规则数来验证它是否具有正确数量的规则。
- 删除生命周期配置并通过再次尝试检索它来验证它是否已被删除。

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;
import java.util.Arrays;

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

```
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration.Transition;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.Tag;
import com.amazonaws.services.s3.model.lifecycle.LifecycleAndOperator;
import com.amazonaws.services.s3.model.lifecycle.LifecycleFilter;
import com.amazonaws.services.s3.model.lifecycle.LifecyclePrefixPredicate;
import com.amazonaws.services.s3.model.lifecycle.LifecycleTagPredicate;

public class LifecycleConfiguration {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";

        // Create a rule to archive objects with the "glacierobjects/" prefix to Glacier
        // immediately.
        BucketLifecycleConfiguration.Rule rule1 = new BucketLifecycleConfiguration.Rule()
            .withId("Archive immediately rule")
            .withFilter(new LifecycleFilter(new
                LifecyclePrefixPredicate("glacierobjects/")))
            .addTransition(new
                Transition().withDays(0).withStorageClass(StorageClass.Glacier))
            .withStatus(BucketLifecycleConfiguration.ENABLED);

        // Create a rule to transition objects to the Standard-Infrequent Access storage
        // class
        // after 30 days, then to Glacier after 365 days. Amazon S3 will delete the objects
        // after 3650 days.
        // The rule applies to all objects with the tag "archive" set to "true".
        BucketLifecycleConfiguration.Rule rule2 = new BucketLifecycleConfiguration.Rule()
            .withId("Archive and then delete rule")
            .withFilter(new LifecycleFilter(new LifecycleTagPredicate(new
                Tag("archive", "true"))))
            .addTransition(new
                Transition().withDays(30).withStorageClass(StorageClass.StandardInfrequentAccess))
            .addTransition(new
                Transition().withDays(365).withStorageClass(StorageClass.Glacier))
            .withExpirationInDays(3650)
            .withStatus(BucketLifecycleConfiguration.ENABLED);

        // Add the rules to a new BucketLifecycleConfiguration.
        BucketLifecycleConfiguration configuration = new BucketLifecycleConfiguration()
            .withRules(Arrays.asList(rule1, rule2));

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Save the configuration.
            s3Client.setBucketLifecycleConfiguration(bucketName, configuration);

            // Retrieve the configuration.
            configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

            // Add a new rule with both a prefix predicate and a tag predicate.
            configuration.getRules().add(new
                BucketLifecycleConfiguration.Rule().withId("NewRule")
                    .withFilter(new LifecycleFilter(new LifecycleAndOperator(
                        Arrays.asList(new LifecyclePrefixPredicate("YearlyDocuments/"),
                            new LifecycleTagPredicate(new Tag("expire_after",
                                "ten_years"))))))
                    .withExpirationInDays(3650)

```

```
.withStatus(BucketLifecycleConfiguration.ENABLED));

// Save the configuration.
s3Client.setBucketLifecycleConfiguration(bucketName, configuration);

// Retrieve the configuration.
configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

// Verify that the configuration now has three rules.
configuration = s3Client.getBucketLifecycleConfiguration(bucketName);
System.out.println("Expected # of rules = 3; found: " +
configuration.getRules().size());

// Delete the configuration.
s3Client.deleteBucketLifecycleConfiguration(bucketName);

// Verify that the configuration has been deleted by attempting to retrieve it.
configuration = s3Client.getBucketLifecycleConfiguration(bucketName);
String s = (configuration == null) ? "No configuration found." : "Configuration
found.";
System.out.println(s);
}
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();
}
}
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包管理对象的生命周期

您可以使用适用于 .NET 的 AWS 开发工具包管理存储桶上的生命周期配置。有关管理生命周期配置的更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。

### Note

在添加生命周期配置时，Amazon S3 将替换指定存储桶上的现有生命周期配置。要更新配置，您必须先检索生命周期配置，进行更改，然后向存储桶添加已修订的生命周期配置。

### Example .NET 代码示例

以下示例说明如何使用适用于 .NET 的 AWS 开发工具包添加、更新和删除存储桶的生命周期配置。该代码示例执行以下操作：

- 向存储桶添加生命周期配置。
- 检索生命周期配置并通过添加其他规则来更新该配置。
- 向存储桶添加已修改的生命周期配置。Amazon S3 将替换现有生命周期配置。
- 再次检索配置并通过输出配置中的规则数进行验证。
- 删除生命周期配置并确认删除操作

有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

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

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        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()
        {
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
                {
                    Rules = new List<LifecycleRule>
                    {
                        new LifecycleRule
                        {
                            Id = "Archive immediately rule",
                            Filter = new LifecycleFilter()
                            {
                                LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                            },
                            Status = LifecycleRuleStatus.Enabled,
                            Transitions = new List<LifecycleTransition>
                            {
                                new LifecycleTransition
                                {
                                    Days = 0,
                                    StorageClass = S3StorageClass.Glacier
                                }
                            },
                            new LifecycleRule
                            {
                                Id = "Archive and then delete rule",
                                Filter = new LifecycleFilter()
                                {
                                    LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                },
                                Prefix = "projectdocs/"
                            },
                            Status = LifecycleRuleStatus.Enabled,
                            Transitions = new List<LifecycleTransition>
                            {
                                new LifecycleTransition
                                {

```

```
        Days = 30,
        StorageClass =
S3StorageClass.StandardInfrequentAccess
    },
    new LifecycleTransition
    {
        Days = 365,
        StorageClass = S3StorageClass.Glacier
    }
},
Expiration = new LifecycleRuleExpiration()
{
    Days = 3650
}
}
};

// Add the configuration to the bucket.
await AddExampleLifecycleConfigAsync(client, lifeCycleConfiguration);

// Retrieve an existing configuration.
lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

// Add a new rule.
lifeCycleConfiguration.Rules.Add(new LifecycleRule
{
    Id = "NewRule",
    Filter = new LifecycleFilter()
    {
        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
        {
            Prefix = "YearlyDocuments/"
        }
    },
    Expiration = new LifecycleRuleExpiration()
    {
        Days = 3650
    }
});

// Add the configuration to the bucket.
await AddExampleLifecycleConfigAsync(client, lifeCycleConfiguration);

// Verify that there are now three rules.
lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
Console.WriteLine("Expected # of rules=3; found:{0}",
lifeCycleConfiguration.Rules.Count);

// Delete the configuration.
await RemoveLifecycleConfigAsync(client);

// Retrieve a nonexistent configuration.
lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

}

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);
}
```

```
}

static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
{
    PutLifecycleConfigurationRequest request = new PutLifecycleConfigurationRequest
    {
        BucketName = bucketName,
        Configuration = configuration
    };
    var response = await client.PutLifecycleConfigurationAsync(request);
}

static async Task<LifecycleConfiguration> RetrieveLifecycleConfigAsync(IAmazonS3
client)
{
    GetLifecycleConfigurationRequest request = new GetLifecycleConfigurationRequest
    {
        BucketName = bucketName
    };
    var response = await client.GetLifecycleConfigurationAsync(request);
    var configuration = response.Configuration;
    return configuration;
}

static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
{
    DeleteLifecycleConfigurationRequest request = new
DeleteLifecycleConfigurationRequest
    {
        BucketName = bucketName
    };
    await client.DeleteLifecycleConfigurationAsync(request);
}
}
```

## 使用适用于 Ruby 的 AWS 开发工具包管理对象的生命周期

通过使用类 [AWS::S3::BucketLifecycleConfiguration](#)，您可以使用适用于 Ruby 的 AWS 开发工具包来管理存储桶上的生命周期配置。有关将适用于 Ruby 的 AWS 开发工具包和 Amazon S3 结合使用的更多信息，请参阅[使用适用于 Ruby 的 AWS 开发工具包 - 版本 3 \(p. 524\)](#)。有关管理生命周期配置的更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

## 使用 REST API 管理对象的生命周期

您可以使用 AWS 管理控制台为存储桶设置生命周期配置。如果您的应用程序需要它，您也可以直接发送 REST 请求。Amazon Simple Storage Service API Reference 中的以下各部分介绍与生命周期配置相关的 REST API 操作。

- [PUT Bucket lifecycle](#)
- [GET Bucket lifecycle](#)
- [DELETE Bucket lifecycle](#)

# 跨源资源共享 (CORS)

跨源资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。利用 CORS 支持，您可以使用 Amazon S3 构建丰富的客户端 Web 应用程序，同时让您可以选择性地允许跨源访问您的 Amazon S3 资源。

本部分提供 CORS 概述。副主题介绍如何通过使用 Amazon S3 控制台或通过以编程方式使用 Amazon S3 REST API 和 AWS 开发工具包来启用 CORS。

## 主题

- [跨源资源共享：使用案例场景 \(p. 133\)](#)
- [如何在我的存储桶上配置 CORS？\(p. 133\)](#)
- [Amazon S3 如何评估针对存储桶的 CORS 配置？\(p. 135\)](#)
- [允许跨源资源共享 \(CORS\) \(p. 135\)](#)
- [排查 CORS 问题 \(p. 141\)](#)

## 跨源资源共享：使用案例场景

以下是有关使用 CORS 的示例场景：

- **场景 1：**假设您在名为 website 的 Amazon S3 存储桶中托管网站（如[在 Amazon S3 上托管静态网站 \(p. 401\)](#)中所述）。您的用户加载了网站终端节点 `http://website.s3-website-us-east-1.amazonaws.com`。现在，您想要使用此存储桶中存储的网页上的 JavaScript，以使用该存储桶的 Amazon S3 API 终端节点 `website.s3.amazonaws.com` 向同一存储桶发出经身份验证的 GET 和 PUT 请求。浏览器通常会阻止 JavaScript 允许这些请求，但借助 CORS，您可以配置您的存储桶以显式支持来自 `website.s3-website-us-east-1.amazonaws.com` 的跨源请求。
- **场景 2：**假设您想要托管来自您的 S3 存储桶的 Web 字体。浏览器会再次要求对正在加载的 Web 字体进行 CORS 检查（也称为预检）。您可以配置托管 Web 字体的存储桶，以允许任何源发出这些请求。

## 如何在我的存储桶上配置 CORS？

要将您的存储桶配置为允许跨源请求，您可以创建一个 CORS 配置，该配置是一个 XML 文档，其中包含一些规则，它们能够识别您允许访问存储桶的源、每个源支持的操作（HTTP 方法）以及其他特定于操作的信息。

您可以向配置添加最多 100 条规则。通过编程方式或者使用控制台将 XML 文档作为 cors 子资源添加到存储桶 Amazon S3。有关更多信息，请参阅[允许跨源资源共享 \(CORS\) \(p. 135\)](#)。

您可以使用自己的域（例如 `example1.com`）提供内容，而不是通过使用 Amazon S3 网站终端节点来访问网站。有关如何使用您自己的域的信息，请参阅[示例：使用自定义域设置静态网站 \(p. 415\)](#)。以下示例 cors 配置具有三个规则，这些规则被指定为 CORSRule 元素：

- 第一个规则允许来自 `http://www.example1.com` 源的跨源 PUT、POST 和 DELETE 请求。该规则还通过 `Access-Control-Request-Headers` 标头允许预检 OPTIONS 请求中的所有标头。作为对预检 OPTIONS 请求的响应，Amazon S3 将返回请求的标头。
- 第二个规则允许与第一个规则具有相同的跨源请求，但第二个规则应用于另一个源 `http://www.example2.com`。
- 第三个规则允许来自所有源的跨源 GET 请求。\* 通配符将引用所有源。

```
<CORSConfiguration>
```

```
<CORSRule>
    <AllowedOrigin>http://www.example1.com</AllowedOrigin>

    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
</CORSRule>
<CORSRule>
    <AllowedOrigin>http://www.example2.com</AllowedOrigin>

    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
</CORSRule>
<CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
</CORSRule>
</CORSConfiguration>
```

CORS 配置还允许可选的配置参数，如下面的 CORS 配置所示。在本示例中，CORS 配置允许来自 <http://www.example.com> 源的跨源 PUT、POST 和 DELETE 请求。

```
<CORSConfiguration>
    <CORSRule>
        <AllowedOrigin>http://www.example.com</AllowedOrigin>
        <AllowedMethod>PUT</AllowedMethod>
        <AllowedMethod>POST</AllowedMethod>
        <AllowedMethod>DELETE</AllowedMethod>
        <AllowedHeader>*</AllowedHeader>
        <MaxAgeSeconds>3000</MaxAgeSeconds>
        <ExposeHeader>x-amz-server-side-encryption</
        ExposeHeader>
        <ExposeHeader>x-amz-request-id</
        ExposeHeader>
        <ExposeHeader>x-amz-id-2</ExposeHeader>
    </CORSRule>
</CORSConfiguration>
```

上述配置中的 `CORSRule` 元素包括以下可选元素：

- `MaxAgeSeconds` - 指定在 Amazon S3 针对特定资源的预检 OPTIONS 请求做出响应后，浏览器缓存该响应的时间（以秒为单位，在本示例中为 3000 秒）。通过缓存响应，在需要重复原始请求时，浏览器无需向 Amazon S3 发送预检请求。
- `ExposeHeader` - 识别可让客户从应用程序（例如，从 JavaScript XMLHttpRequest 对象）进行访问的响应标头（在本示例中，为 `x-amz-server-side-encryption`、`x-amz-request-id` 和 `x-amz-id-2`）。

## AllowedMethod 元素

在 CORS 配置中，您可以为 `AllowedMethod` 元素指定以下值。

- GET
- PUT
- POST

- DELETE
- HEAD

## AllowedOrigin 元素

在 AllowedOrigin 元素中，可指定您希望允许从中发送跨源请求的源，例如 `http://www.example.com`。源字符串只能包含至少一个 \* 通配符，例如 `http://*.example.com`。您可以选择将 \* 指定为源，以允许所有源发送跨源请求。您还可以指定 https 只允许安全的源。

## AllowedHeader 元素

AllowedHeader 元素通过 Access-Control-Request-Headers 标头指定预检请求中允许哪些标头。Access-Control-Request-Headers 标头中的每个标头名称必须匹配规则中的相应条目。Amazon S3 将仅发送请求的响应中允许的标头。有关可以在发送至 Amazon S3 的请求中使用的标头示例列表，请参阅 Amazon Simple Storage Service API Reference 指南中的[常用请求标头](#)。

规则中的每个 AllowedHeader 字符串可以包含至少一个 \* 通配符字符。例如，`<AllowedHeader>x-amz-*</AllowedHeader>` 将允许所有特定于 Amazon 的标头。

## ExposeHeader 元素

每个 ExposeHeader 元素标识您希望客户能够从其应用程序（例如，从 JavaScript XMLHttpRequest 对象）进行访问的响应标头。有关常用 Amazon S3 响应标头的列表，请参阅 Amazon Simple Storage Service API Reference 指南中的[常用响应标头](#)。

## MaxAgeSeconds 元素

MaxAgeSeconds 元素指定在预检请求被资源、HTTP 方法和源识别之后，浏览器将为预检请求缓存响应的时间（以秒为单位）。

## Amazon S3 如何评估针对存储桶的 CORS 配置？

Amazon S3 收到来自浏览器的预检请求后，它将为存储桶评估 CORS 配置，并使用第一个匹配传入浏览器请求的 CORSRule 规则来实现跨源请求。要使规则实现匹配，必须满足以下条件：

- 请求的 Origin 标头必须匹配 AllowedOrigin 元素。
- 预检 OPTIONS 请求中的请求方法（例如，GET 或 PUT）或 Access-Control-Request-Method 标头必须是某个 AllowedMethod 元素。
- 在预检请求中，请求的 Access-Control-Request-Headers 标头中列出的每个标头必须匹配 AllowedHeader 元素。

### Note

使存储桶允许 CORS 后，ACL 和策略仍适用。

## 允许跨源资源共享 (CORS)

通过使用 AWS 管理控制台、REST API、或 AWS 开发工具包对您的存储桶设置 CORS 配置，来实现跨源资源共享。

### 主题

- 使用 AWS 管理控制台实现跨源资源共享 (CORS) (p. 136)
- 使用 AWS SDK for Java 实现跨源资源共享 (CORS) (p. 136)
- 使用适用于 .NET 的 AWS 开发工具包实现跨源资源共享 (CORS) (p. 138)
- 使用 REST API 允许跨源资源共享 (CORS) (p. 141)

## 使用 AWS 管理控制台实现跨源资源共享 (CORS)

您可以使用 AWS 管理控制台对存储桶设置 CORS 配置。有关说明，请参阅[如何通过 CORS 允许跨域资源共享？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

## 使用 AWS SDK for Java 实现跨源资源共享 (CORS)

您可以使用 AWS SDK for Java 管理存储桶的跨源资源共享 (CORS)。有关 CORS 的更多信息，请参阅[跨源资源共享 \(CORS\) \(p. 133\)](#)。

### Example

以下示例：

- 创建 CORS 配置并对存储桶设置该配置
- 通过添加规则来检索并修改配置
- 向存储桶添加修改过的配置
- 删除配置

有关如何创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketCrossOriginConfiguration;
import com.amazonaws.services.s3.model.CORSRule;

public class CORS {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";

        // Create two CORS rules.
        List<CORSRule.AllowedMethods> rule1AM = new ArrayList<CORSRule.AllowedMethods>();
        rule1AM.add(CORSRule.AllowedMethods.PUT);
        rule1AM.add(CORSRule.AllowedMethods.POST);
        rule1AM.add(CORSRule.AllowedMethods.DELETE);
        CORSRule rule1 = new CORSRule().withId("CORSRule1").withAllowedMethods(rule1AM)
            .withAllowedOrigins(Arrays.asList(new String[] { "http://
*.example.com" }));

        List<CORSRule.AllowedMethods> rule2AM = new ArrayList<CORSRule.AllowedMethods>();
        rule2AM.add(CORSRule.AllowedMethods.GET);
        CORSRule rule2 = new CORSRule().withId("CORSRule2").withAllowedMethods(rule2AM)
    }
}
```

```
        .withAllowedOrigins(Arrays.asList(new String[]
{ "*" })).withMaxAgeSeconds(3000)
            .withExposedHeaders(Arrays.asList(new String[] { "x-amz-server-side-
encryption" })));

    List<CORSRule> rules = new ArrayList<CORSRule>();
    rules.add(rule1);
    rules.add(rule2);

    // Add the rules to a new CORS configuration.
    BucketCrossOriginConfiguration configuration = new
BucketCrossOriginConfiguration();
    configuration.setRules(rules);

    try {
        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
            .withCredentials(new ProfileCredentialsProvider())
            .withRegion(clientRegion)
            .build();

        // Add the configuration to the bucket.
        s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

        // Retrieve and display the configuration.
        configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
        printCORSConfiguration(configuration);

        // Add another new rule.
        List<CORSRule.AllowedMethods> rule3AM = new
ArrayList<CORSRule.AllowedMethods>();
        rule3AM.add(CORSRule.AllowedMethods.HEAD);
        CORSRule rule3 = new CORSRule().withId("CORSRule3").withAllowedMethods(rule3AM)
            .withAllowedOrigins(Arrays.asList(new String[] { "http://
www.example.com" }));

        rules = configuration.getRules();
        rules.add(rule3);
        configuration.setRules(rules);
        s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

        // Verify that the new rule was added by checking the number of rules in the
configuration.
        configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
        System.out.println("Expected # of rules = 3, found " +
configuration.getRules().size());

        // Delete the configuration.
        s3Client.deleteBucketCrossOriginConfiguration(bucketName);
        System.out.println("Removed CORS configuration.");

        // Retrieve and display the configuration to verify that it was
        // successfully deleted.
        configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
        printCORSConfiguration(configuration);
    }
    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();
    }
}
```

```
private static void printCORSConfiguration(BucketCrossOriginConfiguration configuration) {
    if (configuration == null) {
        System.out.println("Configuration is null.");
    } else {
        System.out.println("Configuration has " + configuration.getRules().size() + " rules\n");

        for (CORSRule rule : configuration.getRules()) {
            System.out.println("Rule ID: " + rule.getId());
            System.out.println("MaxAgeSeconds: " + rule.getMaxAgeSeconds());
            System.out.println("AllowedMethod: " + rule.getAllowedMethods());
            System.out.println("AllowedOrigins: " + rule.getAllowedOrigins());
            System.out.println("AllowedHeaders: " + rule.getAllowedHeaders());
            System.out.println("ExposeHeader: " + rule.getExposedHeaders());
            System.out.println();
        }
    }
}
```

## 使用适用于 .NET 的 AWS 开发工具包实现跨源资源共享 (CORS)

要管理存储桶的跨源资源共享 (CORS)，您可以使用适用于 .NET 的 AWS 开发工具包。有关 CORS 的更多信息，请参阅 [跨源资源共享 \(CORS\) \(p. 133\)](#)。

### Example

以下 C# 代码：

- 创建 CORS 配置并对存储桶设置该配置
- 通过添加规则来检索并修改配置
- 向存储桶添加修改过的配置
- 删除配置

有关创建和测试有效示例的信息，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CORSTest
    {
        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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CORSConfigTestAsync().Wait();
        }
    }
}
```

```
        }
    private static async Task CORSConfigTestAsync()
    {
        try
        {
            // Create a new configuration request and add two rules
            CORSConfiguration configuration = new CORSConfiguration
            {
                Rules = new System.Collections.Generic.List<CORSRule>
                {
                    new CORSRule
                    {
                        Id = "CORSRule1",
                        AllowedMethods = new List<string> {"PUT", "POST", "DELETE"},
                        AllowedOrigins = new List<string> {"http://*.example.com"}
                    },
                    new CORSRule
                    {
                        Id = "CORSRule2",
                        AllowedMethods = new List<string> {"GET"},
                        AllowedOrigins = new List<string> {"*"},
                        MaxAgeSeconds = 3000,
                        ExposeHeaders = new List<string> {"x-amz-server-side-
encryption"}
                    }
                };
            }

            // Add the configuration to the bucket.
            await PutCORSConfigurationAsync(configuration);

            // Retrieve an existing configuration.
            configuration = await RetrieveCORSConfigurationAsync();

            // Add a new rule.
            configuration.Rules.Add(new CORSRule
            {
                Id = "CORSRule3",
                AllowedMethods = new List<string> { "HEAD" },
                AllowedOrigins = new List<string> { "http://www.example.com" }
            });

            // Add the configuration to the bucket.
            await PutCORSConfigurationAsync(configuration);

            // Verify that there are now three rules.
            configuration = await RetrieveCORSConfigurationAsync();
            Console.WriteLine();
            Console.WriteLine("Expected # of rules=3; found:{0}",
                configuration.Rules.Count);
            Console.WriteLine();
            Console.WriteLine("Pause before configuration delete. To continue, click
Enter...");  

            Console.ReadKey();

            // Delete the configuration.
            await DeleteCORSConfigurationAsync();

            // Retrieve a nonexistent configuration.
            configuration = await RetrieveCORSConfigurationAsync();
        }
        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);
        }
    }

    static async Task PutCORSConfigurationAsync(CORSConfiguration configuration)
    {

        PutCORSConfigurationRequest request = new PutCORSConfigurationRequest
        {
            BucketName = bucketName,
            Configuration = configuration
        };

        var response = await s3Client.PutCORSConfigurationAsync(request);
    }

    static async Task<CORSConfiguration> RetrieveCORSConfigurationAsync()
    {
        GetCORSConfigurationRequest request = new GetCORSConfigurationRequest
        {
            BucketName = bucketName
        };
        var response = await s3Client.GetCORSConfigurationAsync(request);
        var configuration = response.Configuration;
        PrintCORSRules(configuration);
        return configuration;
    }

    static async Task DeleteCORSConfigurationAsync()
    {
        DeleteCORSConfigurationRequest request = new DeleteCORSConfigurationRequest
        {
            BucketName = bucketName
        };
        await s3Client.DeleteCORSConfigurationAsync(request);
    }

    static void PrintCORSRules(CORSConfiguration configuration)
    {
        Console.WriteLine();

        if (configuration == null)
        {
            Console.WriteLine("\nConfiguration is null");
            return;
        }

        Console.WriteLine("Configuration has {0} rules:", configuration.Rules.Count);
        foreach (CORSRule rule in configuration.Rules)
        {
            Console.WriteLine("Rule ID: {0}", rule.Id);
            Console.WriteLine("MaxAgeSeconds: {0}", rule.MaxAgeSeconds);
            Console.WriteLine("AllowedMethod: {0}", string.Join(", ",
rule.AllowedMethods.ToArray()));
            Console.WriteLine("AllowedOrigins: {0}", string.Join(", ",
rule.AllowedOrigins.ToArray()));
            Console.WriteLine("AllowedHeaders: {0}", string.Join(", ",
rule.AllowedHeaders.ToArray()));
            Console.WriteLine("ExposeHeader: {0}", string.Join(", ",
rule.ExposeHeaders.ToArray()));
        }
    }
}
```

```
}
```

## 使用 REST API 允许跨源资源共享 (CORS)

要对存储桶设置 CORS 配置，您可以使用 AWS 管理控制台。如果您的应用程序需要它，您也可以直接发送 REST 请求。Amazon Simple Storage Service API Reference 中的以下各部分介绍与 CORS 配置相关的 REST API 操作：

- [PUT Bucket cors](#)
- [GET Bucket cors](#)
- [DELETE Bucket cors](#)
- [OPTIONS object](#)

## 排查 CORS 问题

如果您在访问设置有 CORS 配置的存储桶时遇到意外行为，请尝试执行以下步骤来排查问题：

1. 验证对存储桶设置的 CORS 配置。

有关说明，请参阅 Amazon Simple Storage Service 控制台用户指南 中的[编辑存储桶权限](#)。如果设置了 CORS 配置，则控制台在 Properties (属性) 存储桶的 Permissions (权限) 部分中显示 Edit CORS Configuration (编辑 CORS 配置) 链接。

2. 使用您选择的工具捕获完整请求和响应。对于 Amazon S3 收到的每个请求，必须存在一个与请求中的数据匹配的 CORS 规则，如下所示：

- a. 验证请求是否具有源标头。

如果缺少该标头，则 Amazon S3 不会将请求视为跨源请求，并且不会在响应中发送 CORS 响应标头。

- b. 验证请求中的源标头是否与指定 AllowedOrigin 中的至少一个 CORSRule 元素匹配。

源请求标头中的方案、主机和端口值必须与 AllowedOrigin 中的 CORSRule 元素匹配。例如，如果设置 CORSRule 以允许源 `http://www.example.com`，则请求中的 `https://www.example.com` 和 `http://www.example.com:80` 都与配置中允许的源不匹配。

- c. 验证请求中的方法 (在预检请求中，则为在 Access-Control-Request-Method 中指定的方法) 是否为同一 AllowedMethod 中的 CORSRule 元素之一。

- d. 对于预检请求，如果请求包含 Access-Control-Request-Headers 标头，请验证对于 CORSRule 标头中的每个值，AllowedHeader 是否包含 Access-Control-Request-Headers 条目。

## 在对象上的操作

Amazon S3 允许您存储、检索和删除对象。您可以检索整个对象或部分对象。如果您已在您的存储桶上启用了版本控制，则可以检索特定版本的对象。您也可以检索与您的对象相关联的子资源，并在合适的位置更新它。您可以创建现有对象的副本。根据对象大小，适用以下与上传和复制相关的注意事项：

- 上传对象 – 您可以在单个操作中上传大小最多为 5 GB 的对象。对于大于 5 GB 的对象，您必须使用分段上传 API。

使用分段上传 API，您可以每次上传最多 5 TB 的对象。有关更多信息，请参阅 [使用分段上传 API 上传对象 \(p. 155\)](#)。

- 复制对象 – 复制操作将创建已存储在 Amazon S3 中的对象的副本。

您可以在单个原子操作中创建大小最多为 5 GB 的对象的副本。但是，要复制大于 5 GB 的对象，您必须使用分段上传 API。有关更多信息，请参阅 [复制对象 \(p. 187\)](#)。

您可以使用 REST API (参阅 [使用 REST API 创建请求 \(p. 41\)](#)) 来使用对象，或使用以下 AWS 开发工具包库之一：

- [AWS SDK for Java](#)
- [适用于 .NET 的 AWS 开发工具包](#)
- [适用于 PHP 的 AWS 开发工具包](#)

这些库提供了更易于使用对象的高级别抽象。但是，如果需要您的应用程序，您可以直接使用 REST API。

## 获取对象

### 主题

- [相关资源 \(p. 142\)](#)
- [使用 AWS SDK for Java 获取对象 \(p. 142\)](#)
- [使用 适用于 .NET 的 AWS 开发工具包 获取对象 \(p. 144\)](#)
- [使用 适用于 PHP 的 AWS 开发工具包 获取对象 \(p. 146\)](#)
- [使用 REST API 获取对象 \(p. 147\)](#)
- [与其他用户共享数据元 \(p. 147\)](#)

您可以直接从 Amazon S3 中检索对象。检索对象时，您可以使用以下选项：

- 检索整个对象 – 单次 GET 操作即可为您返回 Amazon S3 中存储的整个对象。
- 检索分段中的对象 – 通过在 GET 请求中使用 Range HTTP 标头，您可以检索存储在 Amazon S3 中的对象的特定字节范围。

一旦您的应用程序准备就绪，就会恢复提取对象的其他分段的操作。当您仅需要部分对象数据时，这种可恢复的下载十分有用。当网络连接不佳，并且您需要对失败做出反应时，它也十分有用。

### Note

Amazon S3 不支持根据 GET 请求检索多个范围的数据。

当您检索对象时，其元数据将在响应标头中返回。有时，您希望覆盖 GET 响应中返回的特定响应标头值。例如，您可能覆盖 GET 请求中的 Content-Disposition 响应标头值。REST GET Object API (参阅 [GET Object](#)) 允许您指定 GET 请求中的查询字符串参数以覆盖这些值。

适用于 Java、.NET 和 PHP 的 AWS 开发工具包同样提供了必需的对象，您可以使用它们在 GET 请求中指定这些响应标头的值。

检索使用服务器端加密进行加密存储的对象时，需要提供合适的请求标头。有关更多信息，请参阅 [使用加密保护数据 \(p. 345\)](#)。

## 相关资源

- [使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)

## 使用 AWS SDK for Java 获取对象

当您通过 AWS SDK for Java 下载某个对象时，Amazon S3 将返回该对象的所有元数据以及从中读取该对象的内容的输入流。

要检索对象，请执行以下操作：

- 执行 `AmazonS3Client.getObject()` 方法，并在请求中提供存储桶名称和对象键。
- 执行 `S3Object` 实例方法之一以处理输入流。

**Note**

您的网络连接将保持打开状态，直到您读取所有数据或关闭输入流。建议您尽快读取流的内容。

下面是您可能使用的一些变体：

- 您可以通过在请求中指定所需的字节范围来仅读取部分对象数据，而不是读取整个对象。
- 您可以选择通过使用 `ResponseHeaderOverrides` 对象并设置相应的请求属性来替代响应标头值（请参阅[获取对象 \(p. 142\)](#)）。例如，您可以使用此功能指示应该将对象下载到具有与对象键名称不同的文件名的文件中。

下面的例子通过三种方式从 Amazon S3 存储桶中检索对象：作为完整的对象、作为来自该对象的一系列字节以及作为包含被覆盖的响应标头值的完整对象。有关从 Amazon S3 中获取对象的更多信息，请参阅[GET Object](#)。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

public class GetObject {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String key = "*** Object key ***";

        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();

            // Get an object and print its contents.
            System.out.println("Downloading an object");
            fullObject = s3Client.getObject(new GetObjectRequest(bucketName, key));
            System.out.println("Content-Type: " +
                fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(bucketName, key)
                .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());
        }
    }

    private void displayTextInputStream(InputStream inputStream) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
        String line;
        try {
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
// Get an entire object, overriding the specified response headers, and print
the object's content.
ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
    .withCacheControl("No-cache")

    .withContentDisposition("attachment; filename=example.txt");
    GetObjectRequest getObjectRequestHeaderOverride = new
GetObjectRequest(bucketName, key)

    .withResponseHeaders(headerOverrides);
        headerOverrideObject = s3Client.getObject(getObjectRequestHeaderOverride);
        displayTextInputStream(headerOverrideObject.getObjectContent());
    }
    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();
    }
    finally {
        // To ensure that the network connection doesn't remain open, close any open
input streams.
        if(fullObject != null) {
            fullObject.close();
        }
        if(objectPortion != null) {
            objectPortion.close();
        }
        if(headerOverrideObject != null) {
            headerOverrideObject.close();
        }
    }
}

private static void displayTextInputStream(InputStream input) throws IOException {
    // Read the text input stream one line at a time and display each line.
    BufferedReader reader = new BufferedReader(new InputStreamReader(input));
    String line = null;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
    System.out.println();
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包 获取对象

当下载某个对象时，您将获得该对象的所有元数据以及从中读取内容的流。您应该尽快读取流的内容，因为数据直接源自 Amazon S3，在您读取所有数据或关闭输入流之前，您的网络连接将保持打开状态。您通过执行以下操作来获取对象：

- 通过在请求中提供存储桶名称和对象键来执行 `GetObject` 方法。
- 执行 `GetObjectResponse` 方法之一以处理流。

下面是您可能使用的一些变体：

- 您可以通过在请求中指定字节范围来仅读取部分对象，而不是读取整个对象，如以下 C# 示例所示：

### Example

```
GetObjectRequest request = new GetObjectRequest
{
    BucketName = bucketName,
    Key = keyName,
    ByteRange = new ByteRange(0, 10)
};
```

- 在检索对象时，您可以选择通过使用 [获取对象 \(p. 142\)](#) 对象并设置相应的请求属性来替换响应标头值（参阅 `ResponseHeaderOverrides`）。以下 C# 代码示例演示了如何执行此操作。您可以使用此功能指示应该将对象下载到具有与对象键名称不同的文件名的文件中。

### Example

```
GetObjectRequest request = new GetObjectRequest
{
    BucketName = bucketName,
    Key = keyName
};

ResponseHeaderOverrides responseHeaders = new ResponseHeaderOverrides();
responseHeaders.CacheControl = "No-cache";
responseHeaders.ContentDisposition = "attachment; filename=testing.txt";

request.ResponseHeaderOverrides = responseHeaders;
```

### Example

以下 C# 代码示例从 Amazon S3 存储桶检索对象。通过该响应，示例将读取使用 `GetObjectResponse.ResponseStream` 属性的对象数据。该示例还介绍如何使用 `GetObjectResponse.Metadata` 集合读取对象元数据。如果您检索的对象具有 `x-amz-meta-title` 元数据，则该代码会打印元数据值。

有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

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

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ReadObjectDataAsync().Wait();
        }

        private static void ReadObjectDataAsync()
        {
            var request = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName
            };
            var response = client.GetObject(request);
            var stream = response.ResponseStream;
            var metadata = response.Metadata;
            var title = metadata["x-amz-meta-title"];
            Console.WriteLine("Title: {0}", title);
        }
    }
}
```

```

        }

        static async Task ReadObjectDataAsync()
        {
            string responseBody = "";
            try
            {
                GetObjectRequest request = new GetObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
                using (GetObjectResponse response = await client.GetObjectAsync(request))
                using (Stream responseStream = response.ResponseStream)
                using (StreamReader reader = new StreamReader(responseStream))
                {
                    string title = response.Metadata["x-amz-meta-title"]; // Assume you
have "title" as medata added to the object.
                    string contentType = response.Headers["Content-Type"];
                    Console.WriteLine("Object metadata, Title: {0}", title);
                    Console.WriteLine("Content type: {0}", contentType);

                    responseBody = reader.ReadToEnd(); // Now you process the response
body.
                }
            }
            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);
            }
        }
    }
}

```

## 使用适用于 PHP 的 AWS 开发工具包 获取对象

本主题说明如何使用适用于 PHP 的 AWS 开发工具包中的类检索 Amazon S3 对象。您可以检索整个对象或对象中的字节范围。我们假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并且正确安装了适用于 PHP 的 AWS 开发工具包。

在检索对象时，您可以选择覆盖响应标头值，方法是将响应

键、`ResponseContentType`、`ResponseContentLanguage`、`ResponseContentDisposition`、`ResponseCacheControl` 和 `ResponseExpires` 添加到 `getObject()` 方法，如以下 PHP 代码示例所示：

### Example

```
$result = $s3->getObject([
    'Bucket'                  => $bucket,
    'Key'                     => $keyname,
    'ResponseContentType'     => 'text/plain',
    'ResponseContentLanguage' => 'en-US',
    'ResponseContentDisposition' => 'attachment; filename=testing.txt',
    'ResponseCacheControl'    => 'No-cache',
    'ResponseExpires'          => gmdate(DATE_RFC2822, time() + 3600),
]);

```

有关检索对象的更多信息，请参阅[获取对象 \(p. 142\)](#)。

以下 PHP 示例将检索对象，并在浏览器中显示对象的内容。该示例演示如何使用 `getObject()` 方法。有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

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

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

try {
    // Get the object.
    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'     => $keyname
    ]);

    // Display the object in the browser.
    header("Content-Type: {$result['ContentType']}");
    echo $result['Body'];
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 `Aws\S3\S3Client` 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用 REST API 获取对象

您可以使用 AWS 开发工具包从存储桶检索对象键。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。您可以发送 GET 请求以检索对象键。有关请求和响应格式的详细信息，请参阅[Get Object](#)。

## 与其他用户共享数据元

### 主题

- [使用 AWS Explorer for Visual Studio 生成预签名对象 URL \(p. 148\)](#)
- [使用 AWS SDK for Java 生成预签名对象 URL \(p. 148\)](#)
- [使用适用于 .NET 的 AWS 开发工具包生成预签名对象 URL \(p. 149\)](#)

默认情况下，所有的对象都为私有。只有对象所有者具有访问这些对象的权限。但是，对象所有者可以选择使用自己的安全凭证来创建预签名的 URL，以授予有限时间内的下载对象许可，从而与其他用户共享对象。

当您为对象创建预签名的 URL 时，必须提供安全凭证、指定存储桶名称和对象键、指定 HTTP 方法（指定为 GET 以下载对象）和过期日期和时间。预签名 URL 仅在指定的持续时间内有效。

然后，收到预签名 URL 的任何人都可以访问对象。例如，如果您在存储桶中具有一段视频，并且存储桶和对象均为私有，您可以通过生成预签名的 URL 来与其他用户共享视频。

#### Note

具有有效安全凭证的任何人都可以创建预签名 URL。然而，为了成功地访问对象，必须由拥有执行预签名 URL 所基于的操作许可的人创建预签名 URL。

您可以通过使用 AWS SDK for Java 和 .NET，以编程方式生成预签名 URL。

## 使用 AWS Explorer for Visual Studio 生成预签名对象 URL

如果您使用的是 Visual Studio，可以使用 AWS Explorer for Visual Studio 为对象生成预签名 URL，而不需要编写任何代码。具有该 URL 的任何人均可下载对象。有关详细信息，请参阅 [Using Amazon S3 from AWS Explorer](#)。

有关安装 AWS Explorer 的说明，请参阅 [使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)。

## 使用AWS SDK for Java生成预签名对象 URL

#### Example

以下示例将生成预签名 URL，您可以将其提供给其他用户，以便他们可以从 S3 存储桶中检索对象。有关更多信息，请参阅 [与其他用户共享数据元 \(p. 147\)](#)。

有关创建和测试有效示例的说明，请参阅 [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;
import java.net.URL;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;

public class GeneratePresignedURL {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String objectKey = "*** Object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();

            // Set the presigned URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = expiration.getTime();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the presigned URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest =
                new GeneratePresignedUrlRequest(bucketName, objectKey)
                    .withMethod(HttpMethod.GET)
                    .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);
        }
    }
}
```

```
        System.out.println("Pre-Signed URL: " + url.toString());
    }
    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();
    }
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包生成预签名对象 URL

### Example

以下示例将生成预签名 URL，您可以将其提供给其他用户，以便他们可以检索对象。有关更多信息，请参阅 [与其他用户共享数据元 \(p. 147\)](#)。

有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;

namespace Amazon.DocSamples.S3
{
    class GenPresignedURLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string objectKey = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            string urlString = GeneratePreSignedURL();
        }
        static string GeneratePreSignedURL()
        {
            string urlString = "";
            try
            {
                GetPreSignedUrlRequest request1 = new GetPreSignedUrlRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.Now.AddMinutes(5)
                };
                urlString = s3Client.GetPreSignedURL(request1);
            }
            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);
        }
        return urlString;
    }
}
```

## 上传对象

根据上传的数据大小，Amazon S3 提供以下选项：

- 在单个操作中上传对象 - 您可以在单个 PUT 操作中上传最大 5 GB 的对象。

有关更多信息，请参阅 [在单个操作中上传对象 \(p. 150\)](#)。

- 分段上传对象 - 您可以使用分段上传 API 来上传最大 5 TB 的大型对象。

分段上传 API 旨在改进大型对象的上传体验。您可以分段上传对象。这些对象分段可以按任何顺序并行独立上传。您可以对大小在 5 MB 到 5 TB 范围内的对象使用分段上传。有关更多信息，请参阅 [使用分段上传 API 上传对象 \(p. 155\)](#)。

我们建议您按以下方式使用分段上传：

- 如果您在稳定的高带宽网络上传大型对象，请使用分段上传以充分利用您的可用带宽，通过并行分段上传对象实现多线程上传。
- 如果您在断点网络中上传对象，请使用分段上传以提高应对网络错误的复原能力，从而避免重新上传。在使用分段上传时，您只需重新尝试上传在上传期间中断的部分即可，而无需从头重新开始上传对象。

有关分段上传的更多信息，请参阅 [分段上传概述 \(p. 155\)](#)。

### 主题

- [在单个操作中上传对象 \(p. 150\)](#)
- [使用分段上传 API 上传对象 \(p. 155\)](#)
- [使用预签名 URL 上传对象 \(p. 183\)](#)

在上传对象时，您可以选择请求 Amazon S3 在将对象保存到磁盘之前对其进行加密，并在下载时对其进行解密。有关更多信息，请参阅 [使用加密保护数据 \(p. 345\)](#)。

### 相关主题

[使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)

## 在单个操作中上传对象

### 主题

- [使用 AWS SDK for Java 上传对象 \(p. 151\)](#)
- [使用适用于 .NET 的 AWS 开发工具包 上传对象 \(p. 152\)](#)
- [使用适用于 PHP 的 AWS 开发工具包 上传对象 \(p. 153\)](#)
- [使用适用于 Ruby 的 AWS 开发工具包 上传对象 \(p. 154\)](#)
- [使用 REST API 上传对象 \(p. 154\)](#)

您可以使用 AWS 开发工具包上传对象。开发工具包向您提供了包装程序库，使您可以更轻松地上传数据。但是，如果您的应用程序需要它，您可以直接在您的应用程序中使用 REST API。

## 使用 AWS SDK for Java 上传对象

### Example

以下示例将创建两个对象。第一个对象将一个文本字符串作为数据，第二对象是一个文件。该示例通过在对 `AmazonS3Client.putObject()` 的调用中直接指定存储桶名称、对象键和文本数据来创建第一个对象。该示例通过使用指定存储桶、对象键和文件路径的 `PutObjectRequest` 来创建第二个对象。`PutObjectRequest` 还指定 `ContentType` 标头和标题元数据。

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.File;
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

public class UploadObject {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();

            // Upload a text string as a new object.
            s3Client.putObject(bucketName, stringObjKeyName, "Uploaded String Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(bucketName, fileObjKeyName, new
File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("x-amz-meta-title", "someTitle");
            request.setMetadata(metadata);
            s3Client.putObject(request);
        }
        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();
        }
    }
}
```

}

## 使用适用于 .NET 的 AWS 开发工具包 上传对象

### Example

以下 C# 代码示例将使用两个 PutObjectRequest 请求创建两个对象：

- 第一个 PutObjectRequest 请求将文本字符串保存为示例对象数据。它还指定存储桶和对象键名称。
- 第二个 PutObjectRequest 请求通过指定文件名上传文件。此请求还指定 ContentType 标头和可选对象元数据 (标题)。

有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class UploadObjectTest
    {
        private const string bucketName = "**** bucket name ****";
        // Example creates two objects (for simplicity, we upload same file twice).
        // You specify key names for these objects.
        private const string keyName1 = "**** key name for first object created ****";
        private const string keyName2 = "**** key name for second object created ****";
        private const string filePath = @"**** file path ****";
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.EUWest1;

        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                // 1. Put object-specify only key name for the new object.
                var putRequest1 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName1,
                    ContentBody = "sample text"
                };

                PutObjectResponse response1 = await client.PutObjectAsync(putRequest1);

                // 2. Put the object-set ContentType and add metadata.
                var putRequest2 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName2,
```

```
        FilePath = filePath,
        ContentType = "text/plain"
    );
    putRequest2.Metadata.Add("x-amz-meta-title", "someTitle");
}
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);
}
}
}
```

## 使用适用于 PHP 的 AWS 开发工具包 上传对象

此主题将指导您使用适用于 PHP 的 AWS 开发工具包 中的类上传最大 5 GB 的对象。对于大型文件，您必须使用分段上传 API。有关更多信息，请参阅 [使用分段上传 API 上传对象 \(p. 155\)](#)。

此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并正确安装了适用于 PHP 的 AWS 开发工具包。

### Example 通过上传数据在 Amazon S3 存储桶中创建对象

通过使用 `putObject()` 方法来上传数据，以下 PHP 示例在指定存储桶中创建对象。有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

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

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

try {
    // Upload data.
    $result = $s3->putObject([
        'Bucket' => $bucket,
        'Key'     => $keyname,
        'Body'   => 'Hello, world!',
        'ACL'    => 'public-read'
    ]);

    // Print the URL to the object.
    echo $result['ObjectURL'] . PHP_EOL;
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 Aws\S3\S3Client 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用适用于 Ruby 的 AWS 开发工具包 上传对象

适用于 Ruby 的 AWS 开发工具包 - 版本 3 通过两种方式将对象上传到 Amazon S3。第一种方式使用托管的文件上传程序，从而能轻松从磁盘上传任何大小的文件。使用托管文件上传程序方法：

1. 创建 Aws::S3::Resource 类的实例。
2. 按存储桶名称和键引用目标对象。位于存储桶中的对象具有可识别每个对象的唯一密钥。
3. 在对象上调用 #upload\_file。

### Example

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(region:'us-west-2')
obj = s3.bucket('bucket-name').object('key')
obj.upload_file('/path/to/source/file')
```

适用于 Ruby 的 AWS 开发工具包 版本 3 上传对象的第二种方式使用 Aws::S3::Object 的 #put 方法。如果对象为字符串或者为不是磁盘上的文件的 I/O 对象，此方式很有用。要使用此方法，请执行以下操作：

1. 创建 Aws::S3::Resource 类的实例。
2. 按存储桶名称和键引用目标对象。
3. 调用 #put，从而传入字符串或 I/O 对象。

### Example

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(region:'us-west-2')
obj = s3.bucket('bucket-name').object('key')

# string data
obj.put(body: 'Hello World!')

# I/O object
File.open('/path/to/source.file', 'rb') do |file|
  obj.put(body: file)
end
```

## 使用 REST API 上传对象

您可以使用 AWS 开发工具包上传对象。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。您可以发送 PUT 请求以在单个操作中上传数据。有关更多信息，请参阅 [PUT Object](#)。

## 使用分段上传 API 上传对象

### 主题

- [分段上传概述 \(p. 155\)](#)
- [使用适用于分段上传的 AWS Java 开发工具包 \(高级别 API\) \(p. 160\)](#)
- [使用适用于分段上传的 AWS Java 开发工具包 \(低级别 API\) \(p. 164\)](#)
- [使用适用于 .NET 的 AWS 开发工具包进行分段上传 \(高级别 API\) \(p. 169\)](#)
- [使用适用于 .NET 的 AWS 开发工具包 \(低级别 API\) 执行分段上传 \(p. 175\)](#)
- [使用 AWS PHP 开发工具包执行分段上传 \(p. 179\)](#)
- [使用适用于分段上传的 AWS PHP 开发工具包 \(低级别 API\) \(p. 180\)](#)
- [使用适用于 Ruby 的 AWS 开发工具包进行分段上传 \(p. 182\)](#)
- [使用适用于分段上传的 REST API \(p. 183\)](#)

分段上传允许上传单个对象作为一组分段。每个分段都是对象数据的连续部分。您可以独立上传以及按任意顺序上传这些对象分段。如果任意分段传输失败，可以重新传输该分段且不会影响其他分段。当对象的所有分段都上传后，Amazon S3 将这些段汇编起来，然后创建对象。一般而言，如果您的对象大小达到了 100 MB，您应该考虑使用分段上传，而不是在单个操作中上传对象。

使用分段上传可提供以下优势：

- 提高吞吐量 – 您可以并行上传分段以提高吞吐量。
- 从任何网络问题中快速恢复 – 较小的分段大小可以将由于网络错误而需重启失败的上传所产生的影响降至最低。
- 暂停和恢复对象上传 – 您可以在一段时间内逐步上传对象分段。启动分段上传后，不存在过期期限；您必须显式地完成或中止分段上传。
- 在您知道对象的最终大小前开始上传 – 您可以在创建对象时将其上传。

有关更多信息，请参阅 [分段上传概述 \(p. 155\)](#)。

## 分段上传概述

### 主题

- [并发分段上传操作 \(p. 156\)](#)
- [分段上传和定价 \(p. 156\)](#)
- [使用存储桶生命周期策略中止未完成的分段上传 \(p. 156\)](#)
- [Amazon S3 分段上传限制 \(p. 158\)](#)
- [分段上传的 API 支持 \(p. 158\)](#)
- [分段上传 API 和权限 \(p. 159\)](#)

分段上传 API 允许您分段上传大型对象。您可以使用此 API 上传新的大型对象或创建现有对象的副本（参阅 [在对象上的操作 \(p. 141\)](#)）。

分段上传有三个步骤：开始上传、上传对象分段，以及在上传所有分段后，完成分段上传。收到完成分段上传请求后，Amazon S3 将构建来自已上传分段的对象，然后您可以像在您的存储桶中访问任何其他对象一样访问该对象。

您可以列出所有正在进行的分段上传，或获取已对特定分段上传上传的分段列表。以上每个操作都在本节中进行了说明。

### 分段上传开始

当您发送请求以开始分段上传时，Amazon S3 将返回具有上传 ID 的响应，此 ID 是分段上传的唯一标识符。无论您何时上传分段、列出分段、完成上传或中止上传，您都必须包括此上传 ID。如果您想要提供描述已上传的对象的任何元数据，必须在请求中提供它以开始分段上传。

### 分段上传

上传分段时，除了指定上传 ID，还必须指定分段编号。您可以选择 1 和 10000 之间的任意分段编号。分段编号在您正在上传的对象中唯一地识别分段及其位置。您选择的分段编号不必是连续序列（例如，它可以是 1、5 和 14）。如果您使用之前上传的分段的同一分段编号上传新分段，则之前上传的分段将被覆盖。无论您何时上传分段，Amazon S3 都将在其响应中返回 ETag 标头。对于每个分段上传，您必须记录分段编号和 ETag 值。您需要在随后的请求中包括这些值以完成分段上传。

#### Note

启动分段上传并上传一个或多个段之后，您必须完成或中止分段上传，才能停止收取上传的段的存储费用。只有在完成或中止分段上传之后，Amazon S3 才会释放段存储并停止向您收取段存储费用。

### 分段上传完成 (或中止)

完成分段上传后，Amazon S3 会按段编号的升序顺序将各个段连接起来，从而创建对象。如果在开始分段上传请求中提供了任何对象元数据，则 Amazon S3 会将该元数据与对象相关联。成功完成请求后，分段将不再存在。完成分段上传请求必须包括上传 ID 以及分段编号和相应的 ETag 值的列表。Amazon S3 响应包括可唯一地识别组合对象数据的 ETag。此 ETag 无需成为对象数据的 MD5 哈希。您可以选择中止分段上传。中止分段上传后，无法再次使用该上传 ID 上传任何分段。然后，将释放使用的中止分段上传中的任何分段的所有存储。如果任何分段上传正在进行中，即使您已执行中止操作，它们仍可以上传成功或失败。要释放所有分段使用的所有存储，必须在完成所有分段上传后仅中止分段上传。

### 分段上传列表

您可以列出特定分段上传或所有正在进行的分段上传的分段。列出分段操作将返回您已为特定分段上传而上传的分段信息。对于每个列出分段请求，Amazon S3 将返回有关特定分段上传的分段信息，最多为 1000 个分段。如果分段上传中的分段超过 1000 个，您必须发送一系列列出分段请求以检索所有分段。请注意，返回的段列表不包括未完成上传的段。使用列出分段上传操作，您可以获得正在进行的分段上传的列表。正在进行的分段上传是已开始但还未完成或中止的上传。每个请求将返回最多 1000 个分段上传。如果正在进行的分段上传超过 1000 个，您需要发送其他请求以检索剩余的分段上传。仅使用返回的列表进行验证。发送完成分段上传请求时，您不应使用此列表的结果。但是，当上传分段和 Amazon S3 返回的相应的 ETag 值时，将保留您自己的指定分段编号的列表。

### 并发分段上传操作

在分布式开发环境中，您的应用程序可以同时在同一对象上开始多个更新。您的应用程序可能会使用同一对象键开始多个分段上传。然后，对于其中每个上传，您的应用程序可以上传分段并将完成上传请求发送到 Amazon S3，以创建对象。当存储桶启用了版本控制时，完成分段上传将始终创建一个新版本。对于未启用版本控制的存储桶，在开始分段上传和完成分段上传期间接收的某些其他请求可能会优先开始。

#### Note

在开始分段上传和完成分段上传期间接收的某些其他请求可能会优先开始。例如，如果在使用键开始分段上传之后，但在完成分段上传之前其他操作删除了该键，则完成分段上传响应可能表示在未看到对象的情况下即成功创建了对象。

### 分段上传和定价

开始分段上传后，Amazon S3 将保留所有分段，直到您完成或中止上传。在整个其生命周期内，您将支付有关此分段上传及其关联分段的所有存储、带宽和请求的费用。如果您中止分段上传，Amazon S3 将删除上传项目和已上传的任何段，您将不再支付它们的费用。有关定价的更多信息，请参阅 [Amazon S3 定价](#)。

### 使用存储桶生命周期策略中止未完成的分段上传

启动分段上传后，您便开始上传段。Amazon S3 将存储这些段，但它只有在您上传所有这些段并发送完成分段上传的 `successful` 请求（您应验证您的完成分段上传的请求是否成功）之后，才会利用这些段创建对象。在收到完成分段上传请求后，Amazon S3 会将这些段汇集在一起并创建一个对象。

如果您未成功发送完成分段上传请求，则 Amazon S3 不会汇集这些段并且不会创建任何对象。因此，这些段将保留在 Amazon S3 中，您需要为存储在 Amazon S3 中的段付费。作为最佳实践，我们建议您配置生命周期规则（使用 AbortIncompleteMultipartUpload 操作）以最大程度地降低存储成本。

Amazon S3 支持一个存储桶生命周期规则，您可以使用该规则指示 Amazon S3 中止未在启动后的指定天数内完成的分段上传。当分段上传未在规定的时限内完成时，便能够执行中止操作，Amazon S3 将中止分段上传（并删除与分段上传相关的段）。

下面是使用 AbortIncompleteMultipartUpload 操作指定规则的示例生命周期配置。

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix></Prefix>
    <Status>Enabled</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

在示例中，该规则没有为 Prefix 元素（对象键名称前缀）指定值，因此该规则适用于启动了分段上传的存储桶中的所有对象。已启动但未在七天内完成的所有分段上传将能够执行中止操作（该操作对已完成的分段上传没有影响）。

有关存储桶生命周期配置的更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

Note

如果分段上传在规则中指定的天数内完成，则 AbortIncompleteMultipartUpload 生命周期操作不适用（即，Amazon S3 将不会执行任何操作）。此外，此操作不适用于对象，此生命周期操作不会删除任何对象。

以下 put-bucket-lifecycle CLI 命令将为指定的存储桶添加生命周期配置。

```
$ aws s3api put-bucket-lifecycle \
  --bucket bucketname \
  --lifecycle-configuration filename-containing-lifecycle-configuration
```

要测试 CLI 命令，请执行以下操作：

1. 设置 AWS CLI。有关说明，请参阅[设置 AWS CLI \(p. 520\)](#)。
2. 将以下示例生命周期配置保存在一个文件（lifecycle.json）中。该示例配置指定了空前缀，因此它适用于存储桶中的所有对象。您可以指定一个前缀以限制仅对一部分对象应用策略。

```
{
  "Rules": [
    {
      "ID": "Test Rule",
      "Status": "Enabled",
      "Prefix": "",
      "AbortIncompleteMultipartUpload": {
        "DaysAfterInitiation": 7
      }
    }
  ]
}
```

3. 运行以下 CLI 命令以在存储桶上设置生命周期配置。

```
aws s3api put-bucket-lifecycle \
--bucket bucketname \
--lifecycle-configuration file://lifecycle.json
```

4. 要进行验证，请使用 `get-bucket-lifecycle` CLI 命令检索生命周期配置。

```
aws s3api get-bucket-lifecycle \
--bucket bucketname
```

5. 要删除生命周期配置，请使用 `delete-bucket-lifecycle` CLI 命令。

```
aws s3api delete-bucket-lifecycle \
--bucket bucketname
```

## Amazon S3 分段上传限制

下表提供了分段上传的核心规范。有关更多信息，请参阅 [分段上传概述 \(p. 155\)](#)。

Item	规范
最大对象大小	5 TB
每次上传的分段的最大数量	10000
分段编号	1 到 10000 (含)
分段大小	5 MB 到 5 GB，上一个分段可以 <5 MB
列出分段请求返回的分段的最大数量	1000
在列出分段上传请求中返回的分段的最大数量	1000

## 分段上传的 API 支持

您可以使用 AWS 开发工具包分段上传对象。以下 AWS 开发工具库支持分段上传：

- [AWS SDK for Java](#)
- [适用于 .NET 的 AWS 开发工具包](#)
- [适用于 PHP 的 AWS 开发工具包](#)

这些库提供了易于上传分段对象的高级别抽象。但是，如果需要您的应用程序，您可以直接使用 REST API。Amazon Simple Storage Service API Reference 中的以下各部分介绍适用于分段上传的 REST API。

- [开始分段上传](#)
- [上传分段](#)
- [上传分段 \(复制\)](#)
- [完成分段上传](#)
- [中止分段上传](#)
- [列出分段](#)
- [列出分段上传](#)

## 分段上传 API 和权限

用户必须具有使用分段上传操作的所需权限。您可以使用 ACL、存储桶策略或用户策略来授予用户以执行这些操作的权限。下表列出了使用 ACL、存储桶策略或用户策略时，各种分段上传操作的所需权限。

操作	所需权限
开始分段上传	必须允许您对对象执行 <code>s3:PutObject</code> 操作，才能开始分段上传。 存储桶拥有者可以允许其他委托人执行 <code>s3:PutObject</code> 操作。
启动程序	标识分段上传启动者的容器元素。如果启动程序是 AWS 账户，此元素将提供与 Owner 元素相同的信息。如果启动程序是 IAM 用户，此元素将提供用户 ARN 和显示名称。
上传分段	必须允许您对对象执行 <code>s3:PutObject</code> 操作，才能上传分段。 仅分段上传的发起者可以上传分段。存储桶拥有者必须允许发起者对对象执行 <code>s3:PutObject</code> 操作，以便发起者可以上传该对象的分段。
上传分段 (复制)	必须允许您对对象执行 <code>s3:PutObject</code> 操作，才能上传分段。因为您正在上传现有对象的分段，因此必须允许您对源对象执行 <code>s3:GetObject</code> 。 仅分段上传的发起者可以上传分段。存储桶拥有者必须允许发起者对对象执行 <code>s3:PutObject</code> 操作，以便发起者可以上传该对象的分段。
完成分段上传	必须允许您对对象执行 <code>s3:PutObject</code> 操作，才能完成分段上传。 仅分段上传的发起者可以完成该分段上传。存储桶拥有者必须允许发起者对对象执行 <code>s3:PutObject</code> 操作，以便发起者可以完成该对象的分段上传。
中止分段上传	必须允许您执行 <code>s3:AbortMultipartUpload</code> 操作，才能中止分段上传。 默认情况下，允许存储桶拥有者和分段上传的发起者执行此操作。如果发起者是 IAM 用户，也允许该用户的 AWS 账户中止此分段上传。 除了这些默认情况之外，存储桶拥有者可以允许其他委托人对对象执行 <code>s3:AbortMultipartUpload</code> 操作。存储桶拥有者可以拒绝任何委托人，使其无法执行 <code>s3:AbortMultipartUpload</code> 操作。
列出分段	您必须得到可以执行 <code>s3&gt;ListMultipartUploadParts</code> 操作的允许，才能在分段上传中列出分段。 在默认情况下，存储桶拥有者有权为任何针对存储桶的分段上传列出分段。分段上传的发起人有权为特定分段上传列出分段。如果分段上传的发起者是 IAM 用户，则控制该 IAM 用户的 AWS 账户同样有权列出此次上传的分段。 除了这些默认情况之外，存储桶拥有者可以允许其他委托人对对象执行 <code>s3&gt;ListMultipartUploadParts</code> 操作。存储桶拥有者也可以拒绝任何委托人，使其无法执行 <code>s3&gt;ListMultipartUploadParts</code> 操作。
列出分段上传	您必须得到可以对存储桶执行 <code>s3&gt;ListBucketMultipartUploads</code> 操作的允许，才能列出正在上传到该存储桶的分段上传。 除了默认情况之外，存储桶拥有者可以允许其他委托人对存储桶执行 <code>s3&gt;ListBucketMultipartUploads</code> 操作。

有关 ACL 权限与访问策略中的权限之间关系的信息，请参阅[ACL 权限和访问策略权限的映射 \(p. 336\)](#)。有关 IAM 用户的信息，请参阅[使用用户和组](#)。

## 使用适用于分段上传的 AWS Java 开发工具包 (高级别 API)

### 主题

- [上传文件 \(p. 160\)](#)
- [中止分段上传 \(p. 161\)](#)
- [跟踪分段上传进度 \(p. 162\)](#)

AWS SDK for Java公开了一个名为 TransferManager 的高级别 API，用于简化分段上传 (请参阅[使用分段上传 API 上传对象 \(p. 155\)](#))。您可以从文件或流上传数据。您还可以设置高级选项，例如，您想要用于分段上传的分段大小或在上传分段时要使用的并发线程数。您也可以设置可选的对象属性、存储类或 ACL。您可以使用 PutObjectRequest 和 TransferManagerConfiguration 类来设置这些高级选项。

在可能的情况下，TransferManager 会尝试使用多个线程来一次性上传单个上传的多个分段。当处理大型内容大小和高带宽时，此操作可以大幅增加吞吐量。

除了文件上传功能，TransferManager 类能让您中止正在进行的分段上传。启动上传后，上传将被视为正在进行，直到您完成或中止它。TransferManager 将中止在指定的日期和时间之前启动的指定存储桶上的所有正在进行的分段上传。

有关分段上传的更多信息 (包括低级别 API 方法提供的额外功能)，请参阅[使用分段上传 API 上传对象 \(p. 155\)](#)。

### [上传文件](#)

#### Example

下面的示例演示如何使用高级别分段上传 API (TransferManager 类) 上传对象。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.File;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

public class HighLevelMultipartUpload {

    public static void main(String[] args) throws Exception {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String keyName = "**** Object key ****";
        String filePath = "**** Path for file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();
            TransferManager tm = TransferManagerBuilder.standard()
                .withS3Client(s3Client)
                .build();

            // TransferManager processes all transfers asynchronously,
            // so this call returns immediately.
            Upload upload = tm.upload(bucketName, keyName, new File(filePath));
        }
    }
}
```

```
System.out.println("Object upload started");

// Optionally, wait for the upload to finish before continuing.
upload.waitForCompletion();
System.out.println("Object upload complete");
}
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();
}
}
}
```

## 中止分段上传

### Example

以下示例使用高级别 Java API (TransferManager 类) 中止一周前在特定存储桶上启动的正在进行的所有分段上传。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.util.Date;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;

public class HighLevelAbortMultipartUpload {

    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();
            TransferManager tm = TransferManagerBuilder.standard()
                .withS3Client(s3Client)
                .build();

            // sevenDays is the duration of seven days in milliseconds.
            long sevenDays = 1000 * 60 * 60 * 24 * 7;
            Date oneWeekAgo = new Date(System.currentTimeMillis() - sevenDays);
            tm.abortMultipartUploads(bucketName, oneWeekAgo);
        }
        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();
    }
}
```

## 跟踪分段上传进度

高级别 Java 分段上传 API 提供了侦听接口 `ProgressListener`，用于跟踪何时将对象上传到 Amazon S3。进度事件定期向侦听器通知已传输的字节。

以下示例演示了如何订阅 `ProgressEvent` 事件和编写处理程序：

### Example

```
import java.io.File;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

public class HighLevelTrackMultipartUpload {

    public static void main(String[] args) throws Exception {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String keyName = "**** Object key ****";
        String filePath = "**** Path to file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();
            TransferManager tm = TransferManagerBuilder.standard()
                .withS3Client(s3Client)
                .build();
            PutObjectRequest request = new PutObjectRequest(bucketName, keyName, new
File(filePath));

            // To receive notifications when bytes are transferred, add a
            // ProgressListener to your request.
            request.setGeneralProgressListener(new ProgressListener() {
                public void progressChanged(ProgressEvent progressEvent) {
                    System.out.println("Transferred bytes: " +
progressEvent.getBytesTransferred());
                }
            });
            // TransferManager processes all transfers asynchronously,
            // so this call returns immediately.
            Upload upload = tm.upload(request);

            // Optionally, you can wait for the upload to finish before continuing.
            upload.waitForCompletion();
        }
        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();  
}  
}  
}
```

## 使用适用于分段上传的 AWS Java 开发工具包 (低级别 API)

### 主题

- [上传文件 \(p. 164\)](#)
- [列出分段上传 \(p. 166\)](#)
- [中止分段上传 \(p. 166\)](#)

AWS SDK for Java公开了一个低级别 API，与适用于分段上传的 Amazon S3 REST API 非常类似 (请参阅[使用分段上传 API 上传对象 \(p. 155\)](#))。当您需要暂停和恢复分段上传，在上传期间更改分段的大小或者事先不知道上传的数据大小的情况下，可使用低级别 API。当您没有这些要求时，请使用高级别 API (参阅[使用适用于分段上传的 AWS Java 开发工具包 \(高级别 API\) \(p. 160\)](#))。

### 上传文件

以下示例演示如何使用低级别 Java 类上传文件。它将执行以下步骤：

- 使用 `AmazonS3Client.initiateMultipartUpload()` 方法初始化分段上传，并传入 `InitiateMultipartUploadRequest` 对象。
- 保存 `AmazonS3Client.initiateMultipartUpload()` 方法返回的上传 ID。您为随后的每个分段上传操作提供此上传 ID。
- 上传对象的分段。对于每个分段，您将调用 `AmazonS3Client.uploadPart()` 方法。您使用 `UploadPartRequest` 对象提供分段上传信息。
- 对于每个分段，在列表中保存来自 `AmazonS3Client.uploadPart()` 方法的响应的 ETag。您使用 ETag 值完成分段上传。
- 调用 `AmazonS3Client.completeMultipartUpload()` 方法来完成分段上传。

### Example

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CompleteMultipartUploadRequest;
import com.amazonaws.services.s3.model.InitiateMultipartUploadRequest;
import com.amazonaws.services.s3.model.InitiateMultipartUploadResult;
import com.amazonaws.services.s3.model.PartETag;
import com.amazonaws.services.s3.model.UploadPartRequest;
import com.amazonaws.services.s3.model.UploadPartResult;

public class LowLevelMultipartUpload {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String keyName = "**** Key name ****";
        String filePath = "**** Path to file to upload ****";

        File file = new File(filePath);
        long contentLength = file.length();
        long partSize = 5 * 1024 * 1024; // Set part size to 5 MB.
```

```
try {
    AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
        .withRegion(clientRegion)
        .withCredentials(new ProfileCredentialsProvider())
        .build();

    // Create a list of ETag objects. You retrieve ETags for each object part
uploaded,
    // then, after each individual part has been uploaded, pass the list of ETags
to
    // the request to complete the upload.
List<PartETag> partETags = new ArrayList<PartETag>();

    // Initiate the multipart upload.
InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(bucketName, keyName);
    InitiateMultipartUploadResult initResponse =
s3Client.initiateMultipartUpload(initRequest);

    // Upload the file parts.
long filePosition = 0;
for (int i = 1; filePosition < contentLength; i++) {
    // Because the last part could be less than 5 MB, adjust the part size as
needed.
    partSize = Math.min(partSize, (contentLength - filePosition));

    // Create the request to upload a part.
UploadPartRequest uploadRequest = new UploadPartRequest()
        .withBucketName(bucketName)
        .withKey(keyName)
        .withUploadId(initResponse.getUploadId())
        .withPartNumber(i)
        .withFileOffset(filePosition)
        .withFile(file)
        .withPartSize(partSize);

    // Upload the part and add the response's ETag to our list.
UploadPartResult uploadResult = s3Client.uploadPart(uploadRequest);
partETags.add(uploadResult.getPartETag());

    filePosition += partSize;
}

    // Complete the multipart upload.
CompleteMultipartUploadRequest compRequest = new
CompleteMultipartUploadRequest(bucketName, keyName,
        initResponse.getUploadId(), partETags);
    s3Client.completeMultipartUpload(compRequest);
}
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();
}
}
```

## 列出分段上传

### Example

以下示例介绍如何使用低级别 Java API 检索正在进行的分段上传的列表：

```
import java.util.List;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

public class ListMultipartUploads {

    public static void main(String[] args) {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Retrieve a list of all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
ListMultipartUploadsRequest(bucketName);
            MultipartUploadListing multipartUploadListing =
s3Client.listMultipartUploads(allMultipartUploadsRequest);
            List<MultipartUpload> uploads = multipartUploadListing.getMultipartUploads();

            // Display information about all in-progress multipart uploads.
            System.out.println(uploads.size() + " multipart upload(s) in progress.");
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \'" + u.getKey() + "\', id =
" + u.getUploadId());
            }
        } 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();
        }
    }
}
```

## 中止分段上传

您可以通过调用 `AmazonS3Client.abortMultipartUpload()` 方法中止正在进行的分段上传。此方法将删除所有上传到 Amazon S3 的分段并释放资源。您应提供上传 ID、存储桶名称和键名称。

### Example

以下示例介绍如何使用低级别 Java API 中止分段上传。

```
import java.util.List;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AbortMultipartUploadRequest;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

public class LowLevelAbortMultipartUpload {

    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();

            // Find all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
ListMultipartUploadsRequest(bucketName);
            MultipartUploadListing multipartUploadListing =
s3Client.listMultipartUploads(allMultipartUploadsRequest);

            List<MultipartUpload> uploads = multipartUploadListing.getMultipartUploads();
            System.out.println("Before deletions, " + uploads.size() + " multipart uploads
in progress.");

            // Abort each upload.
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = '" + u.getKey() + "', id =
" + u.getUploadId());
                s3Client.abortMultipartUpload(new AbortMultipartUploadRequest(bucketName,
u.getKey(), u.getUploadId()));
                System.out.println("Upload deleted: Key = '" + u.getKey() + "', id = " +
u.getUploadId());
            }

            // Verify that all in-progress multipart uploads have been aborted.
            multipartUploadListing =
s3Client.listMultipartUploads(allMultipartUploadsRequest);
            uploads = multipartUploadListing.getMultipartUploads();
            System.out.println("After aborting uploads, " + uploads.size() + " multipart
uploads in progress.");
        }
        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();
        }
    }
}
```

Note

您可以中止在特定时间之前启动的正在进行的分段上传，而不是单独中止分段上传。对于中止您启动但未完成或已中止的分段上传，此清除操作很有用。有关更多信息，请参阅 [中止分段上传 \(p. 161\)](#)。

## 使用适用于 .NET 的 AWS 开发工具包进行分段上传 (高级别 API)

### 主题

- 使用适用于 .NET 的 AWS 开发工具包 (高级别 API) 将文件上传到 S3 存储桶 (p. 169)
- 上传目录 (p. 170)
- 使用适用于 .NET 的 AWS 开发工具包 (高级别 API) 中止到 S3 存储桶的分段上传 (p. 172)
- 使用适用于 .NET 的 AWS 开发工具包 (高级别 API) 跟踪到 S3 存储桶的分段上传的进度 (p. 173)

适用于 .NET 的 AWS 开发工具包公开了一个高级别 API，用于简化分段上传 (请参阅[使用分段上传 API 上传对象 \(p. 155\)](#))。您可以从文件、目录或流上传数据。有关 Amazon S3 分段上传的更多信息，请参阅[分段上传概述 \(p. 155\)](#)。

`TransferUtility` 类提供了上传文件和目录、跟踪上传进度和中止分段上传的方法。

### 使用适用于 .NET 的 AWS 开发工具包 (高级别 API) 将文件上传到 S3 存储桶

要将文件上传到 S3 存储桶，请使用 `TransferUtility` 类。在从文件上传数据时，您必须提供对象的键名。如果未提供，该 API 将使用文件名作为键名。在从流上传数据时，您必须提供对象的键名。

要设置高级上传选项 – 如段大小、并发上传段时的线程数、元数据、存储类或 ACL — 请使用 `TransferUtilityUploadRequest` 类。

以下 C# 示例将文件分段上传到 Amazon S3 存储桶。它说明如何使用各种 `TransferUtility.Upload` 重载来上传文件。每个对上传的后续调用都将替换先前的上传。有关示例与特定版本的适用于 .NET 的 AWS 开发工具包的兼容性的信息以及有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPUHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object ***";
        private const string filePath = "*** provide the full path name of the file to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadFileAsync().Wait();
        }

        private static async Task UploadFileAsync()
        {
            try
            {
                var fileTransferUtility =
```

```
new TransferUtility(s3Client);

// Option 1. Upload a file. The file name is used as the object key name.
await fileTransferUtility.UploadAsync(filePath, bucketName);
Console.WriteLine("Upload 1 completed");

// Option 2. Specify object key name explicitly.
await fileTransferUtility.UploadAsync(filePath, bucketName, keyName);
Console.WriteLine("Upload 2 completed");

// Option 3. Upload data from a type of System.IO.Stream.
using (var fileToUpload =
    new FileStream(filePath, FileMode.Open, FileAccess.Read))
{
    await fileTransferUtility.UploadAsync(fileToUpload,
        bucketName, keyName);
}
Console.WriteLine("Upload 3 completed");

// Option 4. Specify advanced settings.
var fileTransferUtilityRequest = new TransferUtilityUploadRequest
{
    BucketName = bucketName,
    FilePath = filePath,
    StorageClass = S3StorageClass.StandardInfrequentAccess,
    PartSize = 6291456, // 6 MB.
    Key = keyName,
    CannedACL = S3CannedACL.PublicRead
};
fileTransferUtilityRequest.Metadata.Add("param1", "Value1");
fileTransferUtilityRequest.Metadata.Add("param2", "Value2");

await fileTransferUtility.UploadAsync(fileTransferUtilityRequest);
Console.WriteLine("Upload 4 completed");
}
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);
}

}
```

## 更多信息

[适用于 .NET 的 AWS 开发工具包](#)

[上传目录](#)

您可以使用 TransferUtility 类上传整个目录。默认情况下，该 API 仅上传位于指定目录的根目录中的文件。但是，您可以指定以递归方式上传所有子目录中的文件。

要根据筛选条件选择指定目录中的文件，请指定筛选表达式。例如，要从目录中仅上传 .pdf 文件，请指定 “\*.pdf” 筛选表达式。

在从目录中上传文件时，您不必为生成的对象指定键名。Amazon S3 会使用原始文件路径构造键名。例如，假设您有一个名为 c:\myfolder 的目录，并且此目录具有以下结构：

Example

```
C:\myfolder
    \a.txt
    \b.pdf
    \media\
        An.mp3
```

上传此目录时，Amazon S3 将使用以下键名称：

Example

```
a.txt
b.pdf
media/An.mp3
```

Example

以下 C# 示例将目录上传到 Amazon S3 存储桶。它说明如何使用各种 TransferUtility.UploadDirectory 重载来上传目录。每个对上传的后续调用都将替换先前的上传。有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadDirMPUHighLevelAPITest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string directoryPath = "@*** directory path ***";
        // The example uploads only .txt files.
        private const string wildCard = "*.*";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadDirAsync().Wait();
        }

        private static async Task UploadDirAsync()
        {
            try
            {
                var directoryTransferUtility =
                    new TransferUtility(s3Client);

                // 1. Upload a directory.
                await directoryTransferUtility.UploadDirectoryAsync(directoryPath,
                    existingBucketName);
                Console.WriteLine("Upload statement 1 completed");
            }
        }
    }
}
```

```
// 2. Upload only the .txt files from a directory
//      and search recursively.
await directoryTransferUtility.UploadDirectoryAsync(
    directoryPath,
    existingBucketName,
    wildCard,
    SearchOption.AllDirectories);
Console.WriteLine("Upload statement 2 completed");

// 3. The same as Step 2 and some optional configuration.
//      Search recursively for .txt files to upload.
var request = new TransferUtilityUploadDirectoryRequest
{
    BucketName = existingBucketName,
    Directory = directoryPath,
    SearchOption = SearchOption.AllDirectories,
    SearchPattern = wildCard
};

await directoryTransferUtility.UploadDirectoryAsync(request);
Console.WriteLine("Upload statement 3 completed");
}
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);
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包 (高级别 API) 中止到 S3 存储桶的分段上传

要中止正在进行的分段上传，请使用适用于 .NET 的 AWS 开发工具包中的 `TransferUtility` 类。您提供一个 `DateTime` 值。然后，API 将中止所有在指定日期和时间前启动的分段上传并删除已上传的段。启动上传后，上传将被视作正在进行，直到它完成或您中止它。

由于您需要为与已上传的段相关的所有存储支付费用，因此，完成分段上传以完成创建对象，或者中止分段上传以删除已上传的分段非常重要。有关 Amazon S3 分段上传的更多信息，请参阅[分段上传概述 \(p. 155\)](#)。有关定价的信息，请参阅[分段上传和定价 \(p. 156\)](#)。

以下 C# 示例将中止一周前在特定存储桶上启动的正在进行的所有分段上传。有关示例与特定版本的适用于 .NET 的 AWS 开发工具包的兼容性的信息以及有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
```

```
class AbortMPUUsingHighLevelAPITest
{
    private const string bucketName = "*** provide 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()
    {
        s3Client = new AmazonS3Client(bucketRegion);
        AbortMPUAsync().Wait();
    }

    private static async Task AbortMPUAsync()
    {
        try
        {
            var transferUtility = new TransferUtility(s3Client);

            // Abort all in-progress uploads initiated before the specified date.
            await transferUtility.AbortMultipartUploadsAsync(
                bucketName, DateTime.Now.AddDays(-7));
        }
        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);
        }
    }
}
```

#### Note

您也可以中止特定的分段上传。有关更多信息，请参阅 [使用适用于 .NET 的 AWS 开发工具包 \(低级别\) 列出到 S3 存储桶的分段上传 \(p. 177\)](#)。

#### 更多信息

[适用于 .NET 的 AWS 开发工具包](#)

[使用适用于 .NET 的 AWS 开发工具包 \(高级别 API\) 跟踪到 S3 存储桶的分段上传的进度](#)

以下 C# 示例使用 TransferUtility 类将文件上传到 S3 存储桶并跟踪上传的进度。有关示例与特定版本的适用于 .NET 的 AWS 开发工具包的兼容性的信息以及有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
```

```
class TrackMPUUUsingHighLevelAPITest
{
    private const string bucketName = "*** provide the bucket name ***";
    private const string keyName = "*** provide the name for the uploaded object ***";
    private const string filePath = " *** provide the full path name of the file to
upload ***";
    // Specify your bucket region (an example region is shown).
    private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
    private static IAmazonS3 s3Client;

    public static void Main()
    {
        s3Client = new AmazonS3Client(bucketRegion);
        TrackMPUAsync().Wait();
    }

    private static async Task TrackMPUAsync()
    {
        try
        {
            var fileTransferUtility = new TransferUtility(s3Client);

            // 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 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);
        }
    }

    static void uploadRequest_UploadPartProgressEvent(object sender, UploadProgressArgs
e)
    {
        // Process event.
        Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
    }
}
```

## 更多信息

[适用于 .NET 的 AWS 开发工具包](#)

## 使用适用于 .NET 的 AWS 开发工具包 (低级别 API) 执行分段上传

适用于 .NET 的 AWS 开发工具包公开了一个低级别 API，与用于分段上传的 Amazon S3 REST API 非常类似 (请参阅[使用适用于分段上传的 REST API \(p. 183\)](#))。当需要暂停并恢复分段上传、在上传期间更改分段大小，或者事先不知道数据大小时，请使用低级别 API。如果您没有这些要求，请使用高级别 API (参阅[使用适用于 .NET 的 AWS 开发工具包进行分段上传 \(高级别 API\) \(p. 169\)](#))。

### 主题

- [使用适用于 .NET 的 AWS 开发工具包 \(低级别 API\) 将文件上传到 S3 存储桶 \(p. 175\)](#)
- [使用适用于 .NET 的 AWS 开发工具包 \(低级别\) 列出到 S3 存储桶的分段上传 \(p. 177\)](#)
- [使用适用于 .NET 的 AWS 开发工具包 \(低级别\) 跟踪到 S3 存储桶的分段上传的进度 \(p. 177\)](#)
- [使用适用于 .NET 的 AWS 开发工具包 \(低级别\) 中止到 S3 存储桶的分段上传 \(p. 178\)](#)

### 使用适用于 .NET 的 AWS 开发工具包 (低级别 API) 将文件上传到 S3 存储桶

以下 C# 示例演示如何使用低级别适用于 .NET 的 AWS 开发工具包分段上传 API 将文件上传到 S3 存储桶。有关 Amazon S3 分段上传的信息，请参阅[分段上传概述 \(p. 155\)](#)。

#### Note

如果使用适用于 .NET 的 AWS 开发工具包 API 上传大型对象，数据写入到请求流时可能出现超时。您可以使用 `UploadPartRequest` 设置显式超时。

以下 C# 示例使用低级别分段上传 API 将文件上传到 S3 存储桶。有关示例与特定版本的适用于 .NET 的 AWS 开发工具包的兼容性的信息以及有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class UploadFileMPULowLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object ***";
        private const string filePath = "*** provide the full path name of the file to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Uploading an object");
            UploadObjectAsync().Wait();
        }

        private static async Task UploadObjectAsync()
        {
            // Create list to store upload part responses.
```

```
List<UploadPartResponse> uploadResponses = new List<UploadPartResponse>();  
  
// Setup information required to initiate the multipart upload.  
InitiateMultipartUploadRequest initiateRequest = new  
InitiateMultipartUploadRequest  
{  
    BucketName = bucketName,  
    Key = keyName  
};  
  
// Initiate the upload.  
InitiateMultipartUploadResponse initResponse =  
    await s3Client.InitiateMultipartUploadAsync(initiateRequest);  
  
// Upload parts.  
long contentLength = new FileInfo(filePath).Length;  
long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB  
  
try  
{  
    Console.WriteLine("Uploading parts");  
  
    long filePosition = 0;  
    for (int i = 1; filePosition < contentLength; i++)  
    {  
        UploadPartRequest uploadRequest = new UploadPartRequest  
        {  
            BucketName = bucketName,  
            Key = keyName,  
            UploadId = initResponse.UploadId,  
            PartNumber = i,  
            PartSize = partSize,  
            FilePosition = filePosition,  
            FilePath = filePath  
        };  
  
        // Track upload progress.  
        uploadRequest.StreamTransferProgress +=  
            new  
Event Handler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);  
  
        // Upload a part and add the response to our list.  
        uploadResponses.Add(await s3Client.UploadPartAsync(uploadRequest));  
  
        filePosition += partSize;  
    }  
  
    // Setup to complete the upload.  
    CompleteMultipartUploadRequest completeRequest = new  
CompleteMultipartUploadRequest  
{  
    BucketName = bucketName,  
    Key = keyName,  
    UploadId = initResponse.UploadId  
};  
completeRequest.AddPartETags(uploadResponses);  
  
// Complete the upload.  
CompleteMultipartUploadResponse completeUploadResponse =  
    await s3Client.CompleteMultipartUploadAsync(completeRequest);  
}  
catch (Exception exception)  
{  
    Console.WriteLine("An AmazonS3Exception was thrown: { 0 }",  
exception.Message);
```

```
// Abort the upload.  
AbortMultipartUploadRequest abortMPURequest = new  
AbortMultipartUploadRequest  
{  
    BucketName = bucketName,  
    Key = keyName,  
    UploadId = initResponse.UploadId  
};  
await s3Client.AbortMultipartUploadAsync(abortMPURequest);  
}  
}  
public static void UploadPartProgressEventCallback(object sender,  
StreamTransferProgressArgs e)  
{  
    // Process event.  
    Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);  
}  
}  
}
```

## 更多信息

### 适用于 .NET 的 AWS 开发工具包

#### 使用适用于 .NET 的 AWS 开发工具包 (低级别) 列出到 S3 存储桶的分段上传

要列出特定存储桶上所有正在进行的分段上传，请使用适用于 .NET 的 AWS 开发工具包低级别分段上传 API 的 `ListMultipartUploadsRequest` 类。AmazonS3Client.ListMultipartUploads 方法将返回 `ListMultipartUploadsResponse` 类 (提供有关正在进行的分段上传的信息) 的实例。

正在进行的分段上传是使用启动分段上传请求启动但尚未完成或中止的分段上传。有关 Amazon S3 分段上传的更多信息，请参阅[分段上传概述 \(p. 155\)](#)。

以下 C# 示例演示如何使用适用于 .NET 的 AWS 开发工具包列出存储桶上所有正在进行的分段上传。有关示例与特定版本的适用于 .NET 的 AWS 开发工具包的兼容性的信息以及有关如何创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest  
{  
    BucketName = bucketName // Bucket receiving the uploads.  
};  
  
ListMultipartUploadsResponse response = await  
    AmazonS3Client.ListMultipartUploadsAsync(request);
```

## 更多信息

### 适用于 .NET 的 AWS 开发工具包

#### 使用适用于 .NET 的 AWS 开发工具包 (低级别) 跟踪到 S3 存储桶的分段上传的进度

要跟踪分段上传的进度，请使用适用于 .NET 的 AWS 开发工具包低级别分段上传 API 提供的 `UploadPartRequest.StreamTransferProgress` 事件。该事件定期发生。它将返回要传输的总字节数以及已传输的字节数等信息。

以下 C# 示例演示如何跟踪分段上传的进度。有关包含以下代码的完整 C# 示例，请参阅[使用适用于 .NET 的 AWS 开发工具包 \(低级别 API\) 将文件上传到 S3 存储桶 \(p. 175\)](#)。

```
UploadPartRequest uploadRequest = new UploadPartRequest  
{  
    // Provide the request data.
```

```
};

uploadRequest.StreamTransferProgress +=
    new EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

...
public static void UploadPartProgressEventCallback(object sender,
    StreamTransferProgressArgs e)
{
    // Process the event.
    Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
}
```

## 更多信息

[适用于 .NET 的 AWS 开发工具包](#)

[使用适用于 .NET 的 AWS 开发工具包 \(低级别\) 中止到 S3 存储桶的分段上传](#)

您可以通过调用 `AmazonS3Client.AbortMultipartUploadAsync` 方法中止正在进行的分段上传。除了中止上传之外，此方法还将删除已上传到 Amazon S3 的所有分段。

要中止分段上传，请提供上传时使用的上传 ID、存储桶名称和键名称。中止一个分段上传之后，您便无法使用相应的上传 ID 上传其他分段。有关 Amazon S3 分段上传的更多信息，请参阅[分段上传概述 \(p. 155\)](#)。

以下 C# 示例演示如何中止分段上传。有关包含以下代码的完整 C# 示例，请参阅[使用适用于 .NET 的 AWS 开发工具包 \(低级别 API\) 将文件上传到 S3 存储桶 \(p. 175\)](#)。

```
AbortMultipartUploadRequest abortMPURequest = new AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = keyName,
    UploadId = initResponse.UploadId
};
await AmazonS3Client.AbortMultipartUploadAsync(abortMPURequest);
```

还可以中止在特定时间之前启动的所有正在进行的分段上传。对于中止未完成或已中止的分段上传，此清理操作很有用。有关更多信息，请参阅[使用适用于 .NET 的 AWS 开发工具包 \(高级别 API\) 中止到 S3 存储桶的分段上传 \(p. 172\)](#)。

## 更多信息

[适用于 .NET 的 AWS 开发工具包](#)

## 使用 AWS PHP 开发工具包执行分段上传

可以将大型文件分成多个分段上传至 Amazon S3。您必须将分段上传用于大于 5 GB 的文件。适用于 PHP 的 AWS 开发工具包将公开可简化分段上传的 [MultipartUploader](#) 类。

[MultipartUploader](#) 类的 `upload` 方法最适合用于简单分段上传。如果需要暂停并恢复分段上传、在上传期间更改分段大小，或者事先不知道数据大小，请使用低级别 PHP API。有关更多信息，请参阅 [使用适用于分段上传的 AWS PHP 开发工具包 \(低级别 API\) \(p. 180\)](#)。

有关分段上传的更多信息，请参阅[使用分段上传 API 上传对象 \(p. 155\)](#)。有关上传大小小于 5GB 的文件的信息，请参阅[使用适用于 PHP 的 AWS 开发工具包 上传对象 \(p. 153\)](#)。

### 使用高级别分段上传来上传文件

本主题介绍如何使用适用于 PHP 的 AWS 开发工具包中的高级别 `Aws\S3\Model\MultipartUpload\UploadBuilder` 类执行文件分段上传。本主题假定已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作并正确安装了适用于 PHP 的 AWS 开发工具包。

以下 PHP 代码示例将文件上传到 Amazon S3 存储桶。该示例演示如何设置 `MultipartUploader` 对象的参数。

有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

use Aws\Common\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// Prepare the upload parameters.
uploader = new MultipartUploader($s3, '/path/to/large/file.zip', [
    'bucket' => $bucket,
    'key'     => $keyname
]);

// Perform the upload.
try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}" . PHP_EOL;
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

### 相关资源

- [用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 `Aws\S3\S3Client` 类](#)
- [Amazon S3 分段上传](#)
- [适用于 PHP 的 AWS 开发工具包文档](#)

## 使用适用于分段上传的 AWS PHP 开发工具包 (低级别 API)

### 主题

- 使用 PHP 开发工具包低级别 API 以多个段的形式上传文件 (p. 180)
- 使用低级别适用于 PHP 的 AWS 开发工具包 API 列出分段上传 (p. 181)
- 中止分段上传 (p. 182)

适用于 PHP 的 AWS 开发工具包公开了一个低级别 API，与用于分段上传的 Amazon S3 REST API 非常类似 (请参阅[使用适用于分段上传的 REST API \(p. 183\)](#))。当您需要暂停和恢复分段上传、在上传期间更改分段大小或者事先不知道数据大小时，请使用低级 API。只要您没有这些要求，就可使用适用于 PHP 的 AWS 开发工具包高级别抽象 (请参阅[使用 AWS PHP 开发工具包执行分段上传 \(p. 179\)](#))。

### 使用 PHP 开发工具包低级别 API 以多个段的形式上传文件

本主题指南说明如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的低级别 `uploadPart` 方法来以分段上传文件。此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并且已正确安装适用于 PHP 的 AWS 开发工具包。

以下 PHP 示例使用低级别 PHP API 分段上传将文件上传到 Amazon S3 存储桶。有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$filename = '*** Path to and Name of the File to Upload ***';

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

$result = $s3->createMultipartUpload([
    'Bucket'      => $bucket,
    'Key'         => $keyname,
    'StorageClass' => 'REDUCED_REDUNDANCY',
    'ACL'          => 'public-read',
    'Metadata'     => [
        'param1' => 'value 1',
        'param2' => 'value 2',
        'param3' => 'value 3'
    ]
]);
$uploadId = $result['UploadId'];

// Upload the file in parts.
try {
    $file = fopen($filename, 'r');
    $partNumber = 1;
    while (!feof($file)) {
        $result = $s3->uploadPart([
            'Bucket'      => $bucket,
            'Key'         => $keyname,
            'UploadId'   => $uploadId,
            'PartNumber'  => $partNumber,
            'Body'        => fread($file, 5 * 1024 * 1024),
        ]);
    }
}
```

```
$parts['Parts'][$partNumber] = [
    'PartNumber' => $partNumber,
    'ETag' => $result['ETag'],
];
$partNumber++;

    echo "Uploading part {$partNumber} of {$filename}." . PHP_EOL;
}
fclose($file);
} catch (S3Exception $e) {
    $result = $s3->abortMultipartUpload([
        'Bucket' => $bucket,
        'Key'     => $keyname,
        'UploadId' => $uploadId
    ]);

    echo "Upload of {$filename} failed." . PHP_EOL;
}

// Complete the multipart upload.
$result = $s3->completeMultipartUpload([
    'Bucket' => $bucket,
    'Key'     => $keyname,
    'UploadId' => $uploadId,
    'MultipartUpload' => $parts,
]);
$url = $result['Location'];

echo "Uploaded {$filename} to {$url}." . PHP_EOL;
```

## 相关资源

- [用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 Aws\S3\S3Client 类](#)
- [Amazon S3 分段上传](#)
- [适用于 PHP 的 AWS 开发工具包文档](#)

## 使用低级别适用于 PHP 的 AWS 开发工具包 API 列出分段上传

本主题说明如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的低级别 API 类列出存储桶上所有正在进行的分段上传。此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并且已正确安装适用于 PHP 的 AWS 开发工具包。

以下 PHP 示例演示如何列出存储桶上所有正在进行的分段上传。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

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

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

// Retrieve a list of the current multipart uploads.
$result = $s3->listMultipartUploads([
    'Bucket' => $bucket
]);
```

```
// Write the list of uploads to the page.  
print_r($result->toArray());
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 Aws\S3\S3Client 类
- Amazon S3 分段上传
- 适用于 PHP 的 AWS 开发工具包文档

## 中止分段上传

本主题介绍如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类中止正在进行的分段上传。此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并且已正确安装适用于 PHP 的 AWS 开发工具包。

以下 PHP 示例说明如何使用 `abortMultipartUpload()` 方法中止正在进行的分段上传。有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php  
  
require 'vendor/autoload.php';  
  
use Aws\S3\S3Client;  
  
$bucket = '*** Your Bucket Name ***';  
$keyname = '*** Your Object Key ***';  
$uploadId = '*** Upload ID of upload to Abort ***';  
  
$s3 = new S3Client([  
    'version' => 'latest',  
    'region'  => 'us-east-1'  
]);  
  
// Abort the multipart upload.  
$s3->abortMultipartUpload([  
    'Bucket'    => $bucket,  
    'Key'       => $keyname,  
    'UploadId'  => $uploadId,  
]);
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 Aws\S3\S3Client 类
- Amazon S3 分段上传
- 适用于 PHP 的 AWS 开发工具包文档

## 使用适用于 Ruby 的 AWS 开发工具包进行分段上传

适用于 Ruby 的 AWS 开发工具包 版本 3 支持两种 Amazon S3 分段上传方式。对于第一个选项，您可以使用托管文件上传帮助程序。这是将文件上传至存储桶的推荐方法，可提供以下优势：

- 管理大于 15MB 的对象的分段上传。
- 正确打开二进制模式的文件，规避编码问题。
- 在并行上传将较大对象的多个部分时，使用多个线程。

有关更多信息，请参阅 AWS 开发人员博客中的[将文件上传至 Amazon S3](#)。

此外，您还可以直接使用以下分段上传客户端操作：

- [create\\_multipart\\_upload](#) – 启动分段上传，并返回上传 ID。
- [upload\\_part](#) – 上传分段上传中的一部分。
- [upload\\_part\\_copy](#) – 将现有对象作为数据源，并复制其中的数据，上传一部分。
- [complete\\_multipart\\_upload](#) – 整合先前上传的部分，完成分段上传。
- [abort\\_multipart\\_upload](#) – 中止分段上传。

有关更多信息，请参阅 [使用适用于 Ruby 的 AWS 开发工具包 - 版本 3 \(p. 524\)](#)。

## 使用适用于分段上传的 REST API

Amazon Simple Storage Service API Reference 中的以下各部分介绍适用于分段上传的 REST API。

- [开始分段上传](#)
- [上传分段](#)
- [完成分段上传](#)
- [中止分段上传](#)
- [列出分段](#)
- [列出分段上传](#)

您可以使用这些 API 生成您自己的 REST 请求，或者您可以使用我们提供的开发工具包。有关开发工具包的更多信息，请参阅 [分段上传的 API 支持 \(p. 158\)](#)。

## 使用预签名 URL 上传对象

### 主题

- [使用预签名 URL 上传对象 \(AWS SDK for Java\) \(p. 184\)](#)
- [使用预签名 URL 将对象上传到 S3 存储桶 \(适用于 .NET 的 AWS 开发工具包\) \(p. 185\)](#)
- [使用预签名 URL 上传对象 \(适用于 Ruby 的 AWS 开发工具包\) \(p. 186\)](#)

预签名 URL 允许您访问在 URL 中识别的对象，条件是预签名 URL 拥有访问该对象的许可。即，如果您收到用于上传对象的预签名 URL，只要该预签名 URL 的创建者拥有上传该对象所需的许可，您即可上传对象。

默认情况下，所有的对象和存储桶都是私有的。如果您希望您的用户/客户能够将特定对象上传到您的存储桶，但您不要求他们拥有 AWS 安全凭证或许可，那么预签名 URL 将非常有用。创建预签名 URL 时，您必须提供安全凭证，然后指定一个存储桶名称、一个对象键、一个 HTTP 方法（对上传对象执行 PUT 操作）和一个截止日期和时间。预签名 URL 仅在指定的持续时间内有效。

您可以使用 AWS SDK for Java 或适用于 .NET 的 AWS 开发工具包以编程方式生成预签名 URL。如果您使用 Microsoft Visual Studio，还可以使用 AWS Explorer 来生成预签名对象 URL，无需编写任何代码。然后，收到有效预签名 URL 的任何人都可以采用编程的方式上传对象。

有关详细信息，请参阅 [Using Amazon S3 from AWS Explorer](#)。

有关如何安装 AWS Explorer 的说明，请参阅 [使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)。

### Note

具有有效安全凭证的任何人都可以创建预签名 URL。但是，为了成功上传对象，必须由有权执行预签名 URL 所基于的操作的人创建预签名 URL。

## 使用预签名 URL 上传对象 (AWS SDK for Java)

可以使用 AWS SDK for Java 生成一个预签名 URL，您或者从您这里获得了该 URL 的任何人可使用该 URL 将对象上传到 Amazon S3。当使用该 URL 上传对象时，Amazon S3 将在指定存储桶中创建对象。如果存储桶中已存在具有预签名 URL 中指定的相同键的对象，则 Amazon S3 会将现有对象替换为上传的对象。要成功完成上传，必须执行以下操作：

- 在创建 `GeneratePresignedUrlRequest` 和 `HttpURLConnection` 对象时指定 HTTP PUT 谓词。
- 在完成上传后以某种方式与 `HttpURLConnection` 对象交互。以下示例使用 `HttpURLConnection` 对象检查 HTTP 响应代码，从而达到此目的。

### Example

此示例将生成预签名 URL 并使用它将示例数据作为对象上传。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;
import com.amazonaws.services.s3.model.S3Object;

public class GeneratePresignedUrlAndUploadObject {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String objectKey = "*** Object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Set the pre-signed URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = expiration.getTime();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the pre-signed URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest = new
            GeneratePresignedUrlRequest(bucketName, objectKey)
                .withMethod(HttpMethod.PUT)
                .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);

            // Create the connection and use it to upload the new object using the pre-
            signed URL.
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestMethod("PUT");
        }
    }
}
```

```
OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
out.write("This text uploaded as an object via presigned URL.");
out.close();

// Check the HTTP response code. To complete the upload and make the object
available,
// you must interact with the connection object in some way.
connection.getResponseCode();
System.out.println("HTTP response code: " + connection.getResponseCode());

// Check to make sure that the object was uploaded successfully.
S3Object object = s3Client.getObject(bucketName, objectKey);
System.out.println("Object " + object.getKey() + " created in bucket " +
object.getBucketName());
}

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();
}
}
}
```

## 使用预签名 URL 将对象上传到 S3 存储桶 (适用于 .NET 的 AWS 开发工具包)

以下 C# 示例演示如何通过适用于 .NET 的 AWS 开发工具包使用预签名 URL 将对象上传到 S3 存储桶。有关预签名 URL 的更多信息，请参阅[使用预签名 URL 上传对象 \(p. 183\)](#)。

以下示例为特定对象生成预签名 URL 并使用此 URL 上传文件。有关示例与特定版本的适用于 .NET 的 AWS 开发工具包的兼容性的信息以及有关如何创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Net;

namespace Amazon.DocSamples.S3
{
    class UploadObjectUsingPresignedURLTest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string objectKey = "*** provide the name for the uploaded object
***";
        private const string filePath = "*** provide the full path name of the file to
upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
```

```

        var url = GeneratePreSignedURL();
        UploadObject(url);
    }

    private static void UploadObject(string url)
    {
        HttpWebRequest httpRequest = WebRequest.Create(url) as HttpWebRequest;
        httpRequest.Method = "PUT";
        using (Stream dataStream = httpRequest.GetRequestStream())
        {
            var buffer = new byte[8000];
            using (FileStream fileStream = new FileStream(filePath, FileMode.Open,
FileAccess.Read))
            {
                int bytesRead = 0;
                while ((bytesRead = fileStream.Read(buffer, 0, buffer.Length)) > 0)
                {
                    dataStream.Write(buffer, 0, bytesRead);
                }
            }
        }
        HttpWebResponse response = httpRequest.GetResponse() as HttpWebResponse;
    }

    private static string GeneratePreSignedURL()
    {
        var request = new GetPreSignedUrlRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Verb = HttpVerb.PUT,
            Expires = DateTime.Now.AddMinutes(5)
        };

        string url = s3Client.GetPreSignedURL(request);
        return url;
    }
}
}

```

## 更多信息

[适用于 .NET 的 AWS 开发工具包](#)

## 使用预签名 URL 上传对象 (适用于 Ruby 的 AWS 开发工具包)

以下任务引导您通过 Ruby 脚本使用适用于 Ruby 的开发工具包 - 版本 3 和预签名 URL 来上传对象。

### 上传对象 - 适用于 Ruby 的开发工具包 - 版本 3

1	创建 <code>Aws::S3::Resource</code> 类的实例。
2	通过调用 <code>Aws::S3::Resource</code> 类实例的 <code>#bucket[]</code> 和 <code>#object[]</code> 方法，提供存储桶名称和对象键。  通过创建 <code>URI</code> 类的实例生成预签名 URL，然后使用它分析 <code>Aws::S3::Resource</code> 类实例的 <code>.presigned_url</code> 方法。您必须将 <code>:put</code> 指定为 <code>.presigned_url</code> 的参数，如果需要上传对象，必须向 <code>Net::HTTP::Session#send_request</code> 指定 <code>PUT</code> 。
3	任何拥有预签名 URL 的人都可以上传对象。  上传将创建对象或将任何现有的对象替换为预签名 URL 中指定的相同键值。

以下 Ruby 代码示例演示使用适用于 Ruby 的开发工具包版本 3 实现的上述任务。

#### Example

```
#Uploading an object using a pre-signed URL for ### Ruby ##### - Version 3.

require 'aws-sdk-s3'
require 'net/http'

s3 = Aws::S3::Resource.new(region:'us-west-2')

obj = s3.bucket('BucketName').object('KeyName')
# Replace BucketName with the name of your bucket.
# Replace KeyName with the name of the object you are creating or replacing.

url = URI.parse(obj.presigned_url(:put))

body = "Hello World!"
# This is the contents of your object. In this case, it's a simple string.

Net::HTTP.start(url.host) do |http|
  http.send_request("PUT", url.request_uri, body, {
    # This is required, or Net::HTTP will add a default unsigned content-type.
    "content-type" => "",
  })
end

puts obj.get.body.read
# This will print out the contents of your object to the terminal window.
```

## 复制对象

### 主题

- [相关资源 \(p. 188\)](#)
- [复制单个操作中的数据元 \(p. 188\)](#)
- [使用分段上传 API 复制对象 \(p. 193\)](#)

复制操作将创建已存储在 Amazon S3 中的对象的副本。在单个原子操作中，您可以创建最大 5 GB 的对象副本。但是，对于复制大于 5 GB 的对象，您必须使用分段上传 API。通过使用 copy 操作，您可以：

- 创建对象的其他副本
- 通过复制对象并删除原始对象来重命名它们。
- 在不同的 Amazon S3 位置之间 (例如，us-west-1 和 EU) 移动对象
- 更改对象元数据

每个 Amazon S3 对象都带有元数据。它是一组名称值对。您可以在上传对象元数据时对其进行设置。上传对象后，您将无法修改对象元数据。修改对象元数据的唯一方式是创建对象的副本并设置元数据。在复制操作中，设置与源和目标相同的对象。

每个对象都带有元数据。有些是系统元数据，而另外一些则是用户定义的元数据。用户可以控制某些系统元数据，例如，用于对象的存储类别配置和配置服务器端加密。复制对象时，还会复制用户控制的系统元数据和用户定义的元数据。Amazon S3 将重置系统控制的元数据。例如，在复制对象时，Amazon S3 将重置已复制对象的创建日期。在复制请求中，您无需设置这些值。

复制对象时，您可能会决定更新某些元数据值。例如，如果您的源对象被配置为使用标准存储，您可能会为对象复制选择低冗余存储。您可能还会决定更改源对象上某些用户定义的元数据值。请注意，如果您选择在

复制期间更新任意对象的用户可配置元数据 (系统或用户定义的元数据) , 则必须显式指定请求中源对象上存在的所有用户可配置的元数据 , 即使您只更改其中一个元数据值也是如此。

有关对象元数据的详细信息 , 请参阅 [对象键和元数据 \(p. 86\)](#)。

Note

复制不同位置上的对象将产生带宽费用。

Note

如果源对象在 Amazon Glacier 中 (对象的存储类别为 GLACIER) 进行存档 , 则必须首先还原临时副本 , 然后才能将对象复制到另一个存储桶。有关对象存档的更多信息 , 请参阅 [转换为 GLACIER 存储类 \(对象存档\) \(p. 107\)](#)。

在复制对象时 , 您可以请求 Amazon S3 保存使用 AWS Key Management Service (KMS) 加密密钥、Amazon S3 托管加密密钥或客户提供的加密密钥加密的目标对象。因此 , 您必须在请求中指定加密信息。如果复制源是通过客户提供的密钥使用服务器端加密存储在 Amazon S3 中的对象 , 则您需要在请求中提供加密信息 , 以便 Amazon S3 可以解密对象进行复制。有关更多信息 , 请参阅 [使用加密保护数据 \(p. 345\)](#)。

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 复制单个操作中的数据元

本节中的示例向您展示了如何复制单个操作中大于 5 GB 的对象。对于复制大于 5 GB 的对象 , 您必须使用分段上传 API。有关更多信息 , 请参阅 [使用分段上传 API 复制对象 \(p. 193\)](#)。

主题

- 使用 AWS SDK for Java 复制对象 (p. 188)
- 使用适用于 .NET 的 AWS 开发工具包在单个操作中复制 Amazon S3 对象 (p. 189)
- 使用适用于 PHP 的 AWS 开发工具包 复制对象 (p. 190)
- 使用适用于 Ruby 的 AWS 开发工具包 复制对象 (p. 191)
- 使用 REST API 复制对象 (p. 192)

## 使用 AWS SDK for Java 复制对象

Example

以下示例演示如何使用 AWS SDK for Java 复制 Amazon S3 中的对象。有关创建和测试有效示例的说明 , 请参阅 [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

public class CopyObjectSingleOperation {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
```

```
String sourceKey = "*** Source object key ***";
String destinationKey = "*** Destination object key ***";

try {
    AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(clientRegion)
        .build();

    // Copy the object into a new object in the same bucket.
    CopyObjectRequest copyObjRequest = new CopyObjectRequest(bucketName, sourceKey,
bucketName, destinationKey);
    s3Client.copyObject(copyObjRequest);
}
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();
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包在单个操作中复制 Amazon S3 对象

以下 C# 示例演示如何使用高级别适用于 .NET 的 AWS 开发工具包在单个操作中复制最大为 5 GB 的对象。对于大于 5 GB 的对象，请使用[使用适用于 .NET 的 AWS 开发工具包分段上传 API 复制 Amazon S3 对象 \(p. 195\)](#)中所述的分段上传复制示例。

此示例将创建最大为 5 GB 的对象的副本。要获得示例与特定版本的适用于 .NET 的 AWS 开发工具包的兼容性的信息以及有关如何创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class CopyObjectTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with source
object ***";
        private const string destinationBucket = "*** provide the name of the bucket to
copy the object to ***";
        private const string objectKey = "*** provide the name of object to copy ***";
        private const string destObjectKey = "*** provide the destination object key 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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
        }
    }
}
```

```
        CopyingObjectAsync().Wait();
    }

    private static async Task CopyingObjectAsync()
    {
        try
        {
            CopyObjectRequest request = new CopyObjectRequest
            {
                SourceBucket = sourceBucket,
                SourceKey = objectKey,
                DestinationBucket = destinationBucket,
                DestinationKey = destObjectKey
            };
            CopyObjectResponse response = await s3Client.CopyObjectAsync(request);
        }
        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);
        }
    }
}
```

## 更多信息

### 适用于 .NET 的 AWS 开发工具包

### 使用 适用于 PHP 的 AWS 开发工具包 复制对象

此主题将指导您使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类，将 Amazon S3 中的单个对象和多个对象从一个存储桶复制到另一个存储桶，或者复制到同一存储桶中。

此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并正确安装了适用于 PHP 的 AWS 开发工具包。

以下任务将引导您使用 PHP 开发工具包类复制已存储在 Amazon S3 中的对象。

以下任务将引导您使用 PHP 类在 Amazon S3 中创建对象的多个副本。

## 复制对象

1	使用 <code>Aws\S3\S3Client</code> 类构造函数创建 Amazon S3 客户端的实例。
2	要创建对象的多个副本，请执行对 Amazon S3 客户端 <code>getCommand()</code> 方法的批量调用，该方法是从 <code>Aws\CommandInterface</code> 类继承的。您提供 <code>CopyObject</code> 命令作为第一个参数，提供包含源存储桶、源键名称、目标存储桶和目标键名称的数组作为第二个参数。

### Example 复制 Amazon S3 中的对象

以下 PHP 示例演示使用 `copyObject()` 方法复制 Amazon S3 中的单个对象，使用 `CopyObject` 方法对 `getCommand()` 进行批量调用来创建对象的多个副本。

```
<?php
```

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

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';
$targetBucket = '*** Your Target Bucket Name ***';

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

// Copy an object.
$s3->copyObject([
    'Bucket'      => $targetBucket,
    'Key'         => "{$sourceKeyname}-copy",
    'CopySource'  => "{$sourceBucket}/{$sourceKeyname}",
]);

// Perform a batch of CopyObject operations.
$batch = array();
for ($i = 1; $i <= 3; $i++) {
    $batch[] = $s3->getCommand('CopyObject', [
        'Bucket'      => $targetBucket,
        'Key'         => "{$targetKeyname}-{$i}",
        'CopySource'  => "{$sourceBucket}/{$sourceKeyname}",
    ]);
}
try {
    $succeeded = $s3->execute($batch);
    $failed = array();
} catch (CommandTransferException $e) {
    $succeeded = $e->getSuccessfulCommands();
    echo "Failed Commands:" . PHP_EOL;
    foreach ($e->getFailedCommands() as $failedCommand) {
        echo $e->getExceptionForFailedCommand($failedCommand)->getMessage() . PHP_EOL;
    }
}
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 `Aws\S3\S3Client` 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用适用于 Ruby 的 AWS 开发工具包 复制对象

以下任务将引导您使用 Ruby 类将 Amazon S3 中的对象从一个存储桶复制到另一存储桶，或者复制同一存储桶中的对象。

### 复制对象

1	对适用于 Ruby 的 AWS 开发工具包 版本 3 使用 Amazon S3 模块化 Gem 需要“aws-sdk-s3”并提供您的 AWS 凭证。有关如何提供您的凭证的更多信息，请参阅 <a href="#">使用 AWS 账户或 IAM 用户凭证进行请求 (p. 16)</a> 。
2	提供源存储桶名称、源键名称、目标存储桶名称和目标键等请求信息。

下面的 Ruby 代码示例通过使用 `#copy_object` 方法将对象从一个存储桶复制到另一个存储桶，演示了上述任务。

### Example

```
require 'aws-sdk-s3'

source_bucket_name = '*** Provide bucket name ***'
target_bucket_name = '*** Provide bucket name ***'
source_key = '*** Provide source key ***'
target_key = '*** Provide target key ***'

s3 = Aws::S3::Client.new(region: 'us-west-2')
s3.copy_object({bucket: target_bucket_name, copy_source: source_bucket_name + '/' +
source_key, key: target_key})

puts "Copying file #{source_key} to #{target_key}."
```

### 使用 REST API 复制对象

本示例描述了如何使用 REST 复制对象。有关 REST API 的更多信息，请参阅 [PUT Object \(Copy\)](#)。

本示例将 pacific 存储桶中的 flotsam 对象复制到 atlantic 存储桶的 jetsam 对象，同时保留其元数据。

```
PUT /jetsam HTTP/1.1
Host: atlantic.s3.amazonaws.com
x-amz-copy-source: /pacific/flotsam
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ENoSbxYByFA0UGLZUqJN5EUnLDg=
Date: Wed, 20 Feb 2008 22:12:21 +0000
```

将从以下信息生成签名。

```
PUT\r\n
\r\n
\r\n
Wed, 20 Feb 2008 22:12:21 +0000\r\n

x-amz-copy-source:/pacific/flotsam\r\n
/atlantic/jetsam
```

Amazon S3 将返回以下响应来指定数据元的 ETag 及其上次修改的时间。

```
HTTP/1.1 200 OK
x-amz-id-2: Vyaxt7qEbzb34BnSu5hctyyNSlHTYZFMWK4FtzO+ix8JQnyaLdTshL0KxatbaOzt
x-amz-request-id: 6B13C3C5B34AF333
Date: Wed, 20 Feb 2008 22:13:01 +0000

Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: AmazonS3
<?xml version="1.0" encoding="UTF-8"?>

<CopyObjectResult>
  <LastModified>2008-02-20T22:13:01</LastModified>
  <ETag>"7e9c608af58950deeb370c98608ed097"</ETag>
</CopyObjectResult>
```

## 使用分段上传 API 复制对象

本节中的示例将向您展示如何使用分段上传 API 复制大于 5 GB 的对象。您可以在单个操作中复制小于 5 GB 的对象。有关更多信息，请参阅 [复制单个操作中的数据元 \(p. 188\)](#)。

### 主题

- [使用 AWS SDK for Java 分段上传 API 复制对象 \(p. 193\)](#)
- [使用适用于 .NET 的 AWS 开发工具包分段上传 API 复制 Amazon S3 对象 \(p. 195\)](#)
- [使用 REST 分段上传 API 复制对象 \(p. 197\)](#)

### 使用AWS SDK for Java分段上传 API 复制对象

要使用AWS SDK for Java复制大于 5 GB 的 Amazon S3 对象，请使用低级别 Java API。对于小于 5 GB 的对象，请使用[使用 AWS SDK for Java 复制对象 \(p. 188\)](#)中所述的单个操作复制。

要使用低级别 Java API 复制对象，请执行以下操作：

- 通过执行 `AmazonS3Client.initiateMultipartUpload()` 方法启动分段上传。
- 从 `AmazonS3Client.initiateMultipartUpload()` 方法返回的响应对象保存上传 ID。您为每个分段上传操作提供此上传 ID。
- 复制所有段。对于需要复制的每个段，创建一个 `CopyPartRequest` 类的新实例。提供段信息，包括源和目标存储桶名称、源和目标对象键、上传 ID、段的第一个字节和最后一个字节的位置以及段编号。
- 保存 `AmazonS3Client.copyPart()` 方法调用的响应。每个响应均包括 ETag 值和已上传分段的段编号。您需要此信息才能完成分段上传。
- 调用 `AmazonS3Client.completeMultipartUpload()` 方法以完成复制操作。

### Example

以下示例说明如何使用 Amazon S3 低级别 Java API 执行分段复制。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.*;
import com.amazonaws.services.s3.model.*;

public class LowLevelMultipartCopy {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String sourceBucketName = "*** Source bucket name ***";
        String sourceObjectKey = "*** Source object key ***";
        String destBucketName = "*** Target bucket name ***";
        String destObjectKey = "*** Target object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Initiate the multipart upload.
```

```
    InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(destBucketName, destObjectKey);
    InitiateMultipartUploadResult initResult =
s3Client.initiateMultipartUpload(initRequest);

    // Get the object size to track the end of the copy operation.
    GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest(sourceBucketName, sourceObjectKey);
    ObjectMetadata metaDataResult = s3Client.getObjectMetadata(metadataRequest);
    long objectSize = metaDataResult.getContentLength();

    // Copy the object using 5 MB parts.
    long partSize = 5 * 1024 * 1024;
    long bytePosition = 0;
    int partNum = 1;
    List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
    while (bytePosition < objectSize) {
        // The last part might be smaller than partSize, so check to make sure
        // that lastByte isn't beyond the end of the object.
        long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

        // Copy this part.
        CopyPartRequest copyRequest = new CopyPartRequest()
            .withSourceBucketName(sourceBucketName)
            .withSourceKey(sourceObjectKey)
            .withDestinationBucketName(destBucketName)
            .withDestinationKey(destObjectKey)
            .withUploadId(initResult.getUploadId())
            .withFirstByte(bytePosition)
            .withLastByte(lastByte)
            .withPartNumber(partNum++);
        copyResponses.add(s3Client.copyPart(copyRequest));
        bytePosition += partSize;
    }

    // Complete the upload request to concatenate all uploaded parts and make the
copied object available.
    CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
    destBucketName,
    destObjectKey,

initResult.getUploadId(),

getETags(copyResponses));
    s3Client.completeMultipartUpload(completeRequest);
    System.out.println("Multipart copy complete.");
}
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();
}
}

// This is a helper function to construct a list of ETags.
private static List<PartETag> getETags(List<CopyPartResult> responses) {
    List<PartETag> etags = new ArrayList<PartETag>();
    for (CopyPartResult response : responses) {
        etags.add(new PartETag(response.getPartNumber(), response.getETag()));
    }
}
```

```
        return etags;
    }
}
```

## 使用适用于 .NET 的 AWS 开发工具包分段上传 API 复制 Amazon S3 对象

以下 C# 示例说明如何使用 适用于 .NET 的 AWS 开发工具包 将大于 5 GB 的 Amazon S3 对象从一个源位置 复制到另一个源位置，例如从一个存储桶复制到另一个存储桶。要复制小于 5 GB 的对象，请使用[使用适用于 .NET 的 AWS 开发工具包在单个操作中复制 Amazon S3 对象 \(p. 189\)](#)中所述的单个操作复制过程。有关 Amazon S3 分段上传的更多信息，请参阅[分段上传概述 \(p. 155\)](#)。

此示例说明如何使用 适用于 .NET 的 AWS 开发工具包 分段上传 API 将大于 5 GB 的 Amazon S3 对象从一个 S3 存储桶复制到另一个存储桶。有关开发工具包兼容性的信息以及有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectUsingMPUapiTest
    {
        private const string sourceBucket = "**** provide the name of the bucket with source object ****";
        private const string targetBucket = "**** provide the name of the bucket to copy the object to ****";
        private const string sourceObjectKey = "**** provide the name of object to copy ****";
        private const string targetObjectKey = "**** provide the name of the object copy ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            MPUCopyObjectAsync().Wait();
        }
        private static async Task MPUCopyObjectAsync()
        {
            // Create a list to store the upload part responses.
            List<UploadPartResponse> uploadResponses = new List<UploadPartResponse>();
            List<CopyPartResponse> copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest =
                new InitiateMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey
                };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =

```

```
        await s3Client.InitiateMultipartUploadAsync(initiateRequest);

        // Save the upload ID.
        String uploadId = initResponse.UploadId;

        try
        {
            // Get the size of the object.
            GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest
            {
                BucketName = sourceBucket,
                Key = sourceObjectKey
            };

            GetObjectMetadataResponse metadataResponse =
                await s3Client.GetObjectMetadataAsync(metadataRequest);
            long objectSize = metadataResponse.ContentLength; // Length in bytes.

            // Copy the parts.
            long partSize = 5 * (long) Math.Pow(2, 20); // Part size is 5 MB.

            long bytePosition = 0;
            for (int i = 1; bytePosition < objectSize; i++)
            {
                CopyPartRequest 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 s3Client.CopyPartAsync(copyRequest));

                bytePosition += partSize;
            }

            // Set up to complete the copy.
            CompleteMultipartUploadRequest completeRequest =
new CompleteMultipartUploadRequest
{
    BucketName = targetBucket,
    Key = targetObjectKey,
    UploadId = initResponse.UploadId
};
            completeRequest.AddPartETags(copyResponses);

            // Complete the copy.
            CompleteMultipartUploadResponse completeUploadResponse =
                await s3Client.CompleteMultipartUploadAsync(completeRequest);
        }
        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);
        }
    }
}
```

```
    }  
}  
}
```

## 更多信息

[适用于 .NET 的 AWS 开发工具包](#)

## 使用 REST 分段上传 API 复制对象

Amazon Simple Storage Service API Reference 中的以下各部分介绍适用于分段上传的 REST API。使用上传分段 (复制) API 复制现有的对象，并通过在请求中添加 `x-amz-copy-source` 请求标头指定元数据源。

- [开始分段上传](#)
- [上传分段](#)
- [上传分段 \(复制\)](#)
- [完成分段上传](#)
- [中止分段上传](#)
- [列出分段](#)
- [列出分段上传](#)

您可以使用这些 API 生成您自己的 REST 请求，或者您可以使用我们提供的开发工具包。有关开发工具包的更多信息，请参阅 [分段上传的 API 支持 \(p. 158\)](#)。

## 列出对象键

可以按前缀列出键。通过为相关键的名称选择一个通用前缀，并使用用于划定层级结构的特殊字符来标记这些键，您可以使用列表操作来按层级结构选择和浏览键。这类似于在文件系统的目录中存储文件的方式。

Amazon S3 公开了列表操作，允许您列出包含在存储桶中的键。将按存储桶和前缀选择用于列表的键。例如，假设一个存储桶的名称为“dictionary”，它为每个英语词汇包含了一个键值。您可能会创建一个调用来列出该存储桶中以字母“q”开头的存储桶。列表结果始终以 UTF-8 二进制顺序返回。

SOAP 和 REST 列表操作将返回一个 XML 文档，其中包含匹配键值的名称和有关由每个键值识别的对象的信息。

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

出于列表目的，可以根据通用前缀收拢那些共享一个前缀并且以特定分隔符终结的键组。这允许应用程序按层级结构组织和浏览键值，与您在文件系统的目录中组织文件的方式相同。例如，要扩展字典存储桶以包含除英语单词外的更多内容，您可以使用语言和分隔符为每个单词添加前缀（例如，“French/logical”）从而构成键值。使用此命名方案和层级列表功能，您可以检索仅包含法语词汇的列表。您也可以浏览可用语言的顶级列表，而无需循环浏览所有按字典顺序排列的预键。

有关列表的此方面的更多信息，请参阅 [使用前缀和分隔符按层次结构列出键 \(p. 198\)](#)。

### 列表执行效率

存储桶中键的总数量，以及前缀、标记、最大键或分隔符参数的存在或缺失都不会影响列表的性能。有关提升存储桶整体性能的信息（包括列表操作），请参阅 [请求速率和性能指南 \(p. 480\)](#)。

## 循环访问多页结果

由于存储桶可以包含几乎无限数量的键，列表查询的完整结果可能会非常大。为了管理大型结果集，Amazon S3 API 支持分页，以将它们分割为多个响应。每个列出键响应将返回一个拥有多达 1000 个键

的页面，同时使用指示器来指示响应是否存在截断。您可以发送一系列的列出键请求，直到您收到了所有的键。AWS 开发工具包包装库提供相同的分页。

以下 Java 和 .NET 开发工具包示例说明了如何在存储桶中列出键时使用分页：

- 使用 AWS SDK for Java 列出键 (p. 199)
- 使用适用于 .NET 的 AWS 开发工具包 列出键 (p. 200)

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 使用前缀和分隔符按层次结构列出键

前缀和分隔符参数将限制列表操作返回的结果类型。前缀将结果限制为仅以特定前缀开头的键，并且分隔符将导致列表收拢在单个摘要列表结果中共享一个通用前缀的所有键。

前缀和分隔符参数的目的是帮助您按层级结构组织，然后浏览您的键。要执行此操作，首先为您的存储桶选取一个分隔符，例如，斜杠 (/)，它不会出现在任何预期的键名称中。接下来，通过串联所有包含层次结构的级别，并使用分隔符分隔每个级别来构建您的键名称。

例如，如果您正在存储关于城市的信息，您可以按大陆、按国家/区域，然后按省份或州来自然地组织他们。因为这些名称通常不包含标点符号，您可以选择斜杠 (/) 作为分隔符。下面的示例使用斜杠 (/) 分隔符。

- Europe/France/Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- North America/USA/Washington/Bellevue
- North America/USA/Washington/Seattle

如果您采用此方式来存储世界上每个城市的数据，那么管理平面键命名空间会很困难。通过使用带列表操作的 Prefix 和 Delimiter，您可以使用已创建的层次结构来列出您的数据。例如，要列出美国的所有州，请设置 Delimiter="/" 和 Prefix='North America/USA/'。要列出您拥有数据的所有加拿大省份，请设置 Delimiter="/" 和 Prefix='North America/Canada/'。

带分隔符的列表请求允许您仅浏览一个级别的层次结构、跳过或总结嵌套在更深级别的键（可能是数百万）。例如，假设您拥有存储桶 (ExampleBucket) 和以下键。

```
sample.jpg
photos/2006/January/sample.jpg
photos/2006/February/sample2.jpg
photos/2006/February/sample3.jpg
photos/2006/February/sample4.jpg
```

示例存储桶仅拥有根级 sample.jpg 对象。要仅列出存储桶中的根级对象，您需要向存储桶发送带 "/" 分隔符的 GET 请求。作为响应，Amazon S3 将返回 sample.jpg 对象键，因为它不包含 "/" 分隔符。所有其他键都包含分隔符。Amazon S3 将组合这些键，并在指定的分隔符首次出现时，返回带前缀值 photos/ 的单个 CommonPrefixes 元素（它是这些键开头的子字符串）。

### Example

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Name>ExampleBucket</Name>
```

```
<Prefix></Prefix>
<Marker></Marker>
<MaxKeys>1000</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>false</IsTruncated>
<Contents>
    <Key>sample.jpg</Key>
    <LastModified>2011-07-24T19:39:30.000Z</LastModified>
    <ETag>&quot;d1a7fb5eab1c16cb4f7cf341cf188c3d&quot;</ETag>
    <Size>6</Size>
    <Owner>
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeefbf76c078efc7c6caea54ba06a</ID>
        <DisplayName>displayname</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
</Contents>
<CommonPrefixes>
    <Prefix>photos/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

## 使用 AWS SDK for Java 列出键

### Example

下面的示例将列出存储桶中的对象键。该示例使用分页来检索一组对象键。如果有多个键要在第一页后要返回，Amazon S3 将在响应中包含一个延续令牌。该示例在后续请求中使用延续令牌来提取下一组对象键。

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListKeys {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
ListObjectsV2Request().withBucketName(bucketName).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);
```

```
        for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
            System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
            objectSummary.getSize());
        }
        // If there are more than maxKeys keys in the bucket, get a continuation
        token
        // and list the next objects.
        String token = result.getNextContinuationToken();
        System.out.println("Next Continuation Token: " + token);
        req.setContinuationToken(token);
    } while (result.isTruncated());
}
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();
}
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包 列出键

### Example

下面的 C# 示例将列出存储桶的对象键。在该示例中，我们使用分页来检索一组对象键。如果有多个键要返回，Amazon S3 将在响应中包含一个延续令牌。该代码在后续请求中使用延续令牌来提取下一组对象键。

有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class ListObjectsTest
    {
        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()
        {
            client = new AmazonS3Client(bucketRegion);
            ListingObjectsAsync().Wait();
        }

        static async Task ListingObjectsAsync()
        {
```

```
try
{
    ListObjectsV2Request 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);
}
catch (AmazonS3Exception amazonS3Exception)
{
    Console.WriteLine("S3 error occurred. Exception: " +
amazonS3Exception.ToString());
    Console.ReadKey();
}
catch (Exception e)
{
    Console.WriteLine("Exception: " + e.ToString());
    Console.ReadKey();
}
}
```

## 使用适用于 PHP 的 AWS 开发工具包 列出键

此主题将指导您使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类列出 Amazon S3 存储桶中包含的对象键。

此主题假定您已按照 [使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#) 的说明执行操作，并正确安装了适用于 PHP 的 AWS 开发工具包。

要使用适用于 PHP 的 AWS 开发工具包 列出存储桶中包含的对象键，首先必须列出该存储桶中包含的对象，然后从列出的每个对象中提取键。在列出存储桶中的对象时，可选择使用低级别 [Aws\S3\S3Client::listObjects\(\)](#) 方法或高级别 [Aws\ResultPaginator](#) 类。

低级别的 `listObjects()` 方法将映射到底层 Amazon S3 REST API。每个 `listObjects()` 请求均返回最多有 1000 个对象的页面。如果您的存储桶中有超过 1000 个对象，则将截断您的响应，并且您需要发送其他 `listObjects()` 请求，以检索下一组 1000 个对象。

您可以使用高级别 `ListObjects` 分页工具使列出存储桶中包含的对象的任务变得更轻松。要使用 `ListObjects` 分页工具创建对象列表，请执行从 [Aws\AwsClientInterface](#) 类继承的 Amazon S3 客户端 `getPaginator()` 方法，将 `ListObjects` 作为第一个参数，将包含从指定存储桶返回的对象的数组作为第二个参数。当作为 `ListObjects` 分页工具使用时，`getPaginator()` 方法将返回指定存储桶中包含的所有对象。不存在 1000 个对象的限制，因此，您无需担心响应是否被截断。

以下任务将引导您使用 PHP Amazon S3 客户端方法列出您能够从中列出对象键的存储桶中包含的对象。

### Example 列出对象键

以下 PHP 示例演示如何列出指定存储桶中的键。它演示如何使用高级别 `getIterator()` 方法列出存储桶中的对象，然后如何从该列表的每个对象中提取键。它还演示如何使用低级别 `listObjects()` 方法列出存储桶中的对象，然后如何从返回的列表的每个对象中提取键。有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

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

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

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

// Use the high-level iterators (returns ALL of your objects).
try {
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}

// Use the plain API (returns ONLY up to 1000 of your objects).
try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;
    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

### 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 `Aws\S3\S3Client` 类
- 分页工具
- 适用于 PHP 的 AWS 开发工具包文档

## 使用 REST API 列出键

您可以使用 AWS 开发工具包来列出存储桶中的对象键。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。您可以发送 GET 请求来返回存储桶中的某些或所有对象，或者您也可以使用选择条件来返回存储桶中对象的子集。有关详细信息，请参阅 [GET Bucket \(List Objects\) 版本 2](#)。

## 删除对象

### 主题

- [从受版本控制的存储桶中删除对象 \(p. 203\)](#)
- [从启用了 MFA 的存储桶中删除对象 \(p. 203\)](#)
- [相关资源 \(p. 204\)](#)
- [每个请求删除一个对象。 \(p. 204\)](#)
- [每个请求删除多个对象 \(p. 209\)](#)

您可以直接从 Amazon S3 删除一个或多个对象。删除对象时，您可以使用以下选项：

- **删除单个对象** – Amazon S3 提供了删除 API，使您能够删除单个 HTTP 请求中的一个对象。
- **删除多个对象** – Amazon S3 还提供了多对象删除 API，使您能够在单个 HTTP 请求中删除最多 1000 个对象。

从未受版本控制的存储桶删除对象时，您只需提供对象键名；从受版本控制的存储桶删除对象时，您可以选择提供对象的版本 ID 以删除特定版本的对象。

### 从受版本控制的存储桶中删除对象

如果您的存储桶受版本控制，则存储桶中可能存在同一对象的多个版本。使用受版本控制的存储桶时，删除 API 将提供以下选项：

- 指定不受版本控制的删除请求 – 即您仅指定对象的键而不是版本 ID。在此情况下，Amazon S3 将创建一个删除标记并在响应中返回版本 ID。这将使您的对象从存储桶中消失。有关对象版本控制和删除标记概念的信息，请参阅 [对象版本控制 \(p. 94\)](#)。
- 指定受版本控制的删除请求 – 即不仅可以指定键，还可以指定版本 ID。在此情况下，可能会出现以下两种结果：
  - 如果版本 ID 映射到特定的对象版本，则 Amazon S3 将删除该特定版本的对象。
  - 如果版本 ID 映射到对象的删除标记，则 Amazon S3 将删除该删除标记。这将使您的对象重新出现在存储桶中。

### 从启用了 MFA 的存储桶中删除对象

从启用了多重身份认证 (MFA) 的存储桶中删除对象时，请注意以下内容：

- 如果您提供了无效的 MFA 令牌，请求将始终失败。
- 如果您拥有一个启用了 MFA 的存储桶，并且发送了一个受版本控制的删除请求（您提供了对象键和版本 ID），若您不能提供有效的 MFA 令牌，请求将失败。此外，对启用了 MFA 的存储桶使用多对象删除 API 时，如果任意的删除是受版本控制的删除请求（即您指定了对象键和版本 ID），若您不能提供有效的 MFA 令牌，则整个请求将失败。

另一方面，在下面的案例中请求将成功：

- 如果您有一个启用了 MFA 的存储桶，并且发送了一个不受版本控制的删除请求（您不打算删除受版本控制的对象），即使您不提供 MFA 令牌，删除也会成功。
- 如果您拥有一个对象删除请求并且仅指定了要从启用了 MFA 的存储桶删除的不受版本控制的对象，即使您不提供 MFA 令牌，删除也会成功。

有关 MFA 删除的信息，请参阅 [MFA 删除 \(p. 381\)](#)。

## 相关资源

- 使用 AWS 开发工具包、CLI 和 Explorer (p. 518)

## 每个请求删除一个对象。

### 主题

- 使用 AWS SDK for Java 删除对象 (p. 204)
- 使用适用于 .NET 的 AWS 开发工具包删除对象 (p. 206)
- 使用适用于 PHP 的 AWS 开发工具包删除对象 (p. 208)
- 使用 REST API 删除对象 (p. 209)

要通过每个请求删除一个对象，请使用 DELETE API (请参阅[删除对象](#))。要了解对象删除的更多信息，请参阅[删除对象 \(p. 203\)](#)。

您可以直接使用 REST API 或使用 AWS 开发工具包提供的可简化应用程序开发的包装程序库。

### 使用 AWS SDK for Java 删除对象

您可以从存储桶删除对象。如果已对存储桶启用版本控制，您将获得以下选项：

- 通过指定版本 ID 来删除特定对象版本。
- 删除对象而不指定版本 ID，在这种情况下，S3 将向对象添加一个删除标记。

有关版本控制的更多信息，请参阅[对象版本控制 \(p. 94\)](#)。

#### Example 示例 1：删除对象 (不受版本控制的存储桶)

下面的示例将从存储桶删除对象。该示例假定存储桶不受版本控制且对象没有任何版本 ID。在删除请求中，您仅指定对象键而不是版本 ID。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

public class DeleteObjectNonVersionedBucket {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
        }
        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();
}
}
```

### Example 示例 2：删除对象 (受版本控制的存储桶)

以下示例将删除受版本控制的存储桶中的对象。该示例通过指定对象键名称和版本 ID 来删除特定对象版本。本示例执行以下操作：

1. 向存储桶添加示例对象。Amazon S3 将返回新添加对象的版本 ID。该示例将在删除请求中使用此版本 ID。
2. 通过指定对象键名称和版本 ID 来删除对象版本。如果对象没有其他版本，则 Amazon S3 将完全删除对象。否则，Amazon S3 仅删除指定版本。

#### Note

可以通过发送 `ListVersions` 请求来获取对象的版本 ID。

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.DeleteVersionRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

public class DeleteObjectVersionEnabledBucket {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String keyName = "**** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Check to ensure that the bucket is versioning-enabled.
            String bucketVersionStatus =
                s3Client.getBucketVersioningConfiguration(bucketName).getStatus();
            if(!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED)) {
                System.out.printf("Bucket %s is not versioning-enabled.", bucketName);
            }
            else {
                // Add an object.
                PutObjectResult putResult = s3Client.putObject(bucketName, keyName, "Sample
content for deletion example.");
                System.out.printf("Object %s added to bucket %s\n", keyName, bucketName);
            }
        }
    }
}
```

```
// Delete the version of the object that we just created.  
System.out.println("Deleting versioned object " + keyName);  
s3Client.deleteVersion(new DeleteVersionRequest(bucketName, keyName,  
putResult.getVersionId()));  
System.out.printf("Object %s, version %s deleted\n", keyName,  
putResult.getVersionId());  
}  
}  
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();  
}  
}  
}
```

## 使用适用于 .NET 的 AWS 开发工具包删除对象

当您删除不受版本控制的存储桶中的某个对象时，该对象将被移除。如果已对存储桶启用版本控制，您将获得以下选项：

- 通过指定版本 ID 来删除对象的特定版本。
- 在不指定版本 ID 的情况下删除对象。Amazon S3 将添加一个删除标记。有关删除标记的更多信息，请参阅[对象版本控制 \(p. 94\)](#)。

以下示例演示如何删除受版本控制和不受版本控制的存储桶中的对象。有关版本控制的更多信息，请参阅[对象版本控制 \(p. 94\)](#)。

### Example 删除不受版本控制的存储桶中的对象

以下 C# 示例将删除不受版本控制的存储桶中的对象。该示例假定对象没有版本 ID，因此您未指定版本 ID。您仅指定了对象键。有关如何创建和测试有效示例的信息，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-  
developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using Amazon.S3;  
using Amazon.S3.Model;  
using System;  
using System.Threading.Tasks;  
  
namespace Amazon.DocSamples.S3  
{  
    class DeleteObjectNonVersionedBucketTest  
    {  
        private const string bucketName = "*** bucket name ***";  
        private const string keyName = "*** object key ***";  
        // Specify your bucket region (an example region is shown).  
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;  
        private static IAmazonS3 client;  
  
        public static void Main()  
        {  
            client = new AmazonS3Client(bucketRegion);  
            DeleteObjectNonVersionedBucketAsync().Wait();  
        }  
    }  
}
```

```

        }

        private static async Task DeleteObjectNonVersionedBucketAsync()
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(deleteObjectRequest);
            }
            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);
            }
        }
    }
}

```

### Example 删除受版本控制的存储桶中的对象

以下 C# 示例将删除受版本控制的存储桶中的对象。它将通过指定对象键名称和版本 ID 来删除对象的特定版本。

代码将执行以下任务：

1. 对指定的存储桶启用版本控制 (如果已启用版本控制，则此操作无效)。
2. 向存储桶添加示例对象。作为响应，Amazon S3 将返回新添加的对象的版本 ID。该示例将在删除请求中使用此版本 ID。
3. 通过指定对象键名称和版本 ID 来删除示例对象。

#### Note

还可以通过发送 `ListVersions` 请求来获取对象的版本 ID：

```

var listResponse = client.ListVersions(new ListVersionsRequest { BucketName =
bucketName, Prefix = keyName });

```

有关如何创建和测试有效示例的信息，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```

// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{

```

```
class DeleteObjectVersion
{
    private const string bucketName = "*** versioning-enabled bucket name ***";
    private const string keyName = "*** Object Key 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()
    {
        client = new AmazonS3Client(bucketRegion);
        CreateAndDeleteObjectVersionAsync().Wait();
    }

    private static async Task CreateAndDeleteObjectVersionAsync()
    {
        try
        {
            // Add a sample object.
            string versionID = await PutAnObject(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 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);
        }
    }

    static async Task<string> PutAnObject(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;
    }
}
```

## 使用适用于 PHP 的 AWS 开发工具包删除对象

本主题演示如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类删除不受版本控制的存储桶中的对象。有关从受版本控制的存储桶中删除对象的信息，请参阅[使用 REST API 删除对象 \(p. 209\)](#)。

此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并正确安装了适用于 PHP 的 AWS 开发工具包。

下面的 PHP 示例将删除存储桶中的对象。由于此示例演示如何删除不受版本控制的存储桶中的对象，因此它在删除请求中仅提供存储桶名称和对象键（而不是版本 ID）。有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// Delete an object from the bucket.
$s3->deleteObject([
    'Bucket' => $bucket,
    'Key'     => $keyname
]);
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 [Aws\S3\S3Client](#) 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用 REST API 删除对象

您可以使用 AWS 开发工具包删除对象。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [DELETE Object](#)。

## 每个请求删除多个对象

### 主题

- [使用 AWS SDK for Java 删除多个对象 \(p. 209\)](#)
- [使用适用于 .NET 的 AWS 开发工具包删除多个对象 \(p. 213\)](#)
- [使用适用于 PHP 的 AWS 开发工具包删除多个对象 \(p. 218\)](#)
- [使用 REST API 删除多个对象 \(p. 220\)](#)

Amazon S3 提供多对象删除 API（请参阅[删除 – 多对象删除](#)），使您能够在单个请求中删除多个对象。API 支持两种模式的响应：详细模式和安静模式。默认情况下，操作将使用详细模式，在该模式下响应将包含请求中处理的每个键删除的结果。在安静模式下，响应仅包含删除操作出错时的键。

如果使用静默模式时，所有的键都已成功删除，Amazon S3 将返回空的响应。

要了解对象删除的更多信息，请参阅[删除对象 \(p. 203\)](#)。

您可以直接使用 REST API 或使用 AWS 开发工具包。

## 使用AWS SDK for Java删除多个对象

AWS SDK for Java 提供了用于删除多个对象的 `AmazonS3Client.deleteObjects()` 方法。对于要删除的每个对象，请指定键名称。如果存储桶受版本控制，您可以选择以下选项：

- 仅指定对象的键名称。Amazon S3 将向对象添加一个删除标记。

- 指定要删除的对象的键名称和版本 ID。Amazon S3 将删除指定版本的对象。

### Example

下面的示例使用多对象删除 API 从不受版本控制的存储桶中删除对象。该示例将示例对象上传到存储桶，然后使用 `AmazonS3Client.deleteObjects()` 方法删除单个请求中的对象。在 `DeleteObjectsRequest` 中，该示例仅指定对象键名称，因为对象没有版本 ID。

有关删除对象的更多信息，请参阅[删除对象 \(p. 203\)](#)。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;
import java.util.ArrayList;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;
import com.amazonaws.services.s3.model.DeleteObjectsResult;

public class DeleteMultipleObjectsNonVersionedBucket {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Upload three sample objects.
            ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
            for (int i = 0; i < 3; i++) {
                String keyName = "delete object example " + i;
                s3Client.putObject(bucketName, keyName, "Object number " + i + " to be
deleted.");
                keys.add(new KeyVersion(keyName));
            }
            System.out.println(keys.size() + " objects successfully created.");

            // Delete the sample objects.
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest(bucketName)
                .withKeys(keys)
                .withQuiet(false);

            // Verify that the objects were deleted successfully.
            DeleteObjectsResult delObjRes =
s3Client.deleteObjects(multiObjectDeleteRequest);
            int successfulDeletes = delObjRes.getDeletedObjects().size();
            System.out.println(successfulDeletes + " objects successfully deleted.");
        }
        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();  
}  
}  
}
```

### Example

下面的示例使用多对象删除 API 从受版本控制的存储桶中删除对象。它将执行以下操作：

1. 创建示例对象然后删除它们，并为要删除的每个对象指定键名称和版本 ID。本操作仅删除指定对象版本。
2. 通过仅指定键名称来创建示例对象然后删除它们。由于该示例不指定版本 ID，本操作将向每个对象添加一个删除标记而不删除任何特定对象版本。在添加删除标记后，这些对象不会显示在 AWS 管理控制台中。
3. 通过指定删除标记的对象键和版本 ID 来移除删除标记。本操作将移除删除标记，从而使对象再次显示在 AWS 管理控制台中。

```
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
  
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.SdkClientException;  
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.services.s3.AmazonS3;  
import com.amazonaws.services.s3.AmazonS3ClientBuilder;  
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;  
import com.amazonaws.services.s3.model.DeleteObjectsRequest;  
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;  
import com.amazonaws.services.s3.model.DeleteObjectsResult;  
import com.amazonaws.services.s3.model.DeleteObjectsResult.DeletedObject;  
import com.amazonaws.services.s3.model.PutObjectResult;  
  
public class DeleteMultipleObjectsVersionEnabledBucket {  
    private static AmazonS3 S3_CLIENT;  
    private static String VERSIONED_BUCKET_NAME;  
  
    public static void main(String[] args) throws IOException {  
        String clientRegion = "**** Client region ****";  
        VERSIONED_BUCKET_NAME = "**** Bucket name ****";  
  
        try {  
            S3_CLIENT = AmazonS3ClientBuilder.standard()  
                .withCredentials(new ProfileCredentialsProvider())  
                .withRegion(clientRegion)  
                .build();  
  
            // Check to make sure that the bucket is versioning-enabled.  
            String bucketVersionStatus =  
                S3_CLIENT.getBucketVersioningConfiguration(VERSIONED_BUCKET_NAME).getStatus();  
            if (!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED)) {  
                System.out.printf("Bucket %s is not versioning-enabled.",  
                    VERSIONED_BUCKET_NAME);  
            }  
            else {  
                // Upload and delete sample objects, using specific object versions.  
                uploadAndDeleteObjectsWithVersions();  
  
                // Upload and delete sample objects without specifying version IDs.  
                // Amazon S3 creates a delete marker for each object rather than deleting  
                // specific versions.  
                DeleteObjectsResult unversionedDeleteResult =  
                    uploadAndDeleteObjectsWithoutVersions();  
            }  
        }  
    }  
}
```

```
// Remove the delete markers placed on objects in the non-versioned create/
delete method.
        multiObjectVersionedDeleteRemoveDeleteMarkers(unversionedDeleteResult);
    }
}
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();
}
}

private static void uploadAndDeleteObjectsWithVersions() {
    System.out.println("Uploading and deleting objects with versions specified.");

    // Upload three sample objects.
    ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
    for (int i = 0; i < 3; i++) {
        String keyName = "delete object without version ID example " + i;
        PutObjectResult putResult = S3_CLIENT.putObject(VERSIONED_BUCKET_NAME, keyName,
            "Object number " + i + " to be deleted.");
        // Gather the new object keys with version IDs.
        keys.add(new KeyVersion(keyName, putResult.getVersionId()));
    }

    // Delete the specified versions of the sample objects.
    DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest(VERSIONED_BUCKET_NAME)
        .withKeys(keys)
        .withQuiet(false);

    // Verify that the object versions were successfully deleted.
    DeleteObjectsResult delObjRes = S3_CLIENT.deleteObjects(multiObjectDeleteRequest);
    int successfulDeletes = delObjRes.getDeletedObjects().size();
    System.out.println(successfulDeletes + " objects successfully deleted");
}

private static DeleteObjectsResult uploadAndDeleteObjectsWithoutVersions() {
    System.out.println("Uploading and deleting objects with no versions specified.");

    // Upload three sample objects.
    ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
    for (int i = 0; i < 3; i++) {
        String keyName = "delete object with version ID example " + i;
        S3_CLIENT.putObject(VERSIONED_BUCKET_NAME, keyName, "Object number " + i + " to
be deleted.");
        // Gather the new object keys without version IDs.
        keys.add(new KeyVersion(keyName));
    }

    // Delete the sample objects without specifying versions.
    DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest(VERSIONED_BUCKET_NAME).withKeys(keys)
        .withQuiet(false);

    // Verify that delete markers were successfully added to the objects.
    DeleteObjectsResult delObjRes = S3_CLIENT.deleteObjects(multiObjectDeleteRequest);
    int successfulDeletes = delObjRes.getDeletedObjects().size();
    System.out.println(successfulDeletes + " objects successfully marked for deletion
without versions.");
}
```

```

        return delObjRes;
    }

    private static void multiObjectVersionedDeleteRemoveDeleteMarkers(DeleteObjectsResult
response) {
    List<KeyVersion> keyList = new ArrayList<KeyVersion>();
    for (DeletedObject deletedObject : response.getDeletedObjects()) {
        // Note that the specified version ID is the version ID for the delete marker.
        keyList.add(new KeyVersion(deletedObject.getKey(),
deletedObject.getDeleteMarkerVersionId()));
    }
    // Create a request to delete the delete markers.
    DeleteObjectsRequest deleteRequest = new
DeleteObjectsRequest(VERSIONED_BUCKET_NAME).withKeys(keyList);

    // Delete the delete markers, leaving the objects intact in the bucket.
    DeleteObjectsResult delObjRes = S3_CLIENT.deleteObjects(deleteRequest);
    int successfulDeletes = delObjRes.getDeletedObjects().size();
    System.out.println(successfulDeletes + " delete markers successfully deleted");
}
}

```

## 使用适用于 .NET 的 AWS 开发工具包删除多个对象

适用于 .NET 的 AWS 开发工具包提供了用于删除多个对象的便利方法：DeleteObjects。对于要删除的每个对象，您可指定对象的键名称和版本。如果存储桶不受版本控制，您可为版本 ID 指定 null。如果出现异常，请查看 DeleteObjectsException 响应以确定哪些对象未删除成功以及为何如此。

### Example 从不受版本控制的存储桶中删除多个对象

下面的 C# 示例使用多对象删除 API 从不受版本控制的存储桶中删除对象。该示例将示例对象上传到存储桶，然后使用 DeleteObjects 方法删除单个请求中的对象。在 DeleteObjectsRequest 中，该示例将仅指定对象键名称，因为版本 ID 为空。

有关创建和测试有效示例的信息，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```

// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteMultipleObjectsNonVersionedBucketTest
    {
        private const string bucketName = "*** versioning-enabled 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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            MultiObjectDeleteAsync().Wait();
        }

        static async Task MultiObjectDeleteAsync()
        {

```

```

// Create sample objects (for subsequent deletion).
var keysAndVersions = await PutObjectsAsync(3);

// a. multi-object delete by specifying the key names and version IDs.
DeleteObjectsRequest multiObjectDeleteRequest = new DeleteObjectsRequest
{
    BucketName = bucketName,
    Objects = keysAndVersions // This includes the object keys and null version
IDs.
};
// You can add specific object key to the delete request using the .AddKey.
// multiObjectDeleteRequest.AddKey("TickerReference.csv", null);
try
{
    DeleteObjectsResponse response = await
s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
    Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
}
catch (DeleteObjectsException e)
{
    PrintDeletionErrorStatus(e);
}
}

private static void PrintDeletionErrorStatus(DeleteObjectsException e)
{
    // var errorResponse = e.ErrorResponse;
    DeleteObjectsResponse errorResponse = e.Response;
    Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

    Console.WriteLine("No. of objects successfully deleted = {0}",
errorResponse.DeletedObjects.Count);
    Console.WriteLine("No. of objects failed to delete = {0}",
errorResponse.DeleteErrors.Count);

    Console.WriteLine("Printing error data...");
    foreach (DeleteError deleteError in errorResponse.DeleteErrors)
    {
        Console.WriteLine("Object Key: {0}\t{1}\t{2}", deleteError.Key,
deleteError.Code, deleteError.Message);
    }
}

static async Task<List<KeyVersion>> PutObjectsAsync(int number)
{
    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 s3Client.PutObjectAsync(request);
        KeyVersion keyVersion = new KeyVersion
        {
            Key = key,
            // For non-versioned bucket operations, we only need object key.
            // VersionId = response.VersionId
        };
        keys.Add(keyVersion);
    }
}

```

```
        return keys;
    }
}
```

### Example 受版本控制的存储桶的多对象删除

下面的 C# 示例使用多对象删除 API 从受版本控制的存储桶中删除对象。该示例将执行以下操作：

1. 创建示例对象然后删除它们，方法是为每个对象指定键名称和版本 ID。本操作将删除特定版本的对象。
2. 通过仅指定键名称来创建示例对象然后删除它们。由于该示例不指定版本 ID，本操作仅添加删除标记。它不会删除特定版本的对象。在删除后，这些对象不会显示在 Amazon S3 控制台中。
3. 通过指定删除标记的对象键和版本 ID 来移除删除标记。当本操作移除删除标记后，这些对象将再次显示在控制台中。

有关创建和测试有效示例的信息，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteMultipleObjVersionedBucketTest
    {
        private const string bucketName = "*** versioning-enabled 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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            DeleteMultipleObjectsFromVersionedBucketAsync().Wait();
        }

        private static async Task DeleteMultipleObjectsFromVersionedBucketAsync()
        {

            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync();

            // Delete objects (without specifying object version in the request).
            var deletedObjects = await DeleteObjectsAsync();

            // Additional exercise - remove the delete markers S3 returned in 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(deletedObjects);
        }

        private static async Task<List<DeletedObject>> DeleteObjectsAsync()
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(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(keysAndVersions2);
    return deletedObjects;
}

private static async Task DeleteObjectVersionsAsync()
{
    // Upload the sample objects.
    var keysAndVersions1 = await PutObjectsAsync(3);

    // Delete the specific object versions.
    await VersionedDeleteAsync(keysAndVersions1);
}

private static void PrintDeletionReport(DeleteObjectsException e)
{
    var errorResponse = e.Response;
    Console.WriteLine("No. of objects successfully deleted = {0}",
errorResponse.DeletedObjects.Count);
    Console.WriteLine("No. of objects failed to delete = {0}",
errorResponse.DeleteErrors.Count);
    Console.WriteLine("Printing error data...");
    foreach (var deleteError in errorResponse.DeleteErrors)
    {
        Console.WriteLine("Object Key: {0}\t{1}\t{2}", deleteError.Key,
deleteError.Code, deleteError.Message);
    }
}

static async Task VersionedDeleteAsync(List<KeyVersion> keys)
{
    // a. Perform a multi-object delete by specifying the key names and version
IDs.
    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
s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
    }
    catch (DeleteObjectsException e)
    {
        PrintDeletionReport(e);
    }
}

static async Task<List<DeletedObject>> NonVersionedDeleteAsync(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 DeleteObjects - Amazon S3 add delete marker for each object
// deletion. The objects disappear from your bucket.
// You can verify that using the Amazon S3 console.
DeleteObjectsResponse response;
try
{
    Console.WriteLine("Executing NonVersionedDelete...");
    response = await s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
    Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
}
catch (DeleteObjectsException e)
{
    PrintDeletionReport(e);
    throw; // Some deletes failed. Investigate before continuing.
}
// This response contains the DeletedObjects list which we use to delete the
delete markers.
return response.DeletedObjects;
}

private static async Task RemoveDeleteMarkersAsync(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
s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} delete markers",
                deleteObjectResponse.DeletedObjects.Count);
    }
    catch (DeleteObjectsException e)
    {
        PrintDeletionReport(e);
    }
}

static async Task<List<KeyVersion>> PutObjectsAsync(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 s3Client.PutObjectAsync(request);
    KeyVersion keyVersion = new KeyVersion
    {
        Key = key,
        VersionId = response.VersionId
    };

    keys.Add(keyVersion);
}
return keys;
}
}
```

## 使用适用于 PHP 的 AWS 开发工具包删除多个对象

本主题将演示如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类从受版本控制和不受版本控制的 Amazon S3 存储桶中删除多个对象。有关版本控制的更多信息，请参阅[使用版本控制 \(p. 380\)](#)。

此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并正确安装了适用于 PHP 的 AWS 开发工具包。

### Example 从不受版本控制的存储桶中删除多个对象

下面的 PHP 示例使用 `deleteObjects()` 方法从不受版本控制的存储桶中删除多个对象。

有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

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

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

// 1. Create a few objects.
for ($i = 1; $i <= 3; $i++) {
    $s3->putObject([
        'Bucket' => $bucket,
        'Key'     => "key{$i}",
        'Body'    => "content {$i}",
    ]);
}

// 2. List the objects and get the keys.
$keys = $s3->listObjects([
    'Bucket' => $bucket
]) ->getPath('Contents/*/Key');

// 3. Delete the objects.
$s3->deleteObjects([
    'Bucket'  => $bucket,
```

```
'Delete' => [
    'Objects' => array_map(function ($key) {
        return ['Key' => $key];
    }, $keys)
],
]);
```

### Example 从受版本控制的存储桶中删除多个对象

下面的 PHP 示例使用 `deleteObjects()` 方法从受版本控制的存储桶中删除多个对象。

有关运行本指南中的 PHP 示例的信息，请参阅[运行 PHP 示例 \(p. 524\)](#)。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// 1. Enable object versioning for the bucket.
$s3->putBucketVersioning([
    'Bucket' => $bucket,
    'Status' => 'Enabled',
]);

// 2. Create a few versions of an object.
for ($i = 1; $i <= 3; $i++) {
    $s3->putObject([
        'Bucket' => $bucket,
        'Key'     => $keyname,
        'Body'   => "content {$i}",
    ]);
}

// 3. List the objects versions and get the keys and version IDs.
$versions = $s3->listObjectVersions(['Bucket' => $bucket])
    ->getPath('Versions');

// 4. Delete the object versions.
$s3->deleteObjects([
    'Bucket'  => $bucket,
    'Delete'  => [
        'Objects' => array_map(function ($version) {
            return [
                'Key'      => $version['Key'],
                'VersionId' => $version['VersionId']
            ], $version),
    ],
]);
echo "The following objects were deleted successfully:". PHP_EOL;
foreach ($result['Deleted'] as $object) {
    echo "Key: {$object['Key']}, VersionId: {$object['VersionId']}" . PHP_EOL;
}

echo PHP_EOL . "The following objects could not be deleted:" . PHP_EOL;
```

```
foreach ($result['Errors'] as $object) {  
    echo "Key: {$object['Key']}, VersionId: {$object['VersionId']}\" . PHP_EOL;  
}  
  
// 5. Suspend object versioning for the bucket.  
$s3->putBucketVersioning([  
    'Bucket' => $bucket,  
    'Status' => 'Suspended',  
]);
```

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 Aws\S3\S3Client 类
- 适用于 PHP 的 AWS 开发工具包文档

## 使用 REST API 删除多个对象

您可以使用 AWS 开发工具包，通过多个对象删除 API 删除多个对象。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [删除多个对象](#)。

## 从对象中选择内容

利用 Amazon S3 Select，您可以使用简单的结构化查询语言 (SQL) 语句筛选 Amazon S3 对象的内容，以便仅检索所需的部分数据。通过使用 Amazon S3 Select 筛选此数据，您可以减少 Amazon S3 传输的数据量，这将减少检索此数据所需的成本和延迟。

Amazon S3 Select 适用于以 CSV 或 JSON 格式存储的对象。它还使用通过 GZIP 或 BZIP2 压缩的对象和服务器端加密的对象。可以将结果的格式指定为 CSV 或 JSON，并且可以确定结果中记录的分隔方式。

可以在请求中将 SQL 表达式传递给 Amazon S3。Amazon S3 Select 支持一部分 SQL。有关 Amazon S3 Select 支持的 SQL 元素的更多信息，请参阅[适用于 Amazon S3 Select 和 Amazon Glacier Select 的 SQL 参考 \(p. 557\)](#)。

您可以使用 AWS 开发工具包、SELECT Object 内容 REST API、AWS Command Line Interface (AWS CLI) 或 Amazon S3 控制台执行 SQL 查询。Amazon S3 控制台将返回的数据量限定为 40 MB。要检索更多数据，请使用 AWS CLI 或 API。

## 要求和限制

以下是使用 Amazon S3 Select 的要求：

- 您必须拥有所查询的对象的 s3:GetObject 权限。
- 如果您查询的对象已使用客户提供的加密密钥 (SSE-C) 进行加密，则必须使用 https，并且您必须在请求中提供加密密钥。

使用 Amazon S3 Select 时存在以下限制：

- SQL 表达式的最大长度为 256 KB。
- 结果中记录的最大长度为 1 MB。

## 构建请求

在构建请求时，您提供通过使用 InputSerialization 对象查询的对象的详细信息。您提供要使用 OutputSerialization 对象返回结果的方式的详细信息。您还包含 Amazon S3 将用于筛选请求的 SQL 表达式。

有关构建 Amazon S3 Select 请求的更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [SELECT Object 内容](#)。您也可以参阅以下部分中的某个开发工具包代码示例。

## 错误

如果在尝试执行查询时遇到问题，Amazon S3 Select 将返回错误代码和关联的错误消息。有关错误代码和描述的列表，请参阅 Amazon Simple Storage Service API Reference 的 SELECT 对象内容页面的[特殊错误部分](#)。

### 主题

- [相关资源 \(p. 221\)](#)
- [使用适用于 Java 的开发工具包 从对象中选择内容 \(p. 221\)](#)
- [使用 REST API 从对象中选择内容 \(p. 223\)](#)
- [使用其他开发工具包从对象中选择内容 \(p. 223\)](#)

## 相关资源

- [使用 AWS 开发工具包、CLI 和 Explorer \(p. 518\)](#)

## 使用适用于 Java 的开发工具包 从对象中选择内容

使用 Amazon S3 Select 通过 `selectObjectContent` 方法来选择对象的内容，成功时将返回 SQL 表达式的结果。指定的存储桶和对象键必须存在，否则将出现错误。

### Example 示例

以下 Java 代码返回对象（包含以 CSV 格式存储的数据）中存储的每条记录的第一列的值。它还请求返回 `Progress` 和 `Stats` 消息。必须提供有效的存储桶名称和包含 CSV 格式的数据的对象。

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
package com.amazonaws;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CSVInput;
import com.amazonaws.services.s3.model.CSVOutput;
import com.amazonaws.services.s3.model.CompressionType;
import com.amazonaws.services.s3.model.ExpressionType;
import com.amazonaws.services.s3.model.InputSerialization;
import com.amazonaws.services.s3.model.OutputSerialization;
import com.amazonaws.services.s3.model.SelectObjectContentEvent;
import com.amazonaws.services.s3.model.SelectObjectContentEventVisitor;
import com.amazonaws.services.s3.model.SelectObjectContentRequest;
import com.amazonaws.services.s3.model.SelectObjectContentResult;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.concurrent.atomic.AtomicBoolean;

import static com.amazonaws.util.IOUtils.copy;

/**
 * This example shows how to query data from S3Select and consume the response in the form
 * of an
```

```
* InputStream of records and write it to a file.  
*/  
  
public class RecordInputStreamExample {  
  
    private static final String BUCKET_NAME = "${my-s3-bucket}";  
    private static final String CSV_OBJECT_KEY = "${my-csv-object-key}";  
    private static final String S3_SELECT_RESULTS_PATH = "${my-s3-select-results-path}";  
    private static final String QUERY = "select s._1 from S3Object s";  
  
    public static void main(String[] args) throws Exception {  
        final AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();  
  
        SelectObjectContentRequest request = generateBaseCSVRequest(BUCKET_NAME,  
CSV_OBJECT_KEY, QUERY);  
        final AtomicBoolean isResultComplete = new AtomicBoolean(false);  
  
        try (OutputStream fileOutputStream = new FileOutputStream(new File  
(S3_SELECT_RESULTS_PATH));  
             SelectObjectContentResult result = s3Client.selectObjectContent(request)) {  
            InputStream resultInputStream = result.getPayload().getRecordsInputStream(  
                new SelectObjectContentEventVisitor() {  
                    @Override  
                    public void visit(SelectObjectContentEvent.StatsEvent event)  
                    {  
                        System.out.println(  
                            "Received Stats, Bytes Scanned: " +  
event.getDetails().getBytesScanned()  
                            + " Bytes Processed: " +  
event.getDetails().getBytesProcessed());  
                    }  
  
                    /*  
                     * An End Event informs that the request has finished successfully.  
                     */  
                    @Override  
                    public void visit(SelectObjectContentEvent.EndEvent event)  
                    {  
                        isResultComplete.set(true);  
                        System.out.println("Received End Event. Result is complete.");  
                    }  
                }  
            );  
  
            copy(resultInputStream, fileOutputStream);  
        }  
  
        /*  
         * The End Event indicates all matching records have been transmitted.  
         * If the End Event is not received, the results may be incomplete.  
         */  
        if (!isResultComplete.get()) {  
            throw new Exception("S3 Select request was incomplete as End Event was not  
received.");  
        }  
    }  
  
    private static SelectObjectContentRequest generateBaseCSVRequest(String bucket, String  
key, String query) {  
        SelectObjectContentRequest request = new SelectObjectContentRequest();  
        request.setBucketName(bucket);  
        request.setKey(key);  
        request.setExpression(query);  
        request.setExpressionType(ExpressionType.SQL);  
  
        InputSerialization inputSerialization = new InputSerialization();
```

```
    inputSerialization.setCsv(new CSVInput());
    inputSerialization.setCompressionType(CompressionType.NONE);
    request.setInputSerialization(inputSerialization);

    OutputSerialization outputSerialization = new OutputSerialization();
    outputSerialization.setCsv(new CSVOutput());
    request.setOutputSerialization(outputSerialization);

    return request;
}
}
```

## 使用 REST API 从对象中选择内容

您可以使用 AWS 开发工具包从对象中选择内容。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。有关请求和响应格式的更多信息，请参阅 [SELECT 对象内容](#)。

## 使用其他开发工具包从对象中选择内容

可使用 Amazon S3 Select 通过其他开发工具包选择对象的内容。有关更多信息，请参阅下列内容：

- Python: [使用AWS SDK for Python \(Boto\) \(p. 525\)](#).

## 恢复存档对象

不能实时访问存档在 Amazon Glacier 中的对象。您必须首先启动还原请求，然后耐心等待，直到对象的临时副本根据您在请求中指定的持续时间 (天数) 变为可用。完成还原作业所需的时间取决于您指定的检索选项：Standard、Expedited 或 Bulk。有关将对象存档到 Amazon Glacier 的更多信息，请参阅 [转换为 GLACIER 存储类 \(对象存档\) \(p. 107\)](#)。

在收到已还原对象的临时副本后，此对象的存储类仍保持为 GLACIER (GET 或 HEAD 请求将返回 GLACIER 存储类)。请注意，还原某个存档时，您需要同时为存档 (GLACIER 费率) 和临时还原的副本 (RRS 费率) 付费。有关定价的信息，请参阅 [Amazon S3 定价](#)。

以下主题可提供更多信息。

### 主题

- [档案检索选项 \(p. 223\)](#)
- [使用 Amazon S3 控制台还原存档对象 \(p. 224\)](#)
- [使用 AWS SDK for Java 还原存档对象 \(p. 224\)](#)
- [使用适用于 .NET 的 AWS 开发工具包还原存档对象 \(p. 225\)](#)
- [使用 REST API 还原存档对象 \(p. 226\)](#)

## 档案检索选项

在还原存档对象时，您可以指定以下项之一：

- **Expedited** - 加速检索允许您在偶尔需要紧急请求档案子集时快速访问数据。对于除最大存档对象 (250 MB+) 之外的所有其他存档对象，使用加速检索访问的数据通常在 1 到 5 分钟内可用。加速检索有两种类型：按需和预配置。按需请求类似于 EC2 按需实例，大部分时间可用。预配置请求可确保在您需要时可用。有关更多信息，请参阅 [预配置容量 \(p. 224\)](#)。
- **Standard** - 标准检索允许您在数小时内访问您的任何存档对象。标准检索通常在 3 到 5 小时内完成。这是未指定检索选项的检索请求的默认选项。

- **Bulk** - 批量检索是 Amazon Glacier 最低成本的检索选项，使您可以在一天内以较低的成本检索大量（甚至是 PB 级）的数据。批量检索通常在 5 到 12 小时内完成。

要进行 Expedited、Standard 或 Bulk 检索，请将 [POST Object restore](#) REST API 请求中的 Tier 请求元素设置为您需要的选项，或者在 AWS CLI 或 AWS 软件开发工具包中设置等同参数。对于加速检索，无需指定加速检索是按需还是预配置的。如果您购买了预配置容量，则所有加速检索都会通过您的预配置容量自动获得处理。

您可以采用编程方式或使用 Amazon S3 控制台还原存档对象。Amazon S3 针对每个对象每次仅处理一个还原请求。使用控制台和 Amazon S3 API 都可以检查还原状态并找出 Amazon S3 将删除还原的副本的时间。

## 预配置容量

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。每个容量单位确保每五分钟至少可以执行三个加速检索，并提供高达 150 MB/秒的检索吞吐量。

如果您的工作负载需要极高的稳定性和对数据子集可预测的访问性能（以分钟为单位），您应该购买预配置检索容量。没有预配置容量的加速检索也可以接受，但是在极少情况下会出现不寻常的高需求。不过，如果您需要随时可以访问加速检索，您必须购买预配置检索容量。您可以使用 Amazon S3 控制台、Amazon Glacier 控制台、[购买预配置容量](#) REST API、AWS 软件开发工具包或 AWS CLI 购买预配置容量。有关预配置容量的定价信息，请参阅 [Amazon Glacier 定价](#)。

## 使用 Amazon S3 控制台还原存档对象

您可以使用 Amazon S3 控制台还原已存档到 Amazon Glacier 中的对象副本。有关如何使用 AWS 管理控制台还原存档的说明，请参阅 [如何还原已存档到 Amazon Glacier 中的 S3 对象？](#)（在 Amazon Simple Storage Service 控制台用户指南中）。

请注意，恢复某个存档时，您需要同时为存档和临时恢复的副本付费。有关定价的信息，请参阅 [Amazon S3 定价](#)。

Amazon S3 只会在指定的持续时间内还原对象的临时副本。在此之后，Amazon S3 会删除还原的对象副本。您可以通过重新发布还原修改已还原的副本的过期时段（在这种情况下，Amazon S3 将根据当前时间更新过期时段）。

Amazon S3 通过将还原请求中指定的天数添加到当前时间，然后将得出的时间舍入至第二天的午夜 UTC 来计算还原对象副本的过期时间。例如，如果对象的创建日期为 2012 年 10 月 15 日上午 10:30 UTC，且还原时段指定为 3 天，则还原的副本将在 2012 年 10 月 19 日 00:00 UTC 过期，Amazon S3 将在该时间删除对象副本。

您可以选择任意的天数来恢复对象副本。然而，鉴于与对象副本相关的存储成本，您应该仅在您所需的持续时间内恢复对象。有关定价信息，请参阅 [Amazon S3 定价](#)。

## 使用 AWS SDK for Java 还原存档对象

### Example

以下示例说明如何使用 AWS SDK for Java 还原已存档到 Amazon Glacier 的对象。该示例为指定的存档对象启动还原请求并检查其还原状态。有关还原已存档对象的更多信息，请参阅 [恢复存档对象 \(p. 223\)](#)。有关创建和测试有效示例的说明，请参阅 [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

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

```
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.RestoreObjectRequest;

public class RestoreArchivedObject {

    public static void main(String[] args) throws IOException {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Create and submit a request to restore an object from Glacier for two days.
            RestoreObjectRequest requestRestore = new RestoreObjectRequest(bucketName,
keyName, 2);
            s3Client.restoreObjectV2(requestRestore);

            // Check the restoration status of the object.
            ObjectMetadata response = s3Client.getObjectMetadata(bucketName, keyName);
            Boolean restoreFlag = response.getOngoingRestore();
            System.out.format("Restoration status: %s.\n",
                restoreFlag ? "in progress" : "not in progress (finished or failed)");
        }
        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();
        }
    }
}
```

## 使用适用于 .NET 的 AWS 开发工具包 还原存档对象

### Example

以下 C# 示例启动将存档对象还原 2 天的请求。Amazon S3 将在对象元数据中保存还原状态。在启动此请求后，示例将检索对象元数据并检查 `RestoreInProgress` 属性的值。有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class RestoreArchivedObjectTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string objectKey = "*** archived object key 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()
{
    client = new AmazonS3Client(bucketRegion);
    RestoreObjectAsync(client, bucketName, objectKey).Wait();
}

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("An AmazonS3Exception was thrown. Exception: " +
amazonS3Exception.ToString());
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.ToString());
    }
}

static async Task CheckRestorationStatusAsync(IAmazonS3 client, string bucketName,
string objectKey)
{
    GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest
    {
        BucketName = bucketName,
        Key = objectKey
    };
    GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
    Console.WriteLine("restoration status: {0}", response.RestoreInProgress ? "in-
progress" : "finished or failed");
}
```

## 使用 REST API 还原存档对象

Amazon S3 向您提供了用于启动存档还原的 API。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [POST Object restore](#)。

## 查询存档对象

利用 [POST Object restore](#) select 类型，针对由 Amazon S3 存档到 Amazon Glacier 的数据，您可以直接使用简单的结构化查询语言 (SQL) 语句执行筛选操作。当您对存档对象进行 SQL 查询时，select 会就地运行

该查询，并将输出结果写入一个 S3 存储桶中。您可以针对存储在 Amazon Glacier 中的数据运行查询和自定义分析，而不必将整个对象还原到 Amazon S3。

当您执行 select 查询时，Amazon Glacier 会提供三个数据访问层 — 加速、标准和批量。所有这些层提供不同的数据访问时间和成本，您可以根据您希望数据可用的速度选择其中任何一个层。有关更多信息，请参阅[数据访问套餐 \(p. 229\)](#)。

您可以将还原的 select 类型与 AWS 开发工具包、Amazon Glacier REST API 和 AWS Command Line Interface (AWS CLI) 结合使用。

#### 主题

- [Select 要求和限制 \(p. 227\)](#)
- [如何使用 Select 查询数据？\(p. 227\)](#)
- [错误处理 \(p. 228\)](#)
- [数据访问套餐 \(p. 229\)](#)
- [更多信息 \(p. 229\)](#)

## Select 要求和限制

以下是使用 select 的要求：

- 由 select 查询的存档对象必须格式化为未压缩的逗号分隔值 (CSV)。
- 作为输出的 S3 存储桶。用于启动 Amazon Glacier select 任务的 AWS 账户必须具有 S3 存储桶的写入权限。该 Amazon S3 存储桶必须位于与包含要查询的存档对象的存储桶相同的 AWS 区域中。
- 提出请求的 AWS 账户必须拥有执行 s3:RestoreObject 和 s3:GetObject 操作的权限。有关这些权限的更多信息，请参阅[与存储桶子资源操作相关的权限 \(p. 284\)](#)。
- 存档一定不能使用 SSE-C 或客户端加密进行加密。

使用 select 时存在以下限制：

- select 可以处理的记录数量没有限制。输入或输出记录不得超过 1 MB，否则查询会失败。每个记录有 1,048,576 列的限制。
- 对最终结果的大小没有限制。但结果分成多个部分。
- SQL 表达式限制为 128 KB。

## 如何使用 Select 查询数据？

通过 select，您可以使用 SQL 命令查询采用加密的未压缩 CSV 格式的 Amazon Glacier 存档对象。通过此限制，您可以对 Amazon Glacier 中基于文本的数据执行简单的查询操作。例如，可以在一系列存档文本文件中查找特定名称或 ID。

要查询您的 Amazon Glacier 数据，请使用 [POST Object restore](#) 操作创建 select 请求。在执行 Select 请求时，您需要提供 SQL 表达式、要查询的档案以及存储结果的位置。

下面的示例表达式从 [POST Object restore](#) 中指定的存档对象返回所有记录。

```
SELECT * FROM object
```

Amazon Glacier Select 支持 ANSI SQL 语言的一部分。它支持通用筛选 SQL 子句，如 SELECT、FROM 和 WHERE。它不支持 SUM、COUNT、GROUP BY、JOINS、DISTINCT、UNION、ORDER BY 和 LIMIT。有关 SQL 支持的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的 [Amazon S3 Select](#) 和 [Amazon Glacier Select 的 SQL 参考](#)。

## Select 输出

在您启动 Select 请求时，需要为 select 查询的结果定义输出位置。该位置必须是一个 Amazon S3 存储桶，与包含要查询的存档对象的存储桶位于相同的 AWS 区域中。启动任务的 AWS 账户必须具有写入 S3 存储桶的权限。

您可以为存储在 Amazon S3 中的输出对象指定 Amazon S3 存储类和加密。Select 支持 SSE-KMS 和 SSE-S3 加密。Select 不支持 SSE-C 和客户端加密。有关 Amazon S3 存储类和加密的更多信息，请参阅[存储类别 \(p. 90\)](#)和[使用服务器端加密保护数据 \(p. 345\)](#)。

Amazon Glacier Select 结果使用在 [POST Object restore](#) 中指定的输出位置中提供的前缀存储在 S3 存储桶中。通过此信息，select 会创建一个引用任务 ID 的唯一前缀。(前缀用于通过以通用字符串开始对象名称将 Amazon S3 对象组合在一起。) 在这个唯一前缀之下，有两个新建的前缀，`results` 用于结果，`errors` 用于日志和错误。完成任务后，将会写入一个结果清单，其中包含所有结果的位置。

还有一个名为 `job.txt` 的占位符文件，它会写入到输出位置。它在写入后便从不更新。占位符文件用于以下事项：

- 同步验证写入权限和大多数 SQL 语法错误。
- 提供与您的 select 请求相关的静态输出，可供您随时方便地引用。

例如，假设您将 select 请求结果的输出位置指定为 `s3://example-bucket/my-prefix`，而且任务响应返回的任务 ID 为 `examplekne1209ualkdjh812elkassdu9012e`。在 select 任务完成后，您可以在存储桶中看到以下 Amazon S3 对象：

```
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/job.txt
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/results/abc
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/results/def
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/results/ghi
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/result_manifest.txt
```

Select 查询结果分成多个部分。在此示例中，select 使用您在设置输出位置时指定的前缀，并附加任务 ID 和 `results` 前缀。然后，它将结果写入三个部分中，其各自的对象名称以 `abc`、`def` 和 `ghi` 结尾。结果清单包含全部三个文件以允许编程检索。如果任务因任何错误而失败，则会在错误前缀下面显示一个文件，并且生成 `error_manifest.txt`。

具有 `result_manifest.txt` 文件但缺少 `error_manifest.txt` 可确保任务成功完成。对如何排序结果不提供保证。

### Note

Amazon S3 对象名称（也称为键）的长度不能超过 1024 字节。Amazon Glacier select 为前缀保留 128 字节。而且，Amazon S3 路径位置的长度不能超过 512 字节。长度大于 512 字节的请求会返回异常，并且该请求不被接受。

## 错误处理

Select 会通知您两种错误。第一组错误会于您在 [POST Object restore](#) 中提交查询时同步发送给您。这些错误将作为 HTTP 响应的一部分发送给您。在成功接受查询后，可能会发生另一组错误，但这些错误是在查询执行过程中发生的。在这种情况下，错误将会写入到在 `errors` 前缀下指定的输出位置中。

Select 在遇到错误后将会停止执行查询。要成功执行查询，您必须解析所有错误。您可以检查日志以确定哪些记录导致了故障。

由于查询在多个计算节点之间并行运行，因此您得到的错误不是按顺序排序的。例如，如果查询在第 6234 行中因出现错误而失败，并不意味着第 6234 行之前的所有行都已成功处理。下次运行查询时可能会在其他行中显示错误。

## 数据访问套餐

在查询存档对象时，您可以指定以下数据访问套餐之一：

- **Expedited** – 允许您在偶尔需要紧急请求档案子集时快速访问数据。对于除了最大型存档对象 (250 MB+) 之外的所有其他档案，使用 Expedited 检索访问的数据通常在 1 到 5 分钟内可用。Expedited 数据访问有两种类型：按需和预配置。按需请求类似于 EC2 按需实例，大部分时间可用。预配置请求可确保在您需要时可用。有关更多信息，请参阅 [预配置容量 \(p. 229\)](#)。
- **Standard** – 允许您在数小时内访问您的任意存档对象。Standard 检索通常可在 3 到 5 小时内完成。这是默认套餐。
- **Bulk** – Amazon Glacier 中成本最低的数据访问选项，使您可以在一天内以较低的成本检索大量（甚至是 PB 级）的数据。Bulk 访问通常可在 5 到 12 小时内完成。

要提出 Expedited、Standard 或 Bulk 请求，请将 [POST Object restore](#) REST API 请求中的 Tier 请求元素设置为您需要的选项，或者在 AWS CLI 或 AWS 开发工具包中设置等同参数。对于 Expedited 访问，无需指定加速检索是按需还是预配置。如果您购买了预配置容量，则所有 Expedited 检索都会通过您的预配置容量自动获得处理。有关层定价的信息，请参阅 [Amazon Glacier 定价](#)。

### 预配置容量

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。每个容量单位确保每五分钟至少可以执行三个加速检索，并提供高达 150 MB/秒的检索吞吐量。

如果您的工作负载需要极高的稳定性和对数据子集可预测的访问性能（以分钟为单位），您应该购买预配置检索容量。没有预配置容量的 Expedited 检索也可以接受，但在极少数出现不寻常的高需求的情况下例外。不过，如果需要随时可以访问 Expedited 检索，您必须购买预配置检索容量。您可以使用 Amazon S3 控制台、Amazon Glacier 控制台、[购买预配置容量](#) REST API、AWS 软件开发工具包或 AWS CLI 购买预配置容量。有关预配置容量的定价信息，请参阅 [Amazon Glacier 定价](#)。

### 更多信息

- Amazon Simple Storage Service API Reference 中的 [POST Object restore](#)
- Amazon Simple Storage Service 开发人员指南 中的 [Amazon S3 Select 和 Amazon Glacier Select 的 SQL 参考](#)

# Amazon S3 分析 - 存储类分析

通过使用 Amazon S3 分析存储类分析，您可以分析存储访问模式以帮助您决定何时将正确的数据转换为正确的存储类。此新的 Amazon S3 分析功能可观察数据访问模式以帮助您确定何时将不常访问的 STANDARD 存储转换为 STANDARD\_IA (IA，适用于不常访问) 存储类。有关存储类别的更多信息，请参阅 [存储类别 \(p. 90\)](#)。

在存储类分析观察一组筛选出的数据的不常访问模式一段时间后，您可以使用分析结果来帮助改进您的生命周期策略。您可以将存储类分析配置为分析存储桶中的所有对象。或者，也可以配置筛选条件以按通用前缀（即，名称以通用字符串开头的对象）、对象标签或前缀和标签的组合对对象进行分组以便进行分析。您很有可能会发现，按对象组进行筛选是从存储类分析获益的最佳方式。

## Important

存储类分析不提供转换到 ONEZONE\_IA 或 GLACIER 存储类的建议。

每个存储桶可具有多个存储类分析筛选条件（最多 1,000 个），并且将收到针对每个筛选条件的单独分析。利用多个筛选配置，您可以分析特定对象组以改进将对象转换为 STANDARD\_IA 的生命周期策略。

存储类分析显示 Amazon S3 控制台中的存储使用率可视化项，这些可视化项每天将进行更新。存储使用率数据也可以每天导出到 S3 存储桶中的文件。您可以在电子表格应用程序中打开导出的使用率报告文件，或者通过您选择的商业智能工具（如 Amazon QuickSight）使用该文件。

## 主题

- [如何设置存储类分析？ \(p. 230\)](#)
- [如何使用存储类分析？ \(p. 231\)](#)
- [如何导出存储类分析数据？ \(p. 233\)](#)
- [Amazon S3 分析 REST API \(p. 234\)](#)

## 如何设置存储类分析？

通过配置要分析的对象数据来设置存储类分析。您可将存储类分析配置为执行以下操作：

- 分析存储桶的全部内容。

您将收到针对存储桶中所有对象的分析。

- 分析按前缀和标签分组的对象。

您可以配置按前缀、对象标签或前缀和标签的组合对对象进行分组以进行分析的筛选条件。您将收到针对配置的每个筛选条件的单独分析。每个存储桶可具有多个筛选配置（最多 1,000 个）。

- 导出分析数据。

当您为存储桶或筛选条件配置存储类分析时，可以选择每天将分析数据导出到一个文件。当天的分析将添加到该文件以形成针对配置的筛选条件的历史分析日志。该文件会在所选目标上每天进行更新。选择要导出的数据时，指定写入文件的目标存储桶和可选目标前缀。

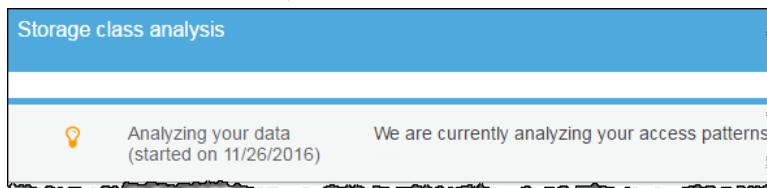
您可以使用 Amazon S3 控制台、REST API、AWS CLI 或 AWS 开发工具包配置存储类分析。

- 有关如何在 Amazon S3 控制台中配置存储类分析的信息，请参阅 [如何配置存储类分析？](#)。
- 要使用 Amazon S3 API，可使用 [PutBucketAnalyticsConfiguration](#) REST API 或者 AWS CLI 或 AWS 开发工具包中的等效项。

## 如何使用存储类分析？

您使用存储类分析观察数据访问模式一段时间来收集信息，以帮助您改进 STANDARD\_IA 存储的生命周期管理。配置一个筛选条件后，您首先会在 Amazon S3 控制台中看到 24 到 48 小时内基于该筛选条件的数据分析。但是，存储类分析会观察筛选的数据集的访问模式 30 天或更长时间，以便在提供结果前收集用于分析的信息。分析会在初始结果后继续运行，并在访问模式发生更改时更新结果。

当您首次配置筛选条件时，Amazon S3 控制台会显示一条与以下内容类似的消息。



存储类分析观察筛选的对象数据集的访问模式 30 天或更长时间，以便收集足量信息来进行分析。在存储类分析收集足量信息后，您将在 Amazon S3 控制台中看到一条与以下内容类似的消息。

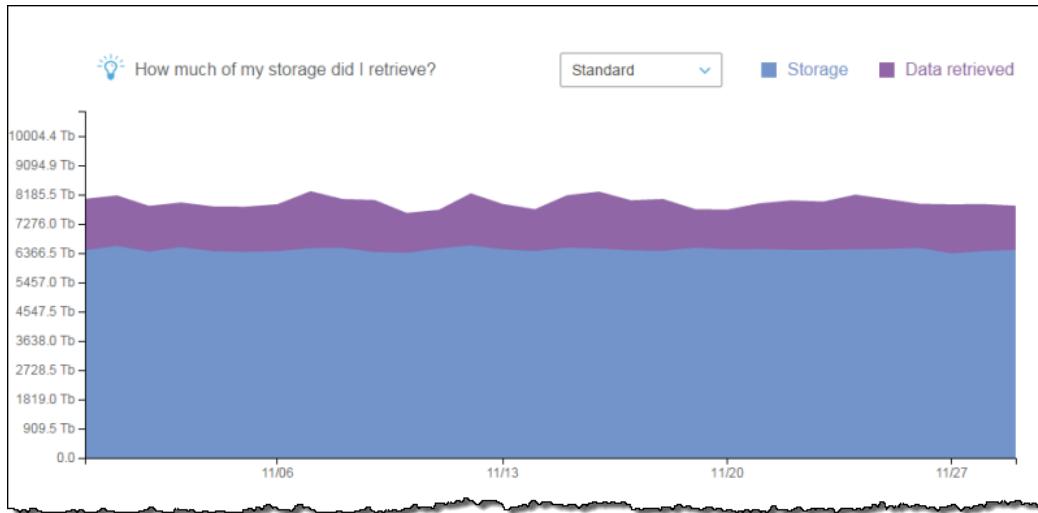


对不常访问的对象执行分析时，存储类分析基于期限查看已分组的筛选的对象集，因为这些对象已上传到 Amazon S3。存储类分析通过查看筛选的数据集的以下因素来确定是否不常访问期限组：

- STANDARD 存储类中的大于 128K 的对象。
- 每个期限组内的平均总存储量。
- 每个期限组传出的平均字节数 (非频率)。
- 分析导出数据仅包含与存储类分析有关的数据请求。这可能导致相对于存储指标所示或您自己内部系统的跟踪结果，请求数量以及上传和请求总字节数存在差异。
- 失败的 GET 和 PUT 请求不计入分析。但是，您会在存储指标中看到失败的请求。

我检索了多少我的存储空间？

Amazon S3 控制台用图表表示观察期间检索到的筛选的数据集中的存储空间量，如以下示例所示。



我检索了多少百分比的我的存储空间？

Amazon S3 控制台也用图表表示观察期间检索到的筛选的数据集中的存储空间百分比，如以下示例所示。



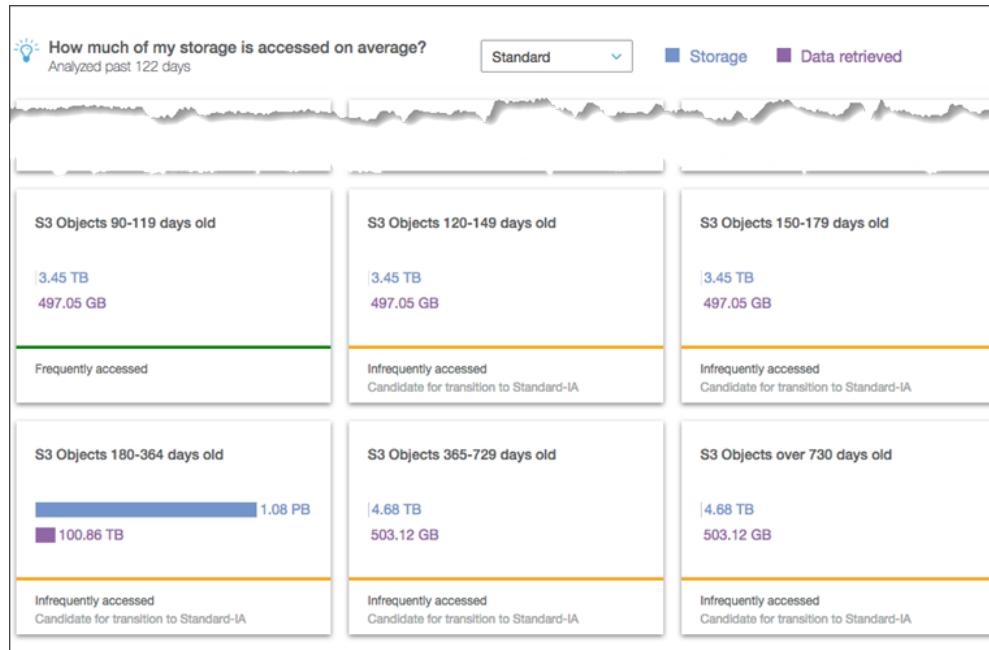
如本主题中先前所述，当您对不常访问的对象执行分析时，存储类分析会基于期限查看已分组的筛选的对象集，因为这些对象已上传到 Amazon S3。存储类分析使用以下预定义的对象期限组：

- &不超过 15 天的 Amazon S3 对象
- 15-29 天的 Amazon S3 对象
- 30-44 天的 Amazon S3 对象
- 45-59 天的 Amazon S3 对象
- 60-74 天的 Amazon S3 对象
- 75-89 天的 Amazon S3 对象
- 90-119 天的 Amazon S3 对象
- 120-149 天的 Amazon S3 对象
- 150-179 天的 Amazon S3 对象
- 180-364 天的 Amazon S3 对象
- 365-729 天的 Amazon S3 对象
- 730 天及更长时间的 Amazon S3 对象

通常大约需要 30 天来观察访问模式以收集足够信息来获得分析结果。根据您的数据的独特访问模式，这可能需要 30 天以上的时间。但在配置一个筛选条件后，您首先会在 Amazon S3 控制台中看到 24 到 48 小时内基于该筛选条件的数据分析。您可以查看 Amazon S3 控制台中每天按对象期限组划分的对象访问的分析。

我的多少存储空间是不常访问的？

Amazon S3 控制台显示按预定义的对象期限组分组的访问模式，如以下示例所示。



每个期限组底部显示的经常访问或不常访问的文本采用的是与正在准备的生命周期策略推荐相同的逻辑。无论当前的累计访问率如何，在生命周期策略的推荐期限就绪后 (RecommendedObjectAge)，所有早于推荐期限的期限层均被标记为不常访问。此文本仅作为直观辅助，用于在生命周期创建过程中为您提供帮助。

## 如何导出存储类分析数据？

您可以选择使存储类分析将分析报告导出到逗号分隔值 (CSV) 平面文件。报告每天都会更新且基于您配置的对象期限组筛选条件。使用 Amazon S3 控制台时，您可以在创建筛选条件时选择导出报告选项。选择要导出的数据时，指定写入文件的目标存储桶和可选目标前缀。您可以将数据导出到不同账户中的目标存储桶。目标存储桶必须位于与您配置为进行分析的存储桶相同的区域中。

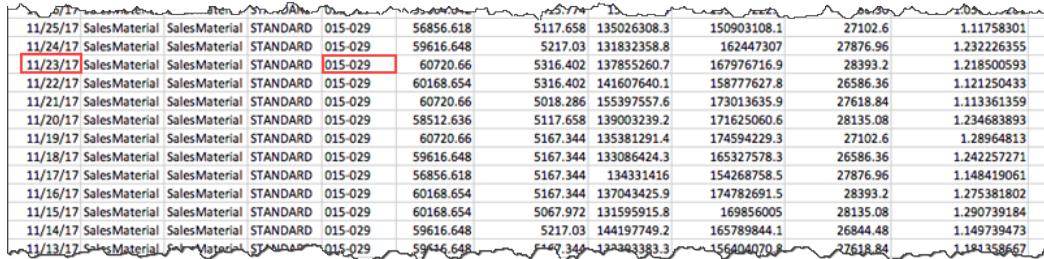
您必须在目标存储桶上创建存储桶策略以向 Amazon S3 授予验证哪些 AWS 账户拥有存储桶以及将对象写入定义位置的存储桶中的权限。有关策略示例，请参阅[向 Amazon S3 清单和 Amazon S3 分析功能授予权限 \(p. 310\)](#)。

在配置存储类分析报告后，您将在 24 小时后开始获得每日导出报告。之后，Amazon S3 会继续监控并提供每日导出。

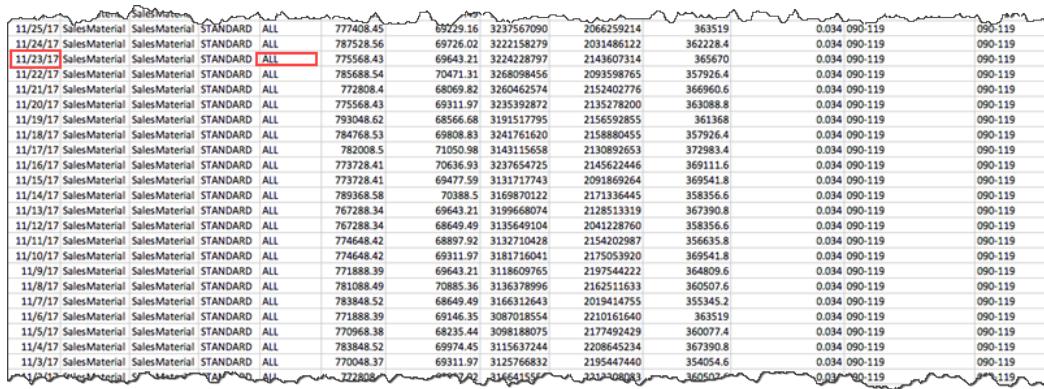
您可以在电子表格应用程序中打开此 CSV 文件，或将此文件导入其他应用程序中，如 [Amazon QuickSight](#)。有关将 Amazon S3 文件用于 Amazon QuickSight 的信息，请参阅 Amazon QuickSight 用户指南 中的[使用 Amazon S3 文件创建数据集](#)。

导出的文件中的数据在对象期限组内按日期进行排序，如以下示例所示。如果存储类为 STANDARD，则行还包含 ObjectAgeForSIATransition 和 RecommendedObjectAgeForSIATransition 列的数据。

Date	ConfigId	Filter	StorageClass	ObjectAge	ObjectCount	DataUploaded_Mb	Storage Mb	DataRetrieved_Mb	GetRequestCount	CumulativeAccessRatio	ObjectAgeForSIATransition	RecommendedObjectAgeForSIATransition
11/26/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39376.428	3610.516	100409322	106131482	17724.24	1.056989281		
11/25/17	SalesMaterial	SalesMaterial	STANDARD	000-014	37904.412	3411.772	90017538.84	100602072	18068.4	1.11758301		
11/24/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39744.432	3478.02	87888239.2	108298204.6	18584.64	1.232226355		
11/23/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40480.44	3544.268	91903507.12	111984477.9	18928.8	1.218500593		
11/22/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40112.436	3544.268	914405093.4	105851751.9	17724.24	1.121250433		
11/21/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40480.44	3345.524	103598371.7	115342424	18412.56	1.113361359		
11/20/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39008.424	3411.772	92668826.16	114416707.1	18756.72	1.234683893		
11/19/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40480.44	3444.896	90254194.28	116396152.9	18068.4	1.28964813		
11/18/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39744.432	3444.896	88724282.88	110218385.6	17724.24	1.242257271		
11/17/17	SalesMaterial	SalesMaterial	STANDARD	000-014	37904.412	3444.896	89554277.32	102845839	18584.64	1.148419061		
11/16/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40112.436	3444.896	91362283.92	116521794.3	18928.8	1.275381802		
11/15/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40112.436	3378.648	87730610.56	113237336.7	18756.72	1.290739184		
11/14/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39744.432	3478.02	96131832.8	110576562.4	17896.31	1.49739473		



报告结束时，对象期限组为 ALL。ALL 行包含当天所有使用期限组的累计总数，如以下示例所示。



下一部分介绍报告中使用的列。

## 导出的文件布局

下表描述导出文件的布局。

# Amazon S3 分析 REST API

下面是用于存储清单的 REST 操作。

- [DELETE Bucket analytics configuration](#)
- [GET Bucket analytics configuration](#)
- [List Bucket Analytics Configuration](#)
- [PUT Bucket analytics configuration](#)

# Amazon S3 清单

Amazon S3 清单是 Amazon S3 提供的用于帮助管理您的存储的工具之一。您可以出于业务、合规性和法规要求，使用它来审核和报告对象的复制和加密状态。您还可以使用 Amazon S3 清名单简化和加快业务工作流和大数据作业，它可以有计划地取代 Amazon S3 同步 List API 操作。

Amazon S3 清单每天或每周为 S3 存储桶或共享前缀（即，其名称以通用字符串开头的对象）提供用于列出您的对象及其对应元数据的逗号分隔值 (CSV) 或 [Apache 优化的行列式 \(ORC\)](#) 输出文件。有关 Amazon S3 清单定价的信息，请参阅 [Amazon S3 定价](#)。

您可以为存储桶配置多个清单列表。您可以配置要包含在该清单中的对象元数据，是列出所有对象版本还是仅列出当前版本，要存储清单列表文件输出的位置以及是每天还是每周生成该清单。您也可以指定对清单列表文件进行加密。

您可以通过 Amazon Athena、Amazon Redshift Spectrum 和其他工具（例如 [Presto](#)、[Apache Hive](#) 和 [Apache Spark](#)）使用标准 SQL 来查询 Amazon S3 清单。使用 Athena 对您的清单文件运行查询非常简单。您可以在提供 Athena 的所有区域中使用 Athena 查询 Amazon S3 清单。

## 主题

- [我如何设置 Amazon S3 清单？\(p. 235\)](#)
- [Amazon S3 清单中包含了什么？\(p. 237\)](#)
- [清单列表位于何处？\(p. 238\)](#)
- [清单完成时我如何知道？\(p. 240\)](#)
- [使用 Amazon Athena 查询清单 \(p. 240\)](#)
- [Amazon S3 清单 REST API \(p. 241\)](#)

## 我如何设置 Amazon S3 清单？

本部分介绍了如何设置清单，包括有关清单源存储桶和目标存储桶的详细信息。

### Amazon S3 清名单源存储桶和目标存储桶

由清单列出其对象的存储桶称为源存储桶。在其中存储清单列表文件的存储桶称为目标存储桶。

#### 源存储桶

清单列出了源存储桶中存储的对象。您可以获取整个存储桶的清单列表或按（对象键名称）前缀筛选过的清单列表。

#### 源存储桶：

- 包含在清单中列出的对象。
- 包含清单的配置。

#### 目标存储桶

Amazon S3 清单列表文件将被写入目标存储桶。要对目标存储桶内公共位置中的所有清单列表文件进行分组，您可以在清单配置中指定目标（对象键名称）前缀。

#### 目标存储桶：

- 包含清单文件列表。

- 包含 Manifest 文件，其中列出了存储在目标存储桶中的所有文件清单列表。有关更多信息，请参阅 [什么是清单 Manifest？\(p. 239\)](#)。
- 必须具有向 Amazon S3 授予验证存储桶的所有权的权限和将文件写入存储桶的权限的存储桶策略。
- 必须与源存储桶位于同一 AWS 区域。
- 可以与源存储桶相同。
- 可以由与拥有源存储桶的账户不同的 AWS 账户拥有。

## 设置 Amazon S3 清单

Amazon S3 清单将帮助您按预定的计划创建 S3 存储桶中的对象的列表，由此管理存储。您可以为存储桶配置多个清单列表。清单列表将发布到目标存储桶中的 CSV 或 ORC 文件。

设置清单的最简单方法是使用 AWS 管理控制台，不过您也可以使用 REST API、AWS CLI 或 AWS 开发工具包。控制台将为您执行以下过程的第一步：将存储桶策略添加到目标存储桶。

### 为 S3 存储桶设置 Amazon S3 清单

#### 1. 为目标存储桶添加存储桶策略。

您必须在目标存储桶上创建存储桶策略，以向 Amazon S3 授予将对象写入已定义位置的存储桶的权限。有关策略示例，请参阅[向 Amazon S3 清单和 Amazon S3 分析功能授予权限 \(p. 310\)](#)。

#### 2. 配置一个清单以列出源存储桶中的对象并将该列表发布到目标存储桶。

当您为源存储桶配置清单列表时，您应指定用于存储该列表的目标存储桶，以及是每天还是每周生成一次列表。您还可以配置要包含的对象元数据以及是列出所有对象版本还是仅列出最新版本。

您可以指定使用 Amazon S3 托管密钥 (SSE-S3) 或 AWS KMS 托管密钥 (SSE-KMS) 来加密清单列表文件。有关 SSE-S3 和 SSE-KMS 的更多信息，请参阅[使用服务器端加密保护数据 \(p. 345\)](#)。如果您打算使用 SSE-KMS 加密，请参阅步骤 3。

- 有关如何使用控制台配置清单列表的信息，请参阅[如何配置 Amazon S3 清单？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。
- 要使用 Amazon S3 API 来配置清单列表，请使用 [PUT Bucket inventory configuration REST API](#)，或者使用 AWS CLI 或 AWS 开发工具包中的等效工具。

#### 3. 要使用 SSE-KMS 加密清单列表文件，请授予 Amazon S3 使用 AWS KMS 密钥的权限。

您可以使用 AWS 管理控制台、REST API、AWS CLI 或 AWS 开发工具包，为清单列表文件配置加密。无论选择哪种方式，您都必须授予 Amazon S3 相应权限以使用 AWS KMS 客户主密钥 (CMK) 加密清单文件。可通过修改用于加密清单文件的 AWS KMS CMK 的密钥策略，向 Amazon S3 授予权限。有关更多信息，请参阅下一部分：[授予 Amazon S3 使用您的 AWS KMS 密钥进行加密的权限 \(p. 236\)](#)。

## 授予 Amazon S3 使用您的 AWS KMS 密钥进行加密的权限

您必须通过密钥策略授予 Amazon S3 权限，以使用您的 AWS KMS 密钥进行加密。以下过程介绍如何使用 AWS Identity and Access Management (IAM) 控制台修改用于加密清单文件的 AWS KMS CMK 的密钥策略。

### 授予使用您的 AWS KMS 密钥进行加密的权限

1. 使用拥有 AWS KMS CMK 的 AWS 账户登录 AWS 管理控制台，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧导航窗格中，选择 Encryption keys。
3. 对于 Region，选择适当的 AWS 区域。请勿使用导航栏中的区域选择器 (右上角)。
4. 选择您在加密清单时要使用的 CMK 的别名。

5. 在页面的 Key Policy 部分中，选择 Switch to policy view。
6. 使用 Key Policy 编辑器，将以下密钥策略插入到现有策略中，然后选择 Save Changes。您可能希望将策略复制到现有策略的末尾。

```
{  
    "Sid": "Allow Amazon S3 use of the key",  
    "Effect": "Allow",  
    "Principal": {  
        "Service": "s3.amazonaws.com"  
    },  
    "Action": [  
        "kms:GenerateDataKey*"  
    ],  
    "Resource": "*"  
}
```

您还可以使用 AWS KMS PUT 密钥策略 API [PutKeyPolicy](#)，将密钥策略复制到用于加密清单文件的 CMK 的密钥策略。有关创建和编辑 AWS KMS CMK 的更多信息，请参阅 AWS Key Management Service Developer Guide 中的 [入门](#)。

## Amazon S3 清单中包含了什么？

清单列表文件包含源存储桶中对象的列表以及每个对象的元数据。清单列表将在目标存储桶中存储为使用 GZIP 压缩的 CSV 文件，或者经过 Apache 优化的行列式 (ORC) 文件。

清单列表包含 S3 存储桶中的对象的列表以及每个列出的对象的以下元数据：

- Bucket name – 清单所针对的存储桶的名称。
- Key name – 唯一地标识存储桶中的对象的对象键名称 (或键)。使用 CSV 文件格式时，键名称采用 URL 编码形式，必须解码然后才能使用。
- Version ID – 对象版本 ID。在存储桶上启用版本控制后，Amazon S3 会为添加到存储桶的对象指定版本号。有关更多信息，请参阅 [对象版本控制 \(p. 94\)](#)。(如果列表仅针对对象的当前版本，则不包含此字段。)
- IsLatest – 如果对象的版本为最新，则设置为 True。(如果列表仅针对对象的当前版本，则不包含此字段。)
- Size – 对象大小 (以字节为单位)。
- Last modified date – 对象创建日期或上次更新日期 (以较晚者为准)。
- ETag – 实体标签是对象的哈希。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 可能是也可能不是对象数据的 MD5 摘要。是与不是取决于对象的创建方式和加密方式。
- Storage class – 用于存储对象的存储类。有关更多信息，请参阅 [存储类别 \(p. 90\)](#)。
- Multipart upload flag – 如果对象以分段上传形式上传，则设置为 True。有关更多信息，请参阅 [分段上传概述 \(p. 155\)](#)。
- Delete marker – 如果对象是删除标记，则设置为 True。有关更多信息，请参阅 [对象版本控制 \(p. 94\)](#)。(如果列表仅针对对象的当前版本，则不包含此字段。)
- Replication status – 设置为 PENDING、COMPLETED、FAILED 或 REPLICA. 有关更多信息，请参阅 [查看跨区域复制状态 \(p. 472\)](#)。
- Encryption status – 设置为 SSE-S3、SSE-C、SSE-KMS 或 NOT-SSE。SSE-S3、SSE-KMS 以及客户提供密钥的 SSE (SSE-C) 的服务器端加密状态。NOT-SSE 状态表示对象未使用服务器端加密进行加密。有关更多信息，请参阅 [使用加密保护数据 \(p. 345\)](#)。

下面是在电子表格应用程序中打开的示例 CSV 清单列表。所示的标题行仅用于帮助阐述示例；不会包含在实际列表中。

Bucket	Key	VersionId	IsLatest	IsDeleteMarker	Size	LastModifiedDate	Etag	StorageClass	MultipartUploaded	ReplicationStatus
example-bucket	object1			FALSE	2.4E+08	2016-08-11T01:19	e80d8eda4	STANDARD	TRUE	
example-bucket	object2			FALSE	0	2016-08-10T22:23	d41d8cd98	STANDARD	FALSE	
example-bucket	object3			FALSE	9	2016-08-10T20:18	9090441e4	STANDARD_IA	FALSE	
example-bucket	object4			FALSE	9	2016-08-10T20:36	9090441e4	STANDARD_IA	FALSE	
example-bucket	object1			FALSE	22	2016-08-10T20:35	9090441e4	STANDARD	FALSE	
example-bucket	object1			FALSE	2016-08-10T20:34	9090441e4	REDUCED_RED	FALSE		
example-bucket	object1			FALSE	2016-08-10T21:13	9090441e4	GLACIER	FALSE		

我们建议创建一个删除旧清单列表的生命周期策略。有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。

## 清单一致性

并非所有对象都会显示每个清单列表中。清单列表提供了新对象和覆盖的 PUT 的最终一致性，并提供了 DELETE 的最终一致性。清单列表是存储桶项的滚动快照，这些项最终是一致的（即，列表可能不包含最近添加或删除的对象）。

要在对对象执行操作之前验证对象的状态，我们建议您执行 HEAD Object REST API 请求以检索对象的元数据，或在 Amazon S3 控制台中检查对象的属性。您还可以使用 AWS CLI 或 AWS 开发工具包检查对象元数据。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [HEAD 对象](#)。

## 清单列表位于何处？

在清单列表发布后，Manifest 文件将发布到目标存储桶中的以下位置。

```
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.checksum
destination-prefix/source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/symlink.txt
```

- **destination-prefix** 是清单配置中设置的（对象键名称）前缀，可用于对目标存储桶中的公共位置的所有清单列表文件进行分组。
- **source-bucket** 是清单列表所针对的源存储桶。添加它是为了防止在将不同的源存储桶中的多个清单发送到同一目标存储桶时发生冲突。
- **config-ID** 为了防止从同一源存储桶发送到同一目标存储桶的多个清单报告发生冲突而添加。
- **YYYY-MM-DDTHH-MMZ** 是时间戳，包含生成清单报告时开始扫描存储桶的开始时间和日期；例如，2016-11-06T21-32Z。报告中不会出现在此时间戳之后添加的存储。
- **manifest.json** 是 Manifest 文件。
- **manifest.checksum** 是 **manifest.json** 文件内容的 MD5。
- **symlink.txt** 是与 Apache Hive 兼容的 Manifest 文件。

清单列表每天或每周发布到目标存储桶中的以下位置。

```
destination-prefix/source-bucket/data/example-file-name.csv.gz
...
destination-prefix/source-bucket/data/example-file-name-1.csv.gz
```

- **destination-prefix** 是清单配置中设置的（对象键名称）前缀。它可用于对目标存储桶公共位置中的所有清单列表文件进行分组。
- **source-bucket** 是清单列表所针对的源存储桶。添加它是为了防止在将不同的源存储桶中的多个清单发送到同一目标存储桶时发生冲突。

- *example-file-name.csv.gz* 是 CSV 清单文件之一。ORC 清单名称以文件扩展名 .orc 结尾。

## 什么是清单 Manifest ?

Manifest 文件 manifest.json 和 symlink.txt 描述清单文件的位置。每次交付新的清单列表时，它均带有一组新的 Manifest 文件。

manifest.json 文件中包含的每个 Manifest 均提供了有关清单的元数据和其他基本信息。这些信息包含：

- 源存储桶名称
- 目标存储桶名称
- 清单版本
- 以纪元日期格式创建时间戳，包含生成清单报告时开始扫描存储桶的开始时间和日期
- 清单文件的格式和架构
- 目标存储桶中清单文件的实际列表

每当写入 manifest.json 文件后，它都会附带一个 manifest.checksum 文件（作为 manifest.json 文件内容的 MD5）。

下面是 CSV 格式清单的 manifest.json 文件中的 Manifest 示例。

```
{  
    "sourceBucket": "example-source-bucket",  
    "destinationBucket": "example-inventory-destination-bucket",  
    "version": "2016-11-30",  
    "creationTimestamp" : "1514944800000",  
    "fileFormat": "CSV",  
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker, Size,  
LastModifiedDate, ETag, StorageClass, MultipartUploaded, ReplicationStatus",  
    "files": [  
        {  
            "key": "Inventory/example-source-bucket/2016-11-06T21-32Z/  
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",  
            "size": 2147483647,  
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc",  
            "inventoriedRecord": 58050695  
        }  
    ]  
}
```

下面是 ORC 格式清单的 manifest.json 文件中的 Manifest 示例。

```
{  
    "sourceBucket": "example-source-bucket",  
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",  
    "version": "2016-11-30",  
    "creationTimestamp" : "1514944800000",  
    "fileFormat": "ORC",  
    "fileSchema":  
    "struct<bucket:string,key:string,version_id:string,is_latest:boolean,is_delete_marker:boolean,size:bigint>  
    "files": [  
        {  
            "key": "inventory/example-source-bucket/data/  
d794c570-95bb-4271-9128-26023c8b4900.orc",  
            "size": 56291,  
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"  
        }  
    ]  
}
```

}

该 `symlink.txt` 文件是一个与 Apache Hive 兼容的 Manifest 文件，使 Hive 能够自动发现清单文件及其关联的数据文件。与 Hive 兼容的 Manifest 文件可以用于与 Hive 兼容的服务 Athena 和 Amazon Redshift Spectrum。它还可用于与 Hive 兼容的应用程序，包括 [Presto](#)、[Apache Hive](#)、[Apache Spark](#) 以及许多其他应用程序。

#### Important

`symlink.txt` Apache Hive 兼容的 Manifest 文件当前不适用于 AWS Glue。  
ORC 格式的清单文件不适用于 [Apache Hive](#) 和 [Apache Spark](#)。

## 清单完成时我如何知道？

您可以设置 Amazon S3 事件通知以在创建清单校验和文件后接收通知，该通知将指示库存列表已添加到目标存储桶。清单是目标位置的所有库存列表的最新列表。

Amazon S3 可以将事件发布到 Amazon Simple Notification Service (Amazon SNS) 主题、Amazon Simple Queue Service (Amazon SQS) 队列或 AWS Lambda 函数。有关更多信息，请参阅 [配置 Amazon S3 事件通知 \(p. 425\)](#)。

以下通知配置定义了新添加到目标存储桶的所有 `manifest.checksum` 文件将由 AWS Lambda `cloud-function-list-write` 处理。

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>1</Id>
    <Filter>
      <S3Key>
        <FilterRule>
          <Name>prefix</Name>
          <Value>destination-prefix/source-bucket</Value>
        </FilterRule>
        <FilterRule>
          <Name>suffix</Name>
          <Value>checksum</Value>
        </FilterRule>
      </S3Key>
    </Filter>
    <Cloudcode>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</Cloudcode>
    <Event>s3:ObjectCreated:*</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

有关更多信息，请参阅 AWS Lambda Developer Guide 中的[将 AWS Lambda 与 Amazon S3 一起使用](#)。

## 使用 Amazon Athena 查询清单

您可以在提供 Athena 的所有区域中通过 Amazon Athena 使用标准 SQL 来查询 Amazon S3 清单。要检查 AWS 区域可用性，请参阅 [AWS 区域表](#)。

Athena 可以查询 ORC 或 CSV 格式的 Amazon S3 清单文件。当您使用 Athena 查询清单时，我们建议您使用 ORC 格式的清单文件，而不是 CSV。ORC 提供了更快的查询性能并能够降低查询成本。ORC 是自我描述、可感知类型的列式文件格式，专为 [Apache Hadoop](#) 而设计。列式格式允许读取器仅读取、解压缩并处理当前查询所需的列。所有 AWS 区域均提供 ORC 格式的 Amazon S3 清单。

## 开始使用 Athena 查询 Amazon S3 清单

1. 创建一个 Athena 表。有关创建表的信息，请参阅 Amazon Athena 用户指南 中的[入门](#)。

下面的示例查询在清单报告中包含所有可选字段。删除您的清单没有选择的所有可选字段，以便查询对应于您的清单的选定字段。此外，您必须使用您的存储桶名称和位置。位置指向清单的目标路径；例如，`s3://destination-prefix/source-bucket/config-ID/hive`。

```
CREATE EXTERNAL TABLE your-table-name(  
    `bucket` string,  
    key string,  
    version_id string,  
    is_latest boolean,  
    is_delete_marker boolean,  
    size bigint,  
    last_modified_date timestamp,  
    e_tag string,  
    storage_class string,  
    is multipart_uploaded boolean,  
    replication_status string,  
    encryption_status string  
)  
PARTITIONED BY (dt string)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'  
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'  
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'  
LOCATION 's3://destination-prefix/source-bucket/config-ID/hive';
```

2. 要向表中添加新的清单列表，请使用以下 MSCK REPAIR TABLE 命令。

```
MSCK REPAIR TABLE your-table-name;
```

3. 在执行前两个步骤后，您可以对自己的清单运行即席查询，如以下示例中所示。

```
SELECT encryption_status, count(*) FROM your-table-name GROUP BY encryption_status;
```

有关使用 Athena 的更多信息，请参阅 [Amazon Athena 用户指南](#)。

## Amazon S3 清单 REST API

下面是用于 Amazon S3 清单的 REST 操作。

- [DELETE Bucket inventory configuration](#)
- [GET Bucket inventory configuration](#)
- [List Bucket Inventory Configuration](#)
- [PUT Bucket inventory configuration](#)

# 管理对 Amazon S3 资源的访问权限

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，`lifecycle` 配置和 `website` 配置）。只有资源拥有者，即创建该资源的 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授予他人访问权限。

Amazon S3 提供的访问策略选项大致可分为基于资源的策略和用户策略两类。附加到资源（存储桶和对象）的访问策略称为基于资源的策略。例如，存储桶策略和访问控制列表（ACL）就是基于资源的策略。您也可以将访问策略附加到您账户中的用户。这些策略称为用户策略。您可以选择使用基于资源的策略、用户策略或这些策略的某种组合来管理您的 Amazon S3 资源权限。本介绍性主题提供了管理权限的一般准则。

建议您首先阅读访问控制概述主题。有关更多信息，请参阅 [Amazon S3 资源访问权限管理介绍 \(p. 242\)](#)。有关特定访问策略选项的更多信息，请参阅以下主题：

- [使用存储桶策略和用户策略 \(p. 279\)](#)
- [使用 ACL 管理访问 \(p. 333\)](#)

## Amazon S3 资源访问权限管理介绍

### 主题

- [访问管理概述 \(p. 243\)](#)
- [Amazon S3 如何对请求授权 \(p. 248\)](#)
- [有关使用可用访问策略选项的准则 \(p. 252\)](#)
- [示例演练：管理对 Amazon S3 资源的访问 \(p. 255\)](#)

本部分中的主题提供 Amazon S3 资源访问权限管理概述，并就何时使用何种访问控制方法提供指导。本主题还提供了介绍性示例演练。建议您按顺序阅读这些主题。

## 访问管理概述

### 主题

- [Amazon S3 资源 \(p. 243\)](#)
- [资源操作 \(p. 244\)](#)
- [管理对资源的访问 \(访问策略选项\) \(p. 244\)](#)
- [我该使用何种访问控制方法? \(p. 247\)](#)
- [相关主题 \(p. 247\)](#)

在授予权限时，您要决定谁获得权限，获得对哪些 Amazon S3 资源的权限，以及您允许对这些资源执行的具体操作。

### Amazon S3 资源

存储桶和对象是主要的 Amazon S3 资源，它们都有关联的子资源。例如，存储桶子资源包括：

- `lifecycle` – 存储生命周期配置信息 (请参阅 [对象生命周期管理 \(p. 104\)](#))。
- `website` – 为网站托管配置存储桶时存储网站配置信息 (请参阅 [在 Amazon S3 上托管静态网站 \(p. 401\)](#))。
- `versioning` – 存储版本控制配置 (请参阅 [PUT Bucket 版本控制](#))。
- `policy` 和 `acl` (访问控制列表) – 保存存储桶的访问权限信息。
- `cors` (跨源资源共享) – 支持配置存储桶以允许跨源请求 (请参阅 [跨源资源共享 \(CORS\) \(p. 133\)](#))。
- `logging` – 使您能够请求 Amazon S3 保存存储桶访问日志。

对象子资源包括：

- `acl` – 存储对象访问权限列表。本主题讨论如何使用这一子资源管理对象权限 (请参阅 [使用 ACL 管理访问 \(p. 333\)](#))。
- `restore` – 支持临时还原存档对象 (请参阅 [POST Object 还原](#))。Glacier 存储类中的对象是存档对象。要访问这类对象，必须首先启动还原请求，这会还原存档对象的副本。在请求中，指定您希望还原的副本保留的天数。有关归档对象的更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。

### 关于资源拥有者

默认情况下，所有 Amazon S3 资源都是私有的。只有资源拥有者才能访问资源。资源拥有者是指创建资源的 AWS 账户。例如：

- 用于创建存储桶和对象的 AWS 账户拥有这些资源。
- 如果您在 AWS 账户中创建了一个 AWS Identity and Access Management (IAM) 用户，您的 AWS 账户就是父级拥有者。如果该 IAM 用户上传一个对象，则该用户所属的父账户拥有该对象。
- 存储桶拥有者可以向其他 AWS 账户 (或其他账户中的用户) 授予上传对象的跨账户权限。在这种情况下，上传对象的 AWS 账户拥有这些对象。存储桶拥有者对其他账户拥有的对象没有权限，以下情况除外：
  - 账单由存储桶拥有者支付。存储桶拥有者可以拒绝对任何对象的访问，或删除存储桶中的任何对象，而无论它们的拥有者是谁。
  - 存储桶拥有者可以存档任何对象或还原存档对象，而无论它们的拥有者是谁。存档是指用于存储对象的存储类。有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。

## Important

AWS 建议不要使用 AWS 账户的根凭证发起请求。而应创建一个 IAM 用户并授予该用户完全访问权限。我们将这些用户称为管理员用户。您可以使用管理员用户凭证而不是您账户的根凭证来与 AWS 交互和执行任务，例如创建存储桶、创建用户和为用户授予权限。有关更多信息，请转到 AWS General Reference 中的[根账户凭证与 IAM 用户凭证](#)和 IAM 用户指南中的[IAM 最佳实践](#)。

下图所示为一个拥有资源、IAM 用户、存储桶和对象的 AWS 账户。



## 资源操作

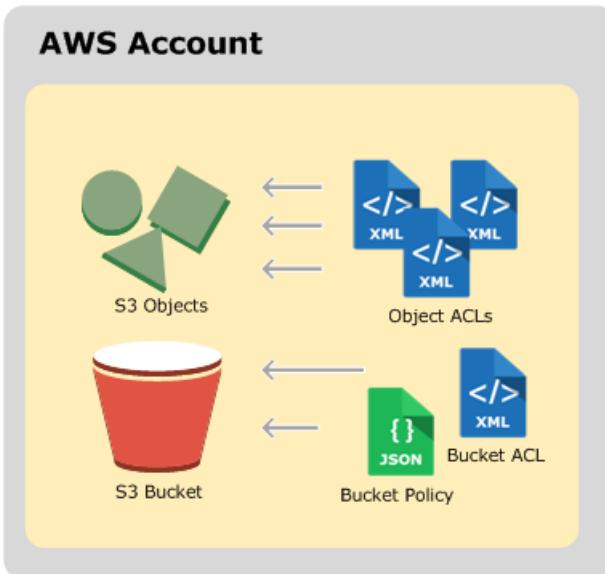
Amazon S3 提供一组操作用来处理 Amazon S3 资源。有关可用操作的列表，请转至 Amazon Simple Storage Service API Reference 中的[在存储桶上的操作](#)和[在对象上的操作](#)。

## 管理对资源的访问 (访问策略选项)

管理访问是指通过编写访问策略向他人 (AWS 账户和用户) 授予执行资源操作的权限。例如，您可以对 AWS 账户中的一个用户授予 PUT Object 权限，使该用户可以向您的存储桶上传对象。除了向单个的用户和账户授予权限之外，您还可以向每个人 (也称为匿名访问) 或所有已验证身份的用户 (拥有 AWS 凭证的用户) 授予权限。例如，当您将存储桶配置为网站时，可能希望通过向每个人授予 GET Object 权限将对象公开。

访问策略描述了谁可以访问哪些内容。您可以将访问策略与资源 (存储桶和对象) 或用户相关联。相应地，您可以将可用的 Amazon S3 访问策略如下分类：

- **基于资源的策略 – 存储桶策略和访问控制列表 (ACL)** 是基于资源的策略，因为您将它们附加到 Amazon S3 资源。



- **ACL** – 每个存储桶和对象都有关联的 ACL。ACL 是一个指定被授权者和所授予权限的授权列表。您可使用 ACL 向其他 AWS 账户授予基本的读/写权限。ACL 使用特定于 Amazon S3 的 XML 架构。

下面是一个示例存储桶 ACL。该 ACL 中的授权显示一个存储桶拥有者具有完全控制权限。

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

存储桶和对象 ACL 使用相同的 XML 架构。

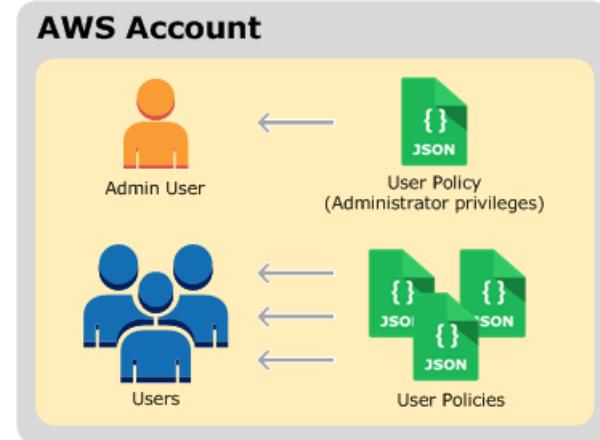
- **存储桶策略** – 对于存储桶，您可以通过添加存储桶策略向其他 AWS 账户或 IAM 用户授予对相应存储桶及其中对象的权限。任何对象权限都仅应用于存储桶拥有者创建的对象。存储桶策略补充（在很多情况下取代）基于 ACL 的访问策略。

下面是一个示例存储桶策略。您使用 JSON 文件来表示存储桶策略（和用户策略）。该策略授予对一个存储桶中所有对象的匿名读取权限。该存储桶策略有一条语句，允许对名为 examplebucket 的存储桶中的对象执行 s3:GetObject 操作（读取权限）。通过使用通配符 (\*) 指定 principal，该策略授予匿名访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::examplebucket/*"
    }
  ]
}
```

```
        "Action": ["s3:GetObject"],
        "Resource": ["arn:aws:s3:::examplebucket/*"]
    }
}
```

- 用户策略 – 您可以使用 IAM 管理对 Amazon S3 资源的访问。您可以在您的账户中创建 IAM 用户、组和角色，并通过附加访问策略授予它们对包括 Amazon S3 在内的 AWS 资源的访问权限。



有关 IAM 的更多信息，请参阅 [AWS Identity and Access Management \(IAM\)](#) 产品详细信息页面。

下面是一个用户策略示例。因为 IAM 用户策略附加到用户，所以不能在其中授予匿名权限。示例策略允许其附加到的关联用户对存储桶及其中的对象执行六种不同的 Amazon S3 操作。您可以将此策略附加到一个具体的 IAM 用户、组或角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleStatement1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3>ListBucket",
                "s3>DeleteObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*",
                "arn:aws:s3:::examplebucket"
            ]
        },
        {
            "Sid": "ExampleStatement2",
            "Effect": "Allow",
            "Action": "s3>ListAllMyBuckets",
            "Resource": "*"
        }
    ]
}
```

当 Amazon S3 收到请求时，它必须评估所有访问策略以决定是授权还是拒绝该请求。有关 Amazon S3 如何评估这些策略的更多信息，请参阅 [Amazon S3 如何对请求授权 \(p. 248\)](#)。

## 我该使用何种访问控制方法？

有了可以编写访问策略的各种方法，您将面临以下问题：

- 我应在何时使用何种访问控制方法？例如，要授予存储桶权限，应当使用存储桶策略还是存储桶 ACL？我拥有一个存储桶和其中的对象。我应当使用基于资源的访问策略还是 IAM 用户策略？如果使用基于资源的访问策略，我应当使用存储桶策略还是对象 ACL 来管理对象权限？
- 我拥有一个存储桶，但其中的对象并不全都属于我。其他人所拥有的对象的访问权限是如何管理的？
- 如果我组合使用这些访问策略方法授予访问权限，Amazon S3 如何确定某用户是否有权执行请求的操作？

以下各节介绍这些访问控制备选方法、Amazon S3 如何评估访问控制机制以及何时使用何种访问控制方法。还提供了示例演练。

[Amazon S3 如何对请求授权 \(p. 248\)](#)

[有关使用可用访问策略选项的准则 \(p. 252\)](#)

[示例演练：管理对 Amazon S3 资源的访问 \(p. 255\)](#)

## 相关主题

我们建议您先阅读有关管理 Amazon S3 资源访问的各种方法的介绍性主题。有关更多信息，请参阅 [Amazon S3 资源访问权限管理介绍 \(p. 242\)](#)。然后再通过以下主题了解有关特定访问策略选项的更多信息。

- [使用存储桶策略和用户策略 \(p. 279\)](#)
- [使用 ACL 管理访问 \(p. 333\)](#)

## Amazon S3 如何对请求授权

### 主题

- [相关主题 \(p. 249\)](#)
- [Amazon S3 如何对存储桶操作请求进行授权 \(p. 249\)](#)
- [Amazon S3 如何授权对象操作的请求 \(p. 251\)](#)

当 Amazon S3 收到请求 (例如，存储桶或对象操作) 时，它首先验证请求者是否拥有必要的权限。Amazon S3 对所有相关访问策略、用户策略和基于资源的策略 (存储桶策略、存储桶 ACL、对象 ACL) 进行评估，以决定是否对该请求进行授权。以下是一些示例方案：

- 如果请求者是 IAM 用户，则 Amazon S3 必须确定该用户所属的父 AWS 账户已授予该用户执行操作的必要权限。此外，如果请求是要执行存储桶操作 (例如，请求列出存储桶内容)，则 Amazon S3 必须验证存储桶拥有者是否已为请求者授予执行该操作的权限。

### Note

要针对某一资源执行特定操作，IAM 用户需要从其所属的父 AWS 账户以及拥有该资源的 AWS 账户双方获得权限。

- 如果请求是要针对存储桶拥有者未拥有的对象执行某一操作，则除了确保请求者拥有来自对象拥有者的权限外，Amazon S3 还必须检查存储桶策略以确保存储桶拥有者未对该对象设置显式拒绝。

### Note

存储桶拥有者 (支付账单者) 可以显式拒绝对存储桶中的对象的访问，而不论谁拥有该对象。存储桶拥有者还可以删除存储桶中的任何对象。

为了确定请求者是否拥有执行特定操作的权限，Amazon S3 会在收到请求时按顺序执行以下操作：

1. 在运行时将所有相关访问策略 (用户策略、存储桶策略、ACL) 转换为一组策略以进行评估。
2. 通过以下步骤评估生成的策略集。在每个步骤中，Amazon S3 都会基于特定上下文机构来评估上下文中的策略子集。
  - a. 用户上下文 – 在用户上下文中，用户所属的父账户是上下文机构。

Amazon S3 对父账户拥有的策略子集进行评估。该子集包括父级附加到该用户的用户策略。如果父级也拥有请求中的资源 (存储桶、对象)，则 Amazon S3 还会同时评估相应资源策略 (存储桶策略、存储桶 ACL 和对象 ACL)。

用户必须从父账户获得执行该操作的权限。

只有在 AWS 账户中的用户发出请求的情况下，此步骤才适用。如果请求是使用 AWS 账户的根凭证发出的，则 Amazon S3 跳过此步骤。

- b. 存储桶上下文 – 在存储桶上下文中，Amazon S3 评估拥有该存储桶的 AWS 账户所拥有的策略。

如果请求是针对存储桶操作发出的，则请求者必须拥有来自存储桶拥有者的权限。如果请求是针对对象发出的，则 Amazon S3 会评估由存储桶拥有者拥有的所有策略，以检查存储桶拥有者是否未显式拒绝对该对象的访问。如果设置了显式拒绝，则 Amazon S3 不对请求授权。

- c. 对象上下文 – 如果请求是针对对象发出的，则 Amazon S3 对由对象拥有者拥有的策略子集进行评估。

以下各部分进行了详细说明并提供了示例：

- [Amazon S3 如何对存储桶操作请求进行授权 \(p. 249\)](#)
- [Amazon S3 如何授权对象操作的请求 \(p. 251\)](#)

## 相关主题

建议您首先阅读介绍性主题，这些主题讲解了用于管理对 Amazon S3 资源的访问的选项。有关更多信息，请参阅 [Amazon S3 资源访问权限管理介绍 \(p. 242\)](#)。然后再通过以下主题了解有关特定访问策略选项的更多信息。

- 使用存储桶策略和用户策略。 ([p. 279](#))
- 使用 ACL 管理访问 ([p. 333](#))

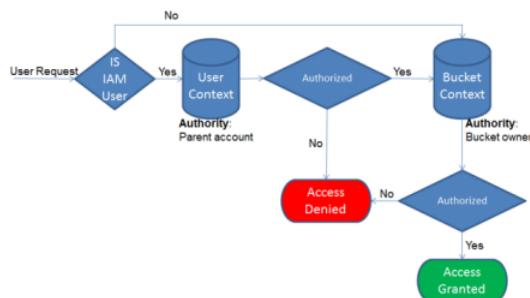
## Amazon S3 如何对存储桶操作请求进行授权

当 Amazon S3 收到存储桶操作请求时，Amazon S3 会将基于资源的所有相关权限（存储桶策略、存储桶访问控制列表（ACL））和 IAM 用户策略（如果请求来自用户）转换为一组用于在运行时进行评估的策略。然后它将通过一系列步骤根据特定上下文（用户上下文或存储桶上下文）来评估生成的策略集。

1. 用户上下文 – 如果请求者是 IAM 用户，则该用户必须拥有来自其所属的父 AWS 账户的权限。在此步骤中，Amazon S3 将评估由父账户（也称为上下文机构）拥有的一个策略子集。该策略子集包含父账户附加到该用户的用户策略。如果父级也拥有请求中的资源（在本例中为存储桶），则 Amazon S3 还会同时评估相应资源策略（存储桶策略和存储桶 ACL）。只要发出存储桶操作的请求，服务器访问日志就会记录请求者的规范用户 ID。有关更多信息，请参阅 [Amazon S3 服务器访问日志记录 \(p. 506\)](#)。
2. 存储桶上下文 – 请求者必须拥有来自存储桶拥有者的权限才能执行特定存储桶操作。在此步骤中，Amazon S3 对由拥有该存储桶的 AWS 账户拥有的策略子集进行评估。

存储桶拥有者可通过使用存储桶策略或存储桶 ACL 来授予权限。请注意，如果拥有存储桶的 AWS 账户也是 IAM 用户的父账户，则该 AWS 账户可以在用户策略中配置存储桶权限。

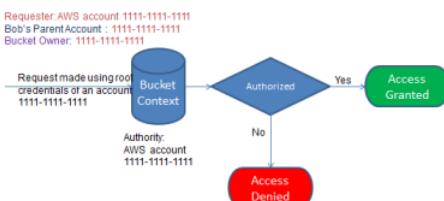
下面是基于上下文对存储桶操作进行评估的图解说明。



下面的示例演示了评估逻辑。

### 示例 1：由存储桶拥有者请求的存储桶操作

在此示例中，存储桶拥有者使用 AWS 账户的根凭证发送存储桶操作请求。

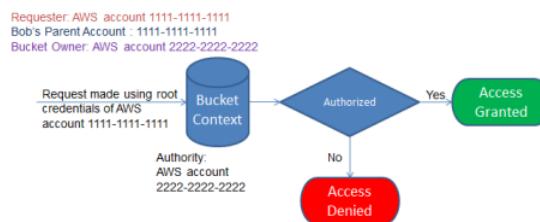


Amazon S3 通过以下方式执行上下文评估：

1. 由于请求是通过使用 AWS 账户的根凭证发出的，因此不评估用户上下文。
2. 在存储桶上下文中，Amazon S3 检查存储桶策略以确定请求者是否拥有执行该操作的权限。Amazon S3 对该请求进行授权。

## 示例 2：由不是存储桶拥有者的 AWS 账户请求的存储桶操作

在此示例中，使用 AWS 账户 1111-1111-1111 的根凭证发出请求，请求执行由 AWS 账户 2222-2222-2222 拥有的存储桶操作。该请求不涉及任何 IAM 用户。

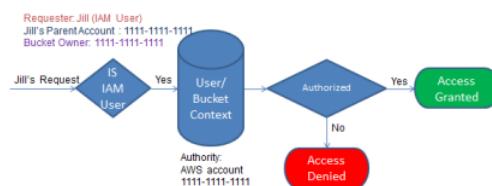


在此情况下，Amazon S3 通过以下方式评估上下文：

1. 由于该请求是使用 AWS 账户的根凭证发出的，因此不评估用户上下文。
2. 在存储桶上下文中，Amazon S3 检查存储桶策略。如果存储桶拥有者 (AWS 账户 2222-2222-2222) 尚未授权 AWS 账户 1111-1111-1111 执行请求的操作，则 Amazon S3 拒绝该请求。否则，Amazon S3 授权该请求并执行该操作。

## 示例 3：由父 AWS 账户同时也是存储桶拥有者的 IAM 用户请求的存储桶操作

在此示例中，请求由 AWS 账户 1111-1111-1111 中的 IAM 用户 Jill 发送，该账户同时也拥有该存储桶。



Amazon S3 执行以下上下文评估：

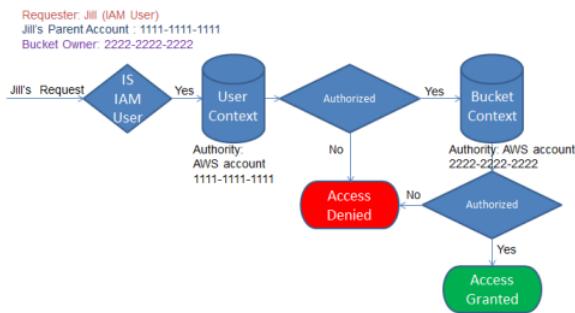
1. 由于该请求来自用户上下文中的 IAM 用户，Amazon S3 将对属于父 AWS 账户的所有策略进行评估，以确定 Jill 是否拥有执行该操作的权限。

在此示例中，该用户所属的父 AWS 账户 1111-1111-1111 也是存储桶拥有者。因此，除用户策略外，Amazon S3 还要评估同一上下文中的存储桶策略和存储桶 ACL，因为它们属于同一账户。

2. 由于 Amazon S3 已将存储桶策略和存储桶 ACL 作为用户上下文的一部分进行了评估，因此它不会评估存储桶上下文。

## 示例 4：由父 AWS 账户并非存储桶拥有者的 IAM 用户请求的存储桶操作

在此示例中，请求是由父 AWS 账户是 1111-1111-1111 的 IAM 用户 Jill 发送的，但存储桶由另一个 AWS 账户 2222-2222-2222 拥有。



Jill 将需要来自父 AWS 账户和存储桶拥有者双方的权限。Amazon S3 通过以下方式评估上下文：

1. 由于该请求来自 IAM 用户，Amazon S3 通过检查由该账户授权的策略以验证 Jill 是否拥有所需的权限，从而对用户上下文进行评估。如果 Jill 拥有权限，则 Amazon S3 进一步评估存储桶上下文；否则，Amazon S3 拒绝该请求。
2. 在存储桶上下文中，Amazon S3 验证存储桶拥有者 2222-2222-2222 是否已授予 Jill (或她的父 AWS 账户) 执行所请求的操作的权限。如果她拥有该权限，则 Amazon S3 授权该请求并执行该操作；否则，Amazon S3 拒绝该请求。

## Amazon S3 如何授权对象操作的请求

当 Amazon S3 收到对象操作请求时，它会将基于资源的所有相关权限（对象访问控制列表（ACL）、存储桶策略、存储桶 ACL）和 IAM 用户策略转换为将在运行时进行评估的策略集。然后它会通过一系列步骤评估生成的策略集。在每个步骤中，它会在三个特定上下文（用户上下文、存储桶上下文和对象上下文）中评估一个策略子集。

1. 用户上下文 – 如果请求者是 IAM 用户，则该用户必须拥有来自其所属的父 AWS 账户的权限。在此步骤中，Amazon S3 会评估由父账户（也称为上下文机构）拥有的一组策略子集。该策略子集包含父级附加到该用户的用户策略。如果父级也拥有请求中的资源（存储桶、对象），则 Amazon S3 还会同时评估相应资源策略（存储桶策略、存储桶 ACL 和对象 ACL）。

### Note

如果父 AWS 账户拥有资源（存储桶或对象），则它可通过使用用户策略或资源策略向其 IAM 用户授予资源权限。

2. 存储桶上下文 – 在此上下文中，Amazon S3 评估拥有该存储桶的 AWS 账户所拥有的策略。

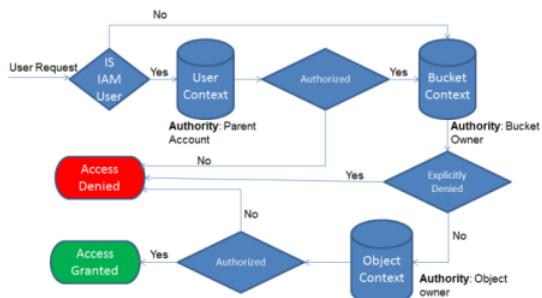
如果拥有请求中的对象的 AWS 账户与存储桶拥有者不同，则 Amazon S3 会在存储桶上下文中检查策略，查看存储桶拥有者是否已显式拒绝对该对象的访问。如果对该对象设置了显式拒绝，则 Amazon S3 不对请求授权。

3. 对象上下文 – 请求者必须拥有来自对象拥有者的权限才能执行特定对象操作。在此步骤中，Amazon S3 将评估对象 ACL。

### Note

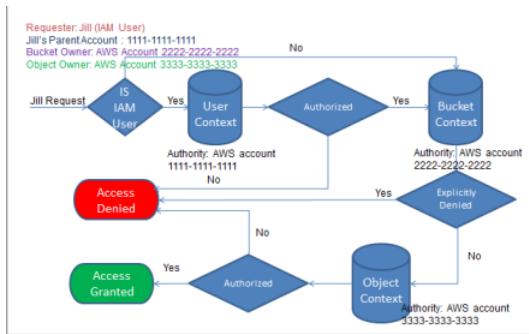
如果存储桶和对象拥有者相同，则可以在存储桶策略中授予对该对象的访问权限，并会在存储桶上下文中对此进行评估。如果拥有者不同，则对象拥有者必须使用对象 ACL 授予权限。如果拥有该对象的 AWS 账户也是 IAM 用户所属的父账户，则该 AWS 账户可以在用户策略中配置对象权限，并会在用户上下文中对此进行评估。有关使用这些备选访问策略的更多信息，请参阅[有关使用可用访问策略选项的准则 \(p. 252\)](#)。

下面是基于上下文来评估对象操作的图解说明。



## 示例 1：对象操作请求

在此示例中，IAM 用户 Jill (父 AWS 账户是 1111-1111-1111) 针对 AWS 账户 3333-3333-3333 拥有的对象发送对象操作请求 (例如，Get object)，该对象位于由 AWS 账户 2222-2222-2222 拥有的存储桶中。



Jill 需要从父 AWS 账户、存储桶拥有者和对象拥有者获得权限。Amazon S3 通过以下方式评估上下文：

- 由于该请求来自 IAM 用户，Amazon S3 将评估用户上下文以验证父 AWS 账户 1111-1111-1111 已授予 Jill 执行所请求的操作的权限。如果她拥有该权限，则 Amazon S3 评估存储桶上下文。否则，Amazon S3 拒绝该请求。
- 在存储桶上下文中，存储桶拥有者 (AWS 账户 2222-2222-2222) 是上下文机构。Amazon S3 对存储桶策略进行评估以确定存储桶拥有者是否显式拒绝了 Jill 对该对象的访问。
- 在对象上下文中，上下文机构是 AWS 账户 3333-3333-3333，即对象拥有者。Amazon S3 评估对象 ACL 以确定 Jill 是否拥有访问该对象的权限。如果她拥有该权限，则 Amazon S3 对该请求进行授权。

## 有关使用可用访问策略选项的准则

Amazon S3 支持使用基于资源的策略和用户策略来管理对 Amazon S3 资源的访问 (请参阅[管理对资源的访问 \(访问策略选项\) \(p. 244\)](#))。基于资源的策略包括存储桶策略、存储桶 ACL 和对象 ACL。本节介绍使用基于资源的访问策略管理对您的 Amazon S3 资源的访问的特定场景。

### 何时使用基于 ACL 的访问策略 (存储桶和对象 ACL)

存储桶和对象都有可用于授予权限的关联 ACL。以下各节描述使用对象 ACL 和存储桶 ACL 的场景。

#### 何时使用对象 ACL

除对象 ACL 外，对象拥有者还有其他方法可以管理对象权限。例如：

- 如果拥有对象的 AWS 账户还拥有存储桶，则可以编写存储桶策略来管理对象权限。
- 如果拥有对象的 AWS 账户要向其账户内的一个用户授予权限，则可以使用用户策略。

那么，何时使用对象 ACL 管理对象权限？下面是使用对象 ACL 管理对象权限的一些场景。

- 若存储桶拥有者不拥有对象，则只能通过对象 ACL 管理对象访问权限 – 拥有存储桶的 AWS 账户可以授予其他 AWS 账户上传对象的权限。存储桶拥有者不拥有这些对象。创建对象的 AWS 账户使用对象 ACL 授予权限。

#### Note

存储桶拥有者不能对其不拥有的对象授予权限。例如，授予对象权限的存储桶策略仅适用于存储桶拥有者拥有的对象。但是，无论存储桶中的对象归谁所有，支付账单的存储桶拥有者都可以编写存储桶策略来拒绝对存储桶中任何对象的访问。存储桶拥有者还可以删除存储桶内的任何对象。

- 权限因对象而异，您需要在对象级别管理权限 – 您可以只编写一条策略语句，向一个 AWS 账户授予对数百万具有特定键名称前缀的对象的读取权限。例如，授予对以键名称前缀“logs”开头的对象的读取权限。但是，如果您的访问权限因对象而异，那么使用存储桶策略授予对各个对象的权限可能不太实际。此外，存储桶策略还有 20 KB 的大小限制。

在这种情况下，使用对象 ACL 可能是比较合适的选择。但是，对象 ACL 也有最多 100 个授权的限制（请参阅[访问控制列表 \(ACL\) 概述 \(p. 333\)](#)）。

- 对象 ACL 仅控制对象级权限 – 整个存储桶只有一个存储桶策略，但对象 ACL 是按对象指定的。

拥有存储桶的 AWS 账户可以授予其他 AWS 账户权限来管理访问策略。它允许该账户更改该策略中的任何内容。为更好地管理权限，您可以选择不授予如此广泛的权限，而只授予对对象子集的 READ-ACP 和 WRITE-ACP 权限。这样可以限制该账户，使其只能通过更新各个对象 ACL 来管理对特定对象的权限。

## 何时使用存储桶 ACL

存储桶 ACL 的唯一建议的使用案例是授予 Amazon S3 日志传输组写入权限，以便将访问日志对象写入您的存储桶（请参阅[Amazon S3 服务器访问日志记录 \(p. 506\)](#)）。如果希望 Amazon S3 将访问日志传输到您的存储桶，您需要向日志传输组授予对存储桶的写入权限。向日志传输组授予必要权限的唯一方法是通过存储桶 ACL，如下面的存储桶 ACL 片段所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    ...
  </Owner>
  <AccessControlList>
    <Grant>
      ...
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

## 何时使用存储桶策略

如果拥有存储桶的 AWS 账户要向其账户中的用户授予权限，则使用存储桶策略或用户策略均可。但在以下情况下则需要使用存储桶策略。

- 要管理所有 Amazon S3 权限的跨账户权限 – 您可以使用 ACL 授予其他账户跨账户权限，但 ACL 仅支持一组有限的权限（[我能授予哪些许可？\(p. 335\)](#)），其中并不包括所有 Amazon S3 权限。例如，使用 ACL 不能授予对存储桶子资源（请参阅[管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)）的权限。

虽然存储桶和用户策略都支持授予所有 Amazon S3 操作权限 (请参阅[在策略中指定权限 \(p. 282\)](#))，但用户策略是用来管理您账户中用户的权限的。对于其他 AWS 账户或其他账户中用户的跨账户权限，则必须使用存储桶策略。

## 何时使用用户策略

一般情况下，您都可以使用用户策略或存储桶策略来管理权限。您可以选择通过创建用户并向用户 (或用户组) 附加策略来分别管理权限，或者也可能发现基于资源的策略 (如存储桶策略) 更适合您的场景。

请注意，使用 AWS Identity and Access Management (IAM) 可以在您的 AWS 账户中创建多个用户并通过用户策略管理其权限。IAM 用户必须拥有两种权限：一种权限来自其父账户，另一种权限来自拥有该用户要访问的资源的 AWS 账户。权限可通过以下方式授予：

- 来自父账户的权限 – 父账户可以通过附加用户策略向其用户授予权限。
- 来自资源拥有者的权限 – 资源拥有者可以向 IAM 用户 (使用存储桶策略) 或父账户 (使用存储桶策略、存储桶 ACL 或对象 ACL) 授予权限。

这类似于一个孩子想要玩别人的玩具。在这种情况下，这个孩子必须征得家长和玩具主人的同意才能玩这个玩具。

## 权限委托

如果某 AWS 账户拥有一项资源，它可以将这些权限授予其他 AWS 账户。然后这个账户就可以将这些权限或其子集委托给该账户中的用户。这称为权限委托。但从另一账户接收权限的账户不能向其他 AWS 账户跨账户委托权限。

## 相关主题

我们建议您先阅读讲解如何管理对您的 Amazon S3 资源的访问的介绍性主题和相关指南。有关更多信息，请参阅[Amazon S3 资源访问权限管理介绍 \(p. 242\)](#)。然后再通过以下主题了解有关特定访问策略选项的更多信息。

- [使用存储桶策略和用户策略。 \(p. 279\)](#)
- [使用 ACL 管理访问 \(p. 333\)](#)

## 示例演练：管理对 Amazon S3 资源的访问

本主题提供了以下介绍性示例演练，演示如何授予针对 Amazon S3 资源的访问权限。这些示例使用 AWS 管理控制台来创建资源（存储桶、对象、用户）并授予它们相应的权限。然后这些示例将向您演示如何使用命令行工具来验证这些权限而不必编写任何代码。我们提供的命令既包括 AWS 命令行界面（CLI），也包括适用于 Windows PowerShell 的 AWS 工具。

- [示例 1：存储桶拥有者向其用户授予存储桶权限 \(p. 258\)](#)

您在您的账户中创建的 IAM 用户默认情况下没有权限。在本练习中，您要授予用户权限来执行存储桶和对象操作。

- [示例 2：存储桶拥有者授予跨账户存储桶权限 \(p. 262\)](#)

在本练习中，存储桶拥有者账户 A 对另一个 AWS 账户（账户 B）授予跨账户权限，然后账户 B 将这些权限委派给其账户中的用户。

- 在对象与存储桶拥有者不同的情况下管理对象权限

在本例中的示例方案是一个存储桶拥有者向其他人授予对象权限，但并不是该存储桶中所有对象都归该存储桶拥有者所有。存储桶拥有者需要什么权限，以及如何能委派这些权限？

创建存储桶的 AWS 账户称为存储桶拥有者。该拥有者可以向其他 AWS 账户授予上传对象的权限，而创建对象的 AWS 账户拥有该对象。该存储桶拥有者对其他 AWS 账户创建的对象没有任何权限。如果该存储桶拥有者编写了一个存储桶策略来授予针对对象的访问权限，则该策略不适用于其他账户拥有的对象。

在这种情况下，对象拥有者必须首先使用对象 ACL 向存储桶拥有者授予权限。然后存储桶拥有者才能够如下所示，将这些对象权限委派给其他人，其自己账户中的用户或另一个 AWS 账户。

- [示例 3：存储桶拥有者向其用户授予不属于自己的对象的权限 \(p. 267\)](#)

在本练习中，存储桶拥有者首先从对象拥有者获取权限。然后存储桶拥有者将这些权限委派给自己的账户中的用户。

- [示例 4：存储桶拥有者针对不属于自己的对象授予跨账户权限 \(p. 271\)](#)

在从对象拥有者获得权限后，存储桶拥有者无法将权限委派给其他 AWS 账户，因为不支持跨账户委派（请参阅[权限委托 \(p. 254\)](#)）。但是，存储桶拥有者可以创建具有执行特定操作（如获取对象）的权限的 IAM 角色，并允许其他 AWS 账户担任该角色。这样，任何担任该角色的人都能够访问对象。此示例显示存储桶拥有者如何使用 IAM 角色来启用跨账户委派。

## 在尝试示例演练之前

这些示例使用 AWS 管理控制台来创建资源和授予权限。为了测试权限，这些示例使用命令行工具 AWS Command Line Interface (CLI) 和适用于 Windows PowerShell 的 AWS 工具，因此您无需编写任何代码。要测试权限，您需要设置其中的一个工具。有关更多信息，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

此外，在创建资源时，这些示例并未使用 AWS 账户的根凭证。而是在这些账户中创建一个管理员用户来执行这些任务。

### 关于使用管理员用户来创建资源和授予权限

AWS Identity and Access Management (IAM) 建议不要使用 AWS 账户的根凭证发起请求。而是应创建 IAM 用户，向该用户授予完整访问权，然后使用该用户的凭证来与 AWS 交互。我们将此用户称为管理员用户。有关更多信息，请转到 AWS General Reference 中的[根账户凭证与 IAM 用户凭证](#)和 IAM 用户指南中的[IAM 最佳实践](#)。

本部分中的所有示例演练都使用管理员用户凭证。如果您还未创建您的 AWS 账户的管理员用户，此处的主题会向您演示这一过程。

请注意，要使用用户凭证登录 AWS 管理控制台，需要使用 IAM 用户登录 URL。IAM 控制台会为您的 AWS 账户提供此 URL。这些主题向您演示如何获取该 URL。

## 设置用于示例演练的工具

介绍性示例（请参阅[示例演练：管理对 Amazon S3 资源的访问 \(p. 255\)](#)）使用 AWS 管理控制台来创建资源和授予权限。为了测试权限，这些示例使用命令行工具 AWS Command Line Interface (CLI) 和适用于 Windows PowerShell 的 AWS 工具，因此您无需编写任何代码。要测试权限，您必须设置其中的一个工具。

### 设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅 AWS Command Line Interface 用户指南 中的以下主题。

[使用 AWS 命令行界面进行设置](#)

[安装 AWS 命令行接口](#)

[配置 AWS 命令行界面](#)

2. 设置默认配置。

您将在 AWS CLI 设置文件中存储用户证书。使用 AWS 账户证书在配置文件中创建默认配置。有关查找并编辑 AWS CLI 配置文件的说明，请参阅[配置和凭证文件](#)。

```
[default]
aws_access_key_id = access key ID
aws_secret_access_key = secret access key
region = us-west-2
```

3. 在命令提示符处输入以下命令来验证设置。这两个命令没有显式提供证书，因此将使用默认配置的证书。

- 尝试 help 命令

```
aws help
```

- 使用 aws s3 ls 获取所配置账户的存储桶列表。

```
aws s3 ls
```

当您进行示例演练时，您将创建用户，并且通过创建配置将用户凭证保存在配置文件中，如以下示例所示。请注意，这些配置都有名称（AccountAdmin 和 AccountBadmin）：

```
[profile AccountAdmin]
aws_access_key_id = User AccountAdmin access key ID
aws_secret_access_key = User AccountAdmin secret access key
region = us-west-2

[profile AccountBadmin]
aws_access_key_id = Account B access key ID
aws_secret_access_key = Account B secret access key
region = us-east-1
```

要使用这些用户凭证来执行命令，请添加参数 --profile 来指定配置名称。以下 AWS CLI 命令检索 examplebucket 中对象的列表，并指定 AccountBadmin 配置。

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

或者，您也可以从命令提示符处更改 AWS\_DEFAULT\_PROFILE 环境变量，将一组用户凭证配置为默认配置文件。完成此操作后，每当不带 --profile 参数执行 AWS CLI 命令的时候，AWS CLI 就会将您在环境变量中设置的配置文件用作默认配置文件。

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

### 设置适用于 Windows PowerShell 的 AWS 工具

1. 下载并配置适用于 Windows PowerShell 的 AWS 工具。有关说明，请参阅适用于 Windows PowerShell 的 AWS 工具 用户指南 中的[下载和安装适用于 Windows PowerShell 的 AWS 工具](#)。

#### Note

为加载适用于 Windows PowerShell 模块的 AWS 工具，您需要启用 PowerShell 脚本执行。有关更多信息，请参阅适用于 Windows PowerShell 的 AWS 工具 用户指南 中的[启用脚本执行](#)。

2. 对于这些练习，您需使用 Set-AWSCredentials 命令指定每个会话的 AWS 证书。该命令会将证书保存到持久存储 (-StoreAs 参数)。

```
Set-AWSCredentials -AccessKey AccessKeyId -SecretKey SecretAccessKey -storeas string
```

3. 验证设置。

- 执行 Get-Command 以检索可用于执行 Amazon S3 操作的命令的列表。

```
Get-Command -module awspowershell -noun s3* -StoredCredentials string
```

- 执行 Get-S3Object 命令以检索存储桶中对象的列表。

```
Get-S3Object -BucketName bucketname -StoredCredentials string
```

有关命令的列表，请参阅 [Amazon Simple Storage Service Cmdlet](#)。

您现在可以着手开始练习。访问本部分开头提供的链接。

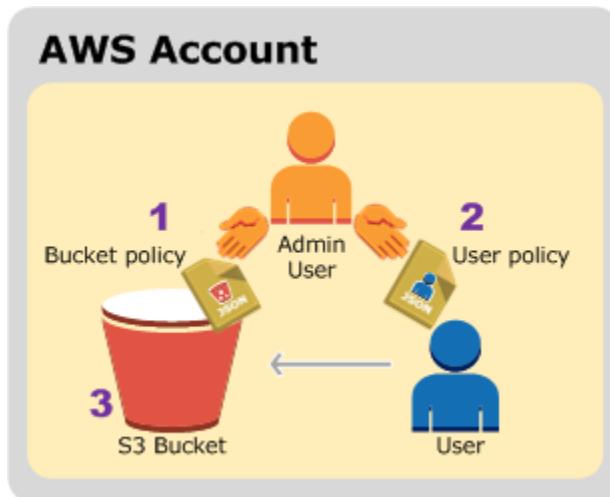
## 示例 1：存储桶拥有者向其用户授予存储桶权限

### 主题

- 步骤 0：准备演练 (p. 258)
- 步骤 1：在账户 A 中创建资源（一个存储桶和一个 IAM 用户）并授予权限 (p. 259)
- 步骤 2：测试权限 (p. 261)

在此练习中，一个 AWS 账户拥有一个存储桶，该账户中有一个 IAM 用户。默认情况下，该用户没有任何权限。父账户必须向该用户授予权限才能执行任何任务。存储桶拥有者和用户所属的父账户相同。因此，AWS 账户可以使用存储桶策略和/或用户策略向其用户授予存储桶权限。您将使用存储桶策略授予一些权限，使用用户策略授予其他权限。

以下是演练的步骤概括：



1. 账户管理员创建存储桶策略，向用户授予一组权限。
2. 账户管理员将用户策略附加到用户，授予其他权限。
3. 然后，用户尝试使用通过存储桶策略和用户策略授予的权限。

对于此示例，您需要一个 AWS 账户。您将创建一个管理员用户（请参阅[关于使用管理员用户来创建资源和授予权限 \(p. 255\)](#)），而不是使用账户的根凭证。我们按如下所示引用 AWS 账户和管理员用户：

账户 ID	账户名称	账户中的管理员用户
1111-1111-1111	账户 A	AccountAdmin

创建用户和授予权限的所有任务都在 AWS 管理控制台中完成。为验证权限，演练中使用命令行工具（AWS 命令行界面 (CLI) 和适用于 Windows PowerShell 的 AWS 工具）验证权限，因此您无需编写任何代码。

### 步骤 0：准备演练

1. 确保您有一个 AWS 账户并且它的一个用户拥有管理员权限。
  - a. 根据需要注册账户。我们将此账户称为账户 A。
    - i. 转到 <https://aws.amazon.com/s3> 并单击 Sign Up。

- ii. 按照屏幕上的说明进行操作。

账户激活可用时，AWS 会通过电子邮件向您发送通知。

- b. 在账户 A 中，创建一个管理员用户 AccountAadmin。使用账户 A 证书登录 IAM 控制台，然后执行以下操作：

- i. 创建用户 AccountAadmin 并记下用户安全证书。

有关说明，请参阅 IAM 用户指南 中的[在您的 AWS 账户中创建 IAM 用户](#)。

- ii. 通过附加一个授予完全访问权限的用户策略来向 AccountAadmin 授予管理员权限。

有关说明，请参阅 IAM 用户指南 中的[使用策略](#)。

- iii. 记下 AccountAadmin 的 IAM User Sign-In URL (IAM 用户登录 URL)。您在登录 AWS 管理控制台时需要使用此 URL。有关在何处查找它的更多信息，请参阅 IAM 用户指南 中的[IAM 用户如何登录您的账户](#)。记下每个账户的 URL。

2. 设置 AWS 命令行界面 (CLI) 或适用于 Windows PowerShell 的 AWS 工具。请务必保存管理员用户证书，如下所示：

- 如果使用 AWS CLI，请在配置文件中创建配置 AccountAadmin。
- 如果使用适用于 Windows PowerShell 的 AWS 工具，请确保将用于会话的证书存储为 AccountAadmin。

有关说明，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

## 步骤 1：在账户 A 中创建资源（一个存储桶和一个 IAM 用户）并授予权限

使用账户 A 中 AccountAadmin 用户的证书和专用的 IAM 用户登录 URL 登录 AWS 管理控制台，然后执行以下操作：

1. 创建资源（一个存储桶和一个 IAM 用户）

- a. 在 Amazon S3 控制台中创建一个存储桶。记下您将它创建在哪个 AWS 区域。有关说明，请参阅[如何创建 S3 存储桶？](#)（在 Amazon Simple Storage Service 控制台用户指南 中）。
- b. 在 IAM 控制台中，执行以下操作：

- i. 创建用户 Dave。

有关说明，请参阅 IAM 用户指南 中的[创建 IAM 用户 \(AWS 管理控制台\)](#)。

- ii. 记下 UserDave 证书。

- iii. 记下用户 Dave 的 Amazon 资源名称 (ARN)。在 IAM 控制台中，选择该用户，Summary (摘要) 选项卡会提供用户 ARN。

2. 授予权限。

因为存储桶拥有者和用户所属的父账户相同，所以 AWS 账户可以使用存储桶策略和/或用户策略向用户授予权限。在此示例中，您同时使用这两种方法。如果对象也由同一个账户拥有，则存储桶拥有者可以在存储桶策略（或 IAM 策略）中授予对象权限。

- a. 在 Amazon S3 控制台中，将以下存储桶策略附加到 `examplebucket`。

该策略包含两个语句。

- 第一个语句向 Dave 授予存储桶操作权限 `s3:GetBucketLocation` 和 `s3>ListBucket`。
- 第二个语句授予 `s3:GetObject` 权限。因为账户 A 还拥有对象，所以账户管理员能够授予 `s3:GetObject` 权限。

在 Principal 语句中，Dave 通过其用户 ARN 进行标识。有关策略元素的更多信息，请参阅[访问策略语言概述 \(p. 279\)](#)。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "statement1",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"  
            },  
            "Action": [  
                "s3:GetBucketLocation",  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket"  
            ]  
        },  
        {  
            "Sid": "statement2",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"  
            },  
            "Action": [  
                "s3:GetObject"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket/*"  
            ]  
        }  
    ]  
}
```

- b. 使用以下策略为用户 Dave 创建一个内联策略。该策略向 Dave 授予 s3:PutObject 权限。您需要通过提供存储桶名称来更新策略。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "PermissionForObjectOperations",  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket/*"  
            ]  
        }  
    ]  
}
```

有关说明，请参阅 IAM 用户指南 中的[使用内联策略](#)。请注意，您需要使用账户 A 证书登录控制台。

## 步骤 2：测试权限

使用 Dave 的凭证验证权限是否发挥作用。您可以使用以下两个过程之一。

使用 AWS CLI 进行测试

1. 通过添加以下 UserDaveAccountA 配置文件更新 AWS CLI 配置文件。有关更多信息，请参阅 [设置用于示例演练的工具 \(p. 256\)](#)。

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. 验证 Dave 是否可以执行在用户策略中授权的操作。使用以下 AWS CLI put-object 命令上传示例对象。

该命令中的 --body 参数指示要上传的源文件。例如，如果文件处于 Windows 计算机上 C: 驱动器的根目录中，则指定 c:\HappyFace.jpg。--key 参数提供对象的键名称。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body HappyFace.jpg --profile UserDaveAccountA
```

执行以下 AWS CLI 命令以获取对象。

```
aws s3api get-object --bucket examplebucket --key HappyFace.jpg OutputFile.jpg --profile UserDaveAccountA
```

使用适用于 Windows PowerShell 的 AWS 工具进行测试

1. 将 Dave 的证书存储为 AccountADave。您随后使用这些证书对对象执行 PUT 和 GET 操作。

```
Set-AWSCredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -StoreAs AccountADave
```

2. 通过用户 Dave 的已存储证书，使用适用于 Windows PowerShell 的 AWS 工具 Write-S3Object 命令上传示例对象。

```
Write-S3Object -BucketName examplebucket -Key HappyFace.jpg -File HappyFace.jpg -StoredCredentials AccountADave
```

下载之前上传的对象。

```
Read-S3Object -BucketName examplebucket -Key HappyFace.jpg -File Output.jpg -StoredCredentials AccountADave
```

## 示例 2：存储桶拥有者授予跨账户存储桶权限

### 主题

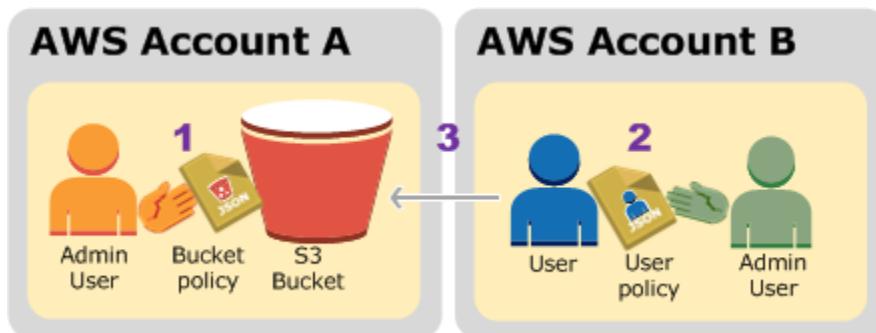
- 步骤 0：准备演练 (p. 263)
- 步骤 1：执行账户 A 任务 (p. 264)
- 步骤 2：执行账户 B 任务 (p. 265)
- 步骤 3：额外练习 - 尝试显式拒绝 (p. 266)
- 步骤 4：清除 (p. 267)

一个 AWS 账户（例如账户 A）可以将访问自身资源（如存储桶和对象）的权限授予另一个 AWS 账户（账户 B）。账户 B 随后可以将这些权限委托给其账户中的用户。在此示例情景中，存储桶拥有者向另一个账户授予执行特定存储桶操作的跨账户权限。

### Note

账户 A 还可以使用存储桶策略直接向账户 B 中的用户授予权限。但是该用户仍需要来自用户所属的父账户（账户 B）的权限，即使账户 B 没有来自账户 A 的权限也是如此。该用户只要同时拥有来自资源拥有者和父账户的权限，便能够访问资源。

以下概括了演练步骤：



1. 账户 A 管理员用户附加一个存储桶策略，该策略向账户 B 授予执行特定存储桶操作的跨账户权限。

请注意，账户 B 中的管理员用户将自动继承这些权限。

2. 账户 B 管理员用户将用户策略附加到用户，委托从账户 A 收到的权限。
3. 账户 B 中的用户随后通过访问账户 A 拥有的存储桶中的对象来验证权限。

对于此示例，您需要两个账户。下表显示我们如何引用这些账户和其中的管理员用户。根据 IAM 准则（请参阅[关于使用管理员用户来创建资源和授予权限 \(p. 255\)](#)），我们在本次演练中并不使用账户根凭证。而是在每个账户中创建一个管理员用户，在创建资源和向他们授予权限时使用这些凭证。

AWS 账户 ID	账户名称	账户中的管理员用户
1111-1111-1111	账户 A	AccountAadmin
2222-2222-2222	账户 B	AccountBadmin

创建用户和授予权限的所有任务都在 AWS 管理控制台中完成。为验证权限，演练使用命令行工具（AWS 命令行界面（CLI）和适用于 Windows PowerShell 的 AWS 工具），因此您无需编写任何代码。

## 步骤 0：准备演练

1. 确保您有两个 AWS 账户并且每个账户都有一个管理员用户，如前面部分的表中所示。
  - a. 根据需要注册 AWS 账户。
    - i. 转到 <https://aws.amazon.com/s3/>，然后单击 Create an AWS Account。
    - ii. 按照屏幕上的说明进行操作。

账户激活可用时，AWS 会通过电子邮件向您发送通知。
  - b. 使用账户 A 证书登录 [IAM 控制台](#)以创建管理员用户：
    - i. 创建用户 AccountAdmin 并记下安全证书。有关说明，请参阅 IAM 用户指南 中的[在您的 AWS 账户中创建 IAM 用户](#)。
    - ii. 通过附加一个授予完全访问权限的用户策略来向 AccountAdmin 授予管理员权限。有关说明，请参阅 IAM 用户指南 中的[使用策略](#)。
  - c. 在 IAM 控制台中，记下 Dashboard (控制面板) 上的 IAM User Sign-In URL (IAM 用户登录 URL)。账户中的所有用户都必须使用此 URL 登录 AWS 管理控制台。

有关更多信息，请参阅 IAM 用户指南 中的[用户如何登录您的账户](#)。

  - d. 使用账户 B 证书重复上一个步骤，然后创建管理员用户 AccountBadmin。
2. 设置 AWS 命令行界面 (CLI) 或适用于 Windows PowerShell 的 AWS 工具。确保按如下方式保存管理员用户证书：
  - 如果使用 AWS CLI，请在配置文件中创建两个配置 AccountAdmin 和 AccountBadmin。
  - 如果使用适用于 Windows PowerShell 的 AWS 工具，请确保您将用于会话的证书存储为 AccountAdmin 和 AccountBadmin。

有关说明，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

3. 保存管理员用户证书，也称为配置文件。您可以对输入的每个命令使用配置文件名称而不是指定证书。有关更多信息，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

- a. 在 AWS CLI 配置文件中为两个账户中的每个管理员用户添加配置文件。

```
[profile AccountAdmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1

[profile AccountBadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1
```

- b. 如果使用适用于 Windows PowerShell 的 AWS 工具

```
set-awscredentials -AccessKey AcctA-access-key-ID -SecretKey AcctA-secret-access-
key -storeas AccountAdmin
set-awscredentials -AccessKey AcctB-access-key-ID -SecretKey AcctB-secret-access-
key -storeas AccountBadmin
```

## 步骤 1：执行账户 A 任务

### 步骤 1.1：登录 AWS 管理控制台

首先使用账户 A 的 IAM 用户登录 URL，以 AccountAdmin 用户身份登录 AWS 管理控制台。此用户将创建一个存储桶并向其附加一个策略。

### 步骤 1.2：创建存储桶

- 在 Amazon S3 控制台中创建一个存储桶。此练习假设该存储桶在美国东部（弗吉尼亚北部）区域创建，命名为 examplebucket。

有关说明，请参阅[如何创建 S3 存储桶？](#)（在 Amazon Simple Storage Service 控制台用户指南 中）。

- 向存储桶上传示例对象。

有关说明，请参阅 Amazon Simple Storage Service 入门指南 中的[向存储桶添加对象](#)。

### 步骤 1.3：附加一个存储桶策略，向账户 B 授予跨账户权限

该存储桶策略向账户 B 授予 s3:GetBucketLocation 和 s3>ListBucket 权限。假设您仍使用 AccountAdmin 用户证书登录控制台。

- 将以下存储桶策略附加到 examplebucket。该策略向账户 B 授予 s3:GetBucketLocation 和 s3>ListBucket 操作的权限。

有关说明，请参阅[如何添加 S3 存储桶策略？](#)（在 Amazon Simple Storage Service 控制台用户指南 中）。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Example permissions",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountB-ID:root"  
            },  
            "Action": [  
                "s3:GetBucketLocation",  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket"  
            ]  
        }  
    ]  
}
```

- 验证账户 B（及其管理员用户）是否可以执行这些操作。

- 使用 AWS CLI

```
aws s3 ls s3://examplebucket --profile AccountBadmin  
aws s3api get-bucket-location --bucket examplebucket --profile AccountBadmin
```

- 使用适用于 Windows PowerShell 的 AWS 工具

```
get-s3object -BucketName example2bucket -StoredCredentials AccountBadmin  
get-s3bucketlocation -BucketName example2bucket -StoredCredentials AccountBadmin
```

## 步骤 2：执行账户 B 任务

现在账户 B 管理员创建用户 Dave，向 Dave 委托从账户 A 收到的权限。

### 步骤 2.1：登录 AWS 管理控制台

首先使用账户 B 的 IAM 用户登录 URL，以 AccountBAdmin 用户身份登录 AWS 管理控制台。

### 步骤 2.2：在账户 B 中创建用户 Dave

1. 在 IAM 控制台中，创建用户 Dave。

有关说明，请参阅 IAM 用户指南 中的[创建 IAM 用户 \(AWS 管理控制台\)](#)。

2. 记下 UserDave 证书。

### 步骤 2.3：向用户 Dave 委托权限

- 使用以下策略为用户 Dave 创建一个内联策略。您需要通过提供存储桶名称来更新策略。

假设您使用 AccountBAdmin 用户证书登录控制台。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Example",  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket"  
            ]  
        }  
    ]  
}
```

有关说明，请参阅 IAM 用户指南 中的[使用内联策略](#)。

### 步骤 2.4：测试权限

现在，账户 B 中的 Dave 可以列出账户 A 拥有的 examplebucket 的内容。您可以使用以下过程之一验证权限。

#### 使用 AWS CLI 进行测试

1. 将 UserDave 配置文件添加到 AWS CLI 配置文件。有关配置文件的更多信息，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

```
[profile UserDave]  
aws_access_key_id = access-key  
aws_secret_access_key = secret-access-key  
region = us-east-1
```

2. 在命令提示符处，输入以下 AWS CLI 命令，验证 Dave 现在是否可以从账户 A 拥有的 examplebucket 获取对象列表。请注意，该命令指定了 UserDave 配置文件。

```
aws s3 ls s3://examplebucket --profile UserDave
```

Dave 没有任何其他权限。因此，如果他尝试任何其他操作（例如，以下获取存储桶位置的操作），则 Amazon S3 会返回权限被拒绝的信息。

```
aws s3api get-bucket-location --bucket examplebucket --profile UserDave
```

使用适用于 Windows PowerShell 的 AWS 工具进行测试

1. 将 Dave 的证书存储为 AccountBDave。

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas AccountBDave
```

2. 尝试列出存储桶的命令。

```
get-s3object -BucketName example2bucket -StoredCredentials AccountBDave
```

Dave 没有任何其他权限。因此，如果他尝试任何其他操作（例如，以下获取存储桶位置的操作），则 Amazon S3 会返回权限被拒绝的信息。

```
get-s3bucketlocation -BucketName example2bucket -StoredCredentials AccountBDave
```

### 步骤 3：额外练习 - 尝试显式拒绝

您可以通过 ACL、存储桶策略和用户策略授予权限。但是，如果通过存储桶策略或用户策略设置了显式拒绝，则显式拒绝优先于任何其他权限。为进行测试，我们更新存储桶策略，对账户 B 显式拒绝 s3>ListBucket 权限。该策略同时还授予了 s3>ListBucket 权限，但是显式拒绝处于优先地位，因此账户 B 或账户 B 中的用户无法列出 examplebucket 中的对象。

1. 使用账户 A 中的用户 AccountAadmin 的凭证，将存储桶策略替换为以下内容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Example permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:root"
            },
            "Action": [
                "s3:GetBucketLocation",
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket"
            ]
        },
        {
            "Sid": "Deny permission",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:root"
            },
            "Action": [
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket"
            ]
        }
    ]
}
```

```
        "Resource": [
            "arn:aws:s3:::examplebucket"
        ]
    }
}
```

2. 现在，如果您尝试使用 AccountAdmin 凭证获取存储桶列表，则您的访问会被拒绝。

- 使用 AWS CLI：

```
aws s3 ls s3://examplebucket --profile AccountAdmin
```

- 使用适用于 Windows PowerShell 的 AWS 工具：

```
get-s3object -BucketName example2bucket -StoredCredentials AccountBDave
```

## 步骤 4：清除

1. 完成测试之后，您可以执行以下操作来进行清理。
  - 使用账户 A 证书登录 AWS 管理控制台 ([AWS 管理控制台](#))，执行以下操作：
    - 在 Amazon S3 控制台中，删除附加到 **examplebucket** 的存储桶策略。在存储桶 Properties (属性) 中，删除 Permissions (权限) 部分中的策略。
    - 如果该存储桶是为此练习而创建的，请在 Amazon S3 控制台中删除对象，然后删除存储桶。
    - 在 IAM 控制台中，删除 AccountAadmin 用户。
2. 使用账户 B 证书登录 AWS 管理控制台 ([AWS 管理控制台](#))。在 IAM 控制台中，删除用户 AccountBadmin。

## 示例 3：存储桶拥有者向其用户授予不属于自己的对象的权限

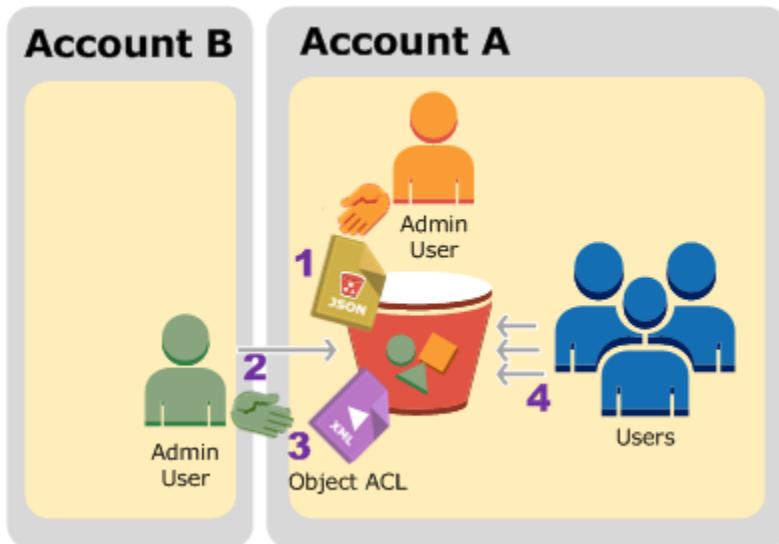
### 主题

- [步骤 0：准备演练 \(p. 268\)](#)
- [步骤 1：执行账户 A 任务 \(p. 269\)](#)
- [步骤 2：执行账户 B 任务 \(p. 270\)](#)
- [步骤 3：测试权限 \(p. 270\)](#)
- [步骤 4：清除 \(p. 271\)](#)

此示例的情况是，存储桶拥有者要授予对象访问权限，但并不是该存储桶中所有对象都归该存储桶拥有者所有。存储桶拥有者如何能够授予不属于自己的对象的权限？在此示例中，存储桶拥有者想要向其账户中的用户授予权限。

存储桶拥有者可让其他 AWS 账户拥有上传对象的权限。这些对象归创建它们的账户所有。存储桶拥有者并不拥有不是自己创建的对象。因此，存储桶拥有者要能够授予对这些对象的访问权，对象拥有者必须首先使用对象 ACL 对该存储桶拥有者授予权限。然后该存储桶拥有者就可以通过存储桶策略委派这些权限。在此示例中，存储桶拥有者向其账户中的用户委派权限。

以下概括了演练步骤：



1. 账户 A 管理员用户使用两条语句来附加存储桶策略。
  - 向账户 B 授予跨账户上传对象的权限。
  - 允许用户使用自己的账户访问存储桶中的对象。
2. 账户 B 管理员用户将对象上传至账户 A 拥有的存储桶。
3. 账户 B 管理员将更新对象 ACL，在其中添加授权，以向存储桶拥有者授予对于该对象的完全控制权限。
4. 账户 A 中的用户将通过访问存储桶中的对象（而不管谁拥有它们）来验证权限。

对于此示例，您需要两个账户。下表显示我们如何引用这些账户和这些账户中的管理员用户。根据 IAM 准则（请参阅[关于使用管理员用户来创建资源和授予权限 \(p. 255\)](#)），我们在本次演练中并不使用账户根凭证。而是在每个账户中创建一个管理员用户，在创建资源和向他们授予权限时使用这些凭证。

AWS 账户 ID	账户名称	账户中的管理员用户
1111-1111-1111	账户 A	AccountAadmin
2222-2222-2222	账户 B	AccountBadmin

创建用户和授予权限的所有任务都在 AWS 管理控制台中完成。为验证权限，演练使用命令行工具（AWS 命令行界面（CLI）和适用于 Windows PowerShell 的 AWS 工具），因此您无需编写任何代码。

## 步骤 0：准备演练

1. 确保您有两个 AWS 账户并且每个账户都有一个管理员用户，如上一部分中的表中所示。
  - a. 根据需要注册 AWS 账户。
    - i. 转到 <https://aws.amazon.com/s3/>，然后单击 Create an AWS Account。
    - ii. 按照屏幕上的说明进行操作。账户激活可用时，AWS 会通过电子邮件向您发送通知。
  - b. 使用账户 A 证书登录 [IAM 控制台](#)，然后执行以下操作以创建管理员用户：
    - 创建用户 AccountAadmin 并记下安全证书。有关添加用户的更多信息，请参阅 IAM 用户指南 中的[在您的 AWS 账户中创建 IAM 用户](#)。
    - 通过附加一个授予完全访问权限的用户策略来向 AccountAadmin 授予管理员权限。有关说明，请参阅 IAM 用户指南 中的[使用策略](#)。

- 在 IAM 控制台 Dashboard (控制面板) 中，记下 IAM User Sign-In URL (IAM 用户登录 URL)。此账户中的用户必须在登录 AWS 管理控制台时使用此 URL。有关更多信息，请参阅 IAM 用户指南中的[用户如何登录您的账户](#)。
- c. 使用账户 B 证书重复上一个步骤，然后创建管理员用户 AccountBadmin。
2. 设置 AWS 命令行界面 (CLI) 或适用于 Windows PowerShell 的 AWS 工具。确保按如下方式保存管理员用户证书：
- 如果使用 AWS CLI，请在配置文件中创建两个配置 AccountAadmin 和 AccountBadmin。
  - 如果使用适用于 Windows PowerShell 的 AWS 工具，请确保您将用于会话的证书存储为 AccountAadmin 和 AccountBadmin。

有关说明，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

## 步骤 1：执行账户 A 任务

### 步骤 1.1：登录 AWS 管理控制台

首先使用账户 A 的 IAM 用户登录 URL，以 AccountAadmin 用户身份登录 AWS 管理控制台。此用户将创建一个存储桶并向其附加一个策略。

### 步骤 1.2：创建存储桶和一个用户，并添加一条授予用户权限的存储桶策略

1. 在 Amazon S3 控制台中创建一个存储桶。此练习假设该存储桶在美国东部 (弗吉尼亚北部) 区域创建，名称为 examplebucket。

有关说明，请参阅[如何创建 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

2. 在 IAM 控制台中，创建用户 Dave。

有关说明，请参阅 IAM 用户指南 中的[创建 IAM 用户 \(AWS 管理控制台\)](#)。

3. 记下 Dave 证书。

4. 在 Amazon S3 控制台中，请将以下存储桶策略附加到 examplebucket 存储桶。有关说明，请参阅[如何添加 S3 存储桶策略？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。按照这些步骤添加存储桶策略。有关如何查找账户 ID 的信息，请参阅[查找您的 AWS 账户 ID](#)。

该策略将向账户 B 授予 s3:PutObject 和 s3>ListBucket 权限。该策略还对用户 Dave 授予 s3:GetObject 权限。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Statement1",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountB-ID:root"  
            },  
            "Action": [  
                "s3:PutObject"  
            ],  
            "Resource": [  
                "arn:aws:s3:::examplebucket/*"  
            ]  
        },  
        {  
            "Sid": "Statement3",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"  
            }  
        }  
    ]  
}
```

```
        "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
    },
    "Action": [
        "s3:GetObject"
    ],
    "Resource": [
        "arn:aws:s3:::examplebucket/*"
    ]
}
]
```

## 步骤 2：执行账户 B 任务

现在账户 B 有权对账户 A 的存储桶执行操作，账户 B 管理员将执行以下操作：

- 将对象上传到账户 A 的存储桶。
- 在对象 ACL 中添加授权，向账户 A（存储桶拥有者）授予完全控制权限。

### 使用 AWS CLI

1. 使用 put-object AWS CLI 命令上传对象。该命令中的 --body 参数指示要上传的源文件。例如，如果该文件在 Windows 计算机上的 C: 驱动器中，您应指定 c:\HappyFace.jpg。--key 参数提供对象的键名称。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body HappyFace.jpg --profile AccountBAdmin
```

2. 在对象 ACL 中添加一个授权，向存储桶拥有者授予对于对象的完全控制权。有关如何查找规范用户 ID 的信息，请参阅[查找账户的规范用户 ID](#)。

```
aws s3api put-object-acl --bucket examplebucket --key HappyFace.jpg --grant-full-control id="AccountA-CanonicalUserID" --profile AccountBAdmin
```

### 使用适用于 Windows PowerShell 的 AWS 工具

1. 使用适用于 Windows PowerShell 的 AWS 工具 Write-S3Object 上传对象。

```
Write-S3Object -BucketName examplebucket -key HappyFace.jpg -file HappyFace.jpg -StoredCredentials AccountBAdmin
```

2. 在对象 ACL 中添加一个授权，向存储桶拥有者授予对于对象的完全控制权。

```
Set-S3ACL -BucketName examplebucket -Key HappyFace.jpg -CannedACLName "bucket-owner-full-control" -StoredCreden
```

## 步骤 3：测试权限

现在验证账户 A 中的用户 Dave 能够访问账户 B 拥有的对象。

### 使用 AWS CLI

1. 将用户 Dave 的证书添加到 AWS CLI 配置文件并创建新的配置 UserDaveAccountA。有关更多信息，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. 执行 get-object AWS CLI 命令下载 HappyFace.jpg 并将它保存在本地。您可以通过添加参数 --profile 为用户 Dave 提供证书。

```
aws s3api get-object --bucket examplebucket --key HappyFace.jpg Outputfile.jpg --profile UserDaveAccountA
```

### 使用适用于 Windows PowerShell 的 AWS 工具

1. 将用户 Dave 的 AWS 证书以 UserDaveAccountA 保存到持久存储。

```
Set-AWSCredentials -AccessKey UserDave-AccessKey -SecretKey UserDave-SecretAccessKey -storeas UserDaveAccountA
```

2. 执行 Read-S3Object 命令下载 HappyFace.jpg 对象并将它保存在本地。您可以通过添加参数 -StoredCredentials 为用户 Dave 提供证书。

```
Read-S3Object -BucketName examplebucket -Key HappyFace.jpg -file HappyFace.jpg -StoredCredentials UserDaveAccountA
```

## 步骤 4：清除

1. 完成测试之后，您可以执行以下操作来进行清理。

- 使用账户 A 证书登录 AWS 管理控制台 ([AWS 管理控制台](#))，执行以下操作：

- 在 Amazon S3 控制台中，删除附加到 `examplebucket` 的存储桶策略。在存储桶 Properties (属性) 中，删除 Permissions (权限) 部分中的策略。
- 如果该存储桶是为此练习而创建的，请在 Amazon S3 控制台中删除对象，然后删除存储桶。
- 在 IAM 控制台中，删除 AccountAdmin 用户。

2. 使用账户 B 证书登录 AWS 管理控制台 ([AWS 管理控制台](#))。在 IAM 控制台中，删除用户 AccountBadmin。

## 示例 4：存储桶拥有者针对不属于自己的对象授予跨账户权限

### 主题

- [背景：跨账户权限和使用 IAM 角色 \(p. 272\)](#)
- [步骤 0：准备演练 \(p. 273\)](#)
- [步骤 1：执行账户 A 任务 \(p. 274\)](#)
- [步骤 2：执行账户 B 任务 \(p. 276\)](#)
- [步骤 3：执行账户 C 任务 \(p. 276\)](#)
- [步骤 4：清除 \(p. 278\)](#)
- [相关资源 \(p. 278\)](#)

在此示例场景中，您拥有一个存储桶，并且启用了其他 AWS 账户以上传对象。也就是说，您的存储桶中可以有其他 AWS 账户所拥有的对象。

现在，假设作为存储桶拥有者，您需要向另一个账户中的用户授予对象（无论拥有者是谁）的跨账户权限。例如，该用户可以是需要访问对象元数据的账单应用程序。存在两个核心问题：

- 存储桶拥有者对其他 AWS 账户创建的对象不拥有权限。因此，存储桶拥有者若要针对不属于自己的对象授予权限，必须先由对象拥有者，也就是创建对象的 AWS 账户向存储桶拥有者授予权限。然后，存储桶拥有者才能委托这些权限。
- 存储桶拥有者账户可以向自己账户中的用户委托权限（请参阅[示例 3：存储桶拥有者向其用户授予不属于自己的对象的权限 \(p. 267\)](#)），但不能向其他 AWS 账户委托权限，因为跨账户委托不受支持。

在这种情况下，存储桶拥有者可以创建具有对象访问权限的 AWS Identity and Access Management (IAM) 角色，然后向另一个 AWS 账户授予临时担任该角色的权限，使它可以访问存储桶中的对象。

## 背景：跨账户权限和使用 IAM 角色

IAM 角色可实现多种资源委托访问权限方案，跨账户访问是重要方案之一。在此示例中，存储桶拥有者（账户 A）使用一个 IAM 角色临时将跨账户对象访问权限委托给另一个 AWS 账户（账户 C）中的用户。您创建的每个 IAM 角色都附加了两个策略：

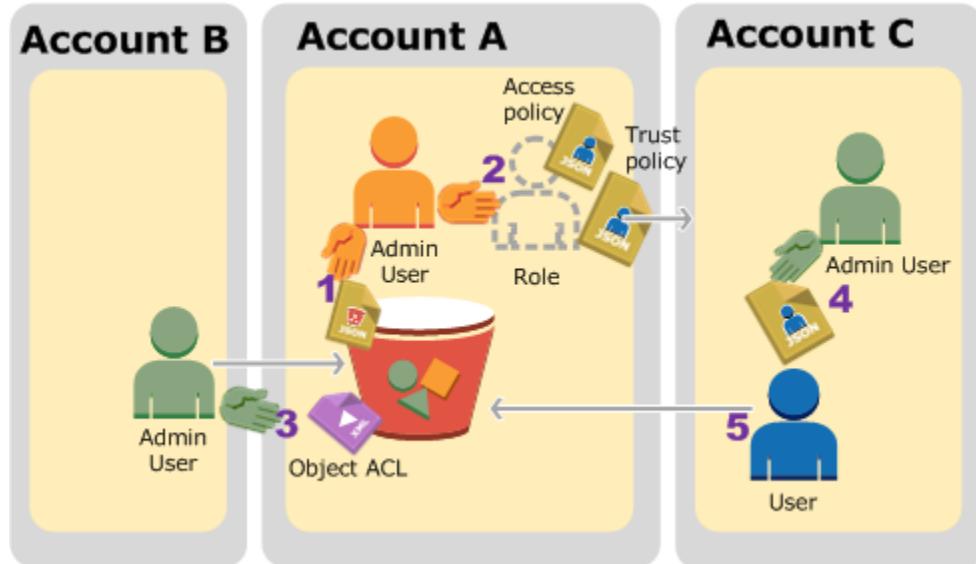
- 一个信任策略，用于标识可以担任角色的其他 AWS 账户。
- 一个访问策略，用于定义在某个用户担任角色时允许的权限，例如 `s3:GetObject`。有关可以在策略中指定的权限的列表，请参阅[在策略中指定权限 \(p. 282\)](#)。

然后，信任策略中标识的 AWS 账户向其用户授予担任角色的权限。该用户即可执行以下操作以访问对象：

- 担任角色，然后在响应中获取临时安全凭证。
- 使用临时安全凭证访问存储桶中的对象。

有关 IAM 角色的更多信息，请转到 IAM 用户指南 中的 [IAM 角色](#)。

下面概括介绍演练步骤：



1. 账户 A 管理员用户附加一个存储桶策略，向账户 B 授予上传对象的条件权限。
2. 账户 A 管理员创建一个 IAM 角色，与账户 C 建立信任，以便该账户中的用户可以访问账户 A。附加到该角色的访问策略限制账户 C 中的用户访问账户 A 时可以执行的操作。
3. 账户 B 管理员将一个对象上传到账户 A 拥有的存储桶，从而向存储桶拥有者授予完全控制权限。

4. 账户 C 管理员创建一个用户，附加一个用户策略允许该用户担任该角色。
5. 账户 C 中的用户首先担任角色，这会向该用户返回临时安全凭证。该用户随后使用这些临时凭证访问存储桶中的对象。

对于此示例，您需要三个账户。下表显示我们如何引用这些账户和这些账户中的管理员用户。根据 IAM 准则（请参阅[关于使用管理员用户来创建资源和授予权限 \(p. 255\)](#)），我们在本次演练中并不使用账户根凭证。而是在每个账户中创建一个管理员用户，在创建资源和向他们授予权限时使用这些凭证

AWS 账户 ID	账户名称	账户中的管理员用户
1111-1111-1111	账户 A	AccountAadmin
2222-2222-2222	账户 B	AccountBadmin
3333-3333-3333	账户 C	AccountCadmin

## 步骤 0：准备演练

### Note

在演练这些步骤时，您可能需要打开文本编辑器，记下某些信息。具体而言，您需要账户 ID、规范用户 ID、供每个账户连接到控制台的 IAM 用户登录 URL，以及 IAM 用户和角色的 Amazon 资源名称 (ARN)。

1. 确保您有三个 AWS 账户并且每个账户都有一个管理员用户，如前一部分表中所示。
  - a. 根据需要注册 AWS 账户。我们将这些账户称为账户 A、账户 B 和账户 C。
    - i. 转到 <https://aws.amazon.com/s3/>，然后单击 Create an AWS Account。
    - ii. 按照屏幕上的说明进行操作。

账户激活可用时，AWS 会通过电子邮件向您发送通知。
  - b. 使用账户 A 证书登录 [IAM 控制台](#)，然后执行以下操作以创建管理员用户：
    - 创建用户 AccountAadmin 并记下安全证书。有关添加用户的更多信息，请参阅 IAM 用户指南 中的[在您的 AWS 账户中创建 IAM 用户](#)。
    - 通过附加一个授予完全访问权限的用户策略来向 AccountAadmin 授予管理员权限。有关说明，请参阅 IAM 用户指南 中的[使用策略](#)。
    - 在 IAM 控制台 Dashboard (控制面板) 中，记下 IAM User Sign-In URL (IAM 用户登录 URL)。此账户中的用户必须在登录 AWS 管理控制台时使用此 URL。有关更多信息，请参阅 IAM 用户指南 中的[用户如何登录您的账户](#)。
  - c. 重复上述步骤，在账户 B 和账户 C 中创建管理员用户。
2. 对于账户 C，记下账户 ID。

在账户 A 中创建 IAM 角色时，信任策略通过指定账户 ID 向账户 C 授予担任该角色的权限。您可以查找账户信息，如下所示：

- a. 转到 <https://aws.amazon.com/>，然后在 My Account/Console 下拉菜单中选择 Security Credentials。
  - b. 使用相应的账户证书登录。
  - c. 单击 Account Identifiers (账户标识符)，然后记下 AWS Account ID (AWS 账户 ID) 和 Canonical User ID (规范用户 ID)。
- 
3. 创建存储桶策略时，您需要以下信息。请记下这些值：  
API 版本 2006-03-01

- 账户 A 的规范用户 ID – 账户 A 管理员向账户 B 管理员授予条件上传对象权限时，条件指定必须获取对象完全控制权的账户 A 用户的规范用户 ID。

**Note**

规范用户 ID 是 Amazon S3 独有的概念。它是 64 字符模糊版本的账户 ID。

- 账户 B 管理员的用户 ARN – 您可以在 IAM 控制台中查找该用户 ARN。您需要选择用户，在 Summary (摘要) 选项卡中查找用户的 ARN。

在存储桶策略中，向 AccountBAdmin 授予上传对象的权限，使用 ARN 指定用户。下面是一个示例 ARN 值：

```
arn:aws:iam::AccountB-ID:user/AccountBAdmin
```

- 设置 AWS 命令行界面 (CLI) 或适用于 Windows PowerShell 的 AWS 工具。请务必保存管理员用户证书，如下所示：

- 如果使用 AWS CLI，请在配置文件中创建配置文件 AccountAAdmin 和 AccountBAdmin。
- 如果使用适用于 Windows PowerShell 的 AWS 工具，请确保您将用于会话的证书存储为 AccountAAdmin 和 AccountBAdmin。

有关说明，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

## 步骤 1：执行账户 A 任务

在此示例中，账户 A 是存储桶拥有者。因此，账户 A 中的用户 AccountAAdmin 将创建一个存储桶，附加向账户 B 管理员授予上传对象权限的存储桶策略，然后创建一个 IAM 角色，向账户 C 授予担任该角色的权限，以便它可以访问存储桶中的对象。

### 步骤 1.1：登录 AWS 管理控制台

首先使用账户 A 的 IAM 用户登录 URL，以 AccountAAdmin 用户身份登录 AWS 管理控制台。此用户将创建一个存储桶并向其附加一个策略。

### 步骤 1.2：创建一个存储桶并附加一个存储桶策略

在 Amazon S3 控制台中，执行以下操作：

- 创建存储桶。此练习假设存储桶名称是 examplebucket。

有关说明，请参阅[如何创建 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

- 附加以下存储桶策略，向账户 B 管理员授予上传对象的条件权限。

您需要通过提供自己的 `examplebucket`、`AccountB-ID` 和 `CanonicalUserId-of-AWSaccountA-BucketOwner` 值来更新策略。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "111",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBAdmin"  
            },  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        },  
    ]  
}
```

```
{  
    "Sid": "112",  
    "Effect": "Deny",  
    "Principal": {  
        "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"  
    },  
    "Action": "s3:PutObject",  
    "Resource": "arn:aws:s3:::examplebucket/*",  
    "Condition": {  
        "StringNotEquals": {  
            "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-AWSaccountA-  
BucketOwner"  
        }  
    }  
}  
]
```

### 步骤 1.3：在账户 A 中创建一个 IAM 角色以允许账户 C 进行跨账户访问

在 IAM 控制台中创建一个 IAM 角色 (“examplerole”)，授予账户 C 担任该角色的权限。请务必仍以账户 A 管理员身份登录，因为该角色必须在账户 A 中创建。

1. 创建角色之前，准备好定义角色所需权限的托管策略。您可在以后的步骤中将此策略附加到角色。
  - a. 在左侧的导航窗格中，单击 Policies (策略)，然后单击 Create Policy (创建策略)。
  - b. 在 Create Your Own Policy (创建您自己的策略) 旁，单击 Select (选择)。
  - c. 在 Policy Name 字段中输入 access-accountA-bucket。
  - d. 复制下面的访问策略，然后将其粘贴到 Policy Document 字段中。该访问策略授予该角色 s3:GetObject 权限，这样，账户 C 用户在担任该角色时只能执行 s3:GetObject 操作。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3:GetObject",  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        }  
    ]  
}
```

- e. 单击 Create Policy (创建策略)。

新策略会显示在托管策略列表中。

2. 在左侧的导航窗格中，单击 Roles (角色)，然后单击 Create New Role (创建新角色)。
3. 输入 exempleroles 作为角色名称，然后单击 Next Step (下一步)。
4. 在 Select Role Type 下，选择 Role for Cross-Account Access，然后单击 Provide access between AWS accounts you own 旁的 Select 按钮。
5. 输入账户 C 的账户 ID。

对于此演练，您不必要求用户经过 Multi-Factor Authentication (MFA) 来承担角色，因此请不要选择该选项。

6. 单击 Next Step (下一步) 设置与该角色关联的权限。
7. 选中您创建的 access-accountA-bucket 策略旁的框，然后单击 Next Step。

“Review (审核)”页面随即出现，使您可以在创建角色之前确认其设置。此页面上有一项极为重要，需要注意，即您可发送给需要使用此角色的用户的链接。单击该链接的用户会直接转到已填写 Account ID

(账户 ID) 和 Role Name (角色名称) 字段的 Switch Role (切换角色) 页面。稍后您可以在任何跨账户角色的 Role Summary (角色摘要) 页面上看到此链接。

8. 检查角色完毕后，单击 创建角色。
- examplerole 角色显示在角色列表中。
9. 单击角色名称 examplerole。
10. 选择 Trust Relationships 选项卡。
11. 单击 Show policy document 并验证所示信任策略是否与以下策略相符。

以下信任策略通过允许账户 C 执行 sts:AssumeRole 操作，与它建立信任。有关更多信息，请参阅 AWS Security Token Service API Reference 中的 [AssumeRole](#)。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountC-ID:root"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

12. 记下您创建的 examplerole 角色的 Amazon 资源名称 (ARN)。

稍后在以下步骤中，您附加一个用户策略以允许 IAM 用户担任此角色，并通过 ARN 值标识角色。

## 步骤 2：执行账户 B 任务

账户 A 拥有的 examplebucket 需要其他账户拥有的对象。在此步骤中，账户 B 管理员使用命令行工具上传对象。

- 使用 put-object AWS CLI 命令将一个对象上传到 examplebucket。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body HappyFace.jpg --grant-full-control id="canonicalUserId-ofTheBucketOwner" --profile AccountBadmin
```

请注意以下几点：

- --Profile 参数指定 AccountBadmin 配置文件，因此该对象由账户 B 拥有。
- 参数 grant-full-control 根据存储桶策略的要求向存储桶拥有者授予对象的完全控制权限。
- --body 参数标识要上传的源文件。例如，如果该文件在 Windows 计算机上的 C: 驱动器中，您应指定 c:\HappyFace.jpg。

## 步骤 3：执行账户 C 任务

在前面的步骤中，账户 A 已创建一个角色 examplerole，与账户 C 建立了信任。这使账户 C 中的用户可以访问账户 A。在此步骤中，账户 C 管理员创建一个用户 (Dave)，向他委托从账户 A 收到的 sts:AssumeRole 权限。这使 Dave 可以担任 examplerole 并临时获取对账户 A 的访问权限。账户 A 附加到该角色的访问策略将限制 Dave 在访问账户 A 时可以执行的操作，具体而言，就是获取 examplebucket 中的对象。

### 步骤 3.1：在账户 C 中创建一个用户并委托担任 exempleroles 的权限

1. 首先使用账户 C 的 IAM 用户登录 URL，以 AccountCadmin 用户身份登录 AWS 管理控制台。
2. 在 IAM 控制台中，创建用户 Dave。

有关说明，请参阅 IAM 用户指南 中的 [创建 IAM 用户 \(AWS 管理控制台\)](#)。
3. 记下 Dave 证书。Dave 需要这些证书才能担任 exempleroles 角色。
4. 为 IAM 用户 Dave 创建一个内联策略，向 Dave 委托担任账户 A 中的 exempleroles 角色的 sts:AssumeRole 权限。
  - a. 在左侧导航窗格中，单击 Users。
  - b. 单击用户名 Dave。
  - c. 在用户详细信息页面上，选择 Permissions 选项卡，然后展开 Inline Policies 部分。
  - d. 选择 click here (或 Create User Policy)。
  - e. 单击 Custom Policy (自定义策略)，然后单击 Select (选择)。
  - f. 在 Policy Name 字段中为策略输入一个名称。
  - g. 将以下策略复制到 Policy Document 字段中。

您需要通过提供账户 A ID 来更新策略。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["sts:AssumeRole"],  
            "Resource": "arn:aws:iam::AccountA-ID:role/exempleroles"  
        }  
    ]  
}
```

- h. 单击 Apply Policy

5. 通过添加另一个配置文件 AccountCDave 将 Dave 的证书保存到 AWS CLI 的配置文件。

```
[profile AccountCDave]  
aws_access_key_id = UserDaveAccessKeyID  
aws_secret_access_key = UserDaveSecretAccessKey  
region = us-west-2
```

### 步骤 3.2：担任角色 (exempleroles) 并访问对象

现在 Dave 可以访问账户 A 拥有的存储桶中的对象，如下所示：

- Dave 首先使用自己的凭证担任 exempleroles。这会返回临时凭证。
- Dave 随后使用临时凭证访问账户 A 的存储桶中的对象。

1. 在命令提示符处，使用 AccountCDave 配置文件执行以下 AWS CLI assume-role 命令。

您需要通过提供账户 A 的 ID (其中定义了 exempleroles) 在命令中更新 ARN 值。

```
aws sts assume-role --role-arn arn:aws:iam::accountA-ID:role/exempleroles --profile  
AccountCDave --role-session-name test
```

在响应中，AWS Security Token Service (STS) 返回临时安全证书（访问密钥 ID、秘密访问密钥和会话令牌）。

- 将临时安全证书保存在 AWS CLI 配置文件中的 TempCred 配置文件下。

```
[profile TempCred]
aws_access_key_id = temp-access-key-ID
aws_secret_access_key = temp-secret-access-key
aws_session_token = session-token
region = us-west-2
```

- 在命令提示符处，使用临时证书执行以下 AWS CLI 命令以访问对象。例如，该命令指定 head-object API 以检索 HappyFace.jpg 对象的对象元数据。

```
aws s3api get-object --bucket examplebucket --key HappyFace.jpg SaveFileAs.jpg --
profile TempCred
```

因为附加到 exempleroles 的访问策略允许执行这些操作，所以 Amazon S3 会处理请求。您可以对存储桶中的任何其他对象尝试任何其他操作。

如果您尝试任何其他操作（例如 get-object-acl），则系统会拒绝权限，因为不允许角色执行该操作。

```
aws s3api get-object-acl --bucket examplebucket --key HappyFace.jpg --profile TempCred
```

我们使用用户 Dave 担任角色，并使用临时证书访问对象。它还可以是账户 C 中访问 examplebucket 中的对象的应用程序。应用程序可以获取临时安全证书，账户 C 可以向应用程序委托担任 exempleroles 的权限。

## 步骤 4：清除

- 完成测试之后，您可以执行以下操作来进行清理。
  - 使用账户 A 证书登录 AWS 管理控制台（[AWS 管理控制台](#)），执行以下操作：
    - 在 Amazon S3 控制台中，删除附加到 *examplebucket* 的存储桶策略。在存储桶 Properties（属性）中，删除 Permissions（权限）部分中的策略。
    - 如果该存储桶是为此练习而创建的，请在 Amazon S3 控制台中删除对象，然后删除存储桶。
    - 在 IAM 控制台中，删除您在账户 A 中创建的 exempleroles。
    - 在 IAM 控制台中，删除 AccountAadmin 用户。
- 使用账户 B 证书登录 AWS 管理控制台（[AWS 管理控制台](#)）。在 IAM 控制台中，删除用户 AccountBadmin。
- 使用账户 C 证书登录 AWS 管理控制台（[AWS 管理控制台](#)）。在 IAM 控制台中，删除用户 AccountCadmin 和用户 Dave。

## 相关资源

- IAM 用户指南 中的 [创建向 IAM 用户委派权限的角色](#)。
- IAM 用户指南 中的 [教程：使用 IAM 角色委派跨 AWS 账户的访问权限](#)。
- IAM 用户指南 中的 [使用策略](#)。

# 使用存储桶策略和用户策略。

## 主题

- 访问策略语言概述 (p. 279)
- 存储桶策略示例 (p. 304)
- 用户策略示例 (p. 312)

存储桶策略和用户策略作为两种访问策略选项，可用于授予对 Amazon S3 资源的权限。这二者均使用基于 JSON 的访问策略语言。这一部分中的主题介绍策略语言核心元素，重点讲述 Amazon S3 特定的详细信息，并且提供存储桶和用户策略的示例。

## Important

我们建议您首先阅读以下介绍性主题，这些主题讲解了管理 Amazon S3 资源访问的基本概念和选项。有关更多信息，请参阅 [Amazon S3 资源访问权限管理介绍 \(p. 242\)](#)。

## 访问策略语言概述

这一部分中的主题描述用于存储桶和用户策略的基本元素在 Amazon S3 中的用法。有关完整策略语言信息，请参阅 IAM 用户指南 中的主题：[IAM 策略概述](#) 和 [AWS IAM 策略参考](#)。

## Note

存储桶策略的大小限制为 20 KB。

## 访问策略中的常用元素

就其最基本的意义而言，策略包含以下元素：

- 资源 – 存储桶和对象是您能够允许或拒绝权限的 Amazon S3 资源。在策略中，使用 Amazon 资源名称 (ARN) 标识资源。
- 操作 – 对于每个资源，Amazon S3 都支持一组操作。允许 (或拒绝) 的资源操作通过操作关键字指定 (请参阅 [在策略中指定权限 \(p. 282\)](#))。

例如，使用 s3>ListBucket 权限，用户可对 Amazon S3 执行 [GET 存储桶 \(列出对象\)](#) 操作。

- Effect – 当用户请求特定操作 (可以是允许或拒绝) 时的效果。

如果没有显式授予 (允许) 对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。

- Principal – 允许访问语句中的操作和资源的账户或用户。在存储桶策略中，委托人是作为权限获得者的用户、账户、服务或其他实体。

下面的存储桶策略示例展示了上述常用策略元素。此策略为 **Account-ID** 账户中的用户 Dave 授予针对 examplebucket 存储桶的 s3:GetObject、s3:GetBucketLocation 和 s3>ListBucket Amazon S3 权限。

```
{  
    "Version": "2012-10-17",  
    "Id": "ExamplePolicy01",  
    "Statement": [  
        {  
            "Sid": "ExampleStatement01",  
            "Effect": "Allow",  
            "Principal": "dave@example.com",  
            "Action": "s3:GetObject",  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        },  
        {  
            "Sid": "ExampleStatement02",  
            "Effect": "Allow",  
            "Principal": "dave@example.com",  
            "Action": "s3:GetBucketLocation",  
            "Resource": "arn:aws:s3:::examplebucket"  
        },  
        {  
            "Sid": "ExampleStatement03",  
            "Effect": "Allow",  
            "Principal": "dave@example.com",  
            "Action": "s3>ListBucket",  
            "Resource": "arn:aws:s3:::examplebucket"  
        }  
    ]  
}
```

```
"Effect": "Allow",
"Principal": {
    "AWS": "arn:aws:iam::Account-ID:user/Dave"
},
>Action": [
    "s3:GetObject",
    "s3:GetBucketLocation",
    "s3>ListBucket"
],
"Resource": [
    "arn:aws:s3::::examplebucket/*",
    "arn:aws:s3::::examplebucket"
]
}
]
```

有关这些访问策略元素的更多信息，请参阅以下主题：

- [在策略中指定资源 \(p. 280\)](#)
- [在策略中指定委托人 \(p. 281\)](#)
- [在策略中指定权限 \(p. 282\)](#)
- [在策略中指定条件 \(p. 286\)](#)

以下主题提供更多策略示例：

- [存储桶策略示例 \(p. 304\)](#)
- [用户策略示例 \(p. 312\)](#)

## 在策略中指定资源

以下是用于标识 AWS 中任何资源的 Amazon 资源名称 (ARN) 常用格式。

```
arn:partition:service:region:namespace:relative-id
```

对于您的 Amazon S3 资源：

- aws 是通用分区名称。如果您的资源位于中国（北京）区域，则分区名为 aws-cn。
- s3 是服务。
- 不指定区域和命名空间。
- 对于 Amazon S3，它可以是 bucket-name 或 bucket-name/object-key。可以使用通配符。

然后 Amazon S3 资源的 ARN 格式缩减成：

```
arn:aws:s3::::bucket_name
arn:aws:s3::::bucket_name/key_name
```

以下是 Amazon S3 资源 ARN 的示例。

- 该 ARN 标识 examplebucket 存储桶中的 /developers/design\_info.doc 对象。

```
arn:aws:s3::::examplebucket/developers/design_info.doc
```

- 您可将通配符作为资源 ARN 的一部分。您可在任何 ARN 分段 (用冒号分隔的部分) 中使用通配符 (\*) 和 (? )。星号 (\*) 表示 0 个或多个字符的任意组合，问号 (?) 表示任何单个字符。您可在每个分段中使用多个 \* 或 ? 字符，但通配符不能跨分段。
  - 此 ARN 在其相对 ID 段使用通配符 \* 来标识 examplebucket 存储桶中的所有对象。

```
arn:aws:s3:::examplebucket/*
```

此 ARN 使用 \* 来表示所有 Amazon S3 资源 (您账户中的所有 S3 存储桶和对象)。

```
arn:aws:s3:::*
```

- 此 ARN 在相对 ID 段同时使用通配符 \* 和 ?。它标识存储桶 (例如 example1bucket、example2bucket、example3bucket 等) 中的所有对象。

```
arn:aws:s3:::example?bucket/*
```

- 您也可以在 Amazon S3 ARN 中使用策略变量。在策略评估时，这些预定义变量被它们的相应值替换。假设您将存储桶组织为一个文件夹的集合，每个用户拥有一个文件夹。文件夹名称与用户名相同。要为他们的文件夹授予用户权限，您可以在资源 ARN 中指定策略变量：

```
arn:aws:s3:::bucket_name/developers/${aws:username}/
```

在运行时，当评估此策略时，资源 ARN 中的变量 \${aws:username} 替换为发出此请求的用户名。

要找到 S3 存储桶的 ARN，您可以查看 Amazon S3 控制台的存储桶策略或 CORS 配置权限页面。有关更多信息，请参阅[如何添加 S3 存储桶策略？](#)或[如何通过 CORS 允许跨域资源共享？](#)(在 Amazon Simple Storage Service 控制台用户指南中)。

有关 ARN 的更多信息，请参阅以下内容：

- IAM 用户指南 中的 [Resource](#)
- IAM 用户指南 中的 [IAM 策略变量概述](#)
- AWS General Reference 中的 [ARN](#)

有关其他访问策略语言元素的更多信息，请参阅[访问策略语言概述 \(p. 279\)](#)。

## 在策略中指定委托人

Principal 元素用于指定被允许或拒绝访问资源的用户、账户、服务或其他实体。下面是指定 Principal 的示例。有关更多信息，请参阅 IAM 用户指南 中的[委托人](#)。

- 要授予 AWS 账户权限，则使用以下格式标识此账户。

```
"AWS": "account-ARN"
```

例如：

```
"Principal": {"AWS": "arn:aws:iam::AccountNumber-WithoutHyphens:root"}
```

Amazon S3 还支持规范用户 ID，这是 AWS 账户 ID 的模糊形式。可使用以下格式指定此 ID。

```
"CanonicalUser": "64-digit-alphanumeric-value"
```

例如：

```
"Principal": {"CanonicalUser": "64-digit-alphanumeric-value"}
```

有关如何查找您的账户的规范用户 ID 的信息，请参阅[查找账户的规范用户 ID](#)。

**Important**

在策略中使用规范用户 ID 时，Amazon S3 可能会将规范 ID 更改为相应的 AWS 账户 ID。这不会影响策略，因为这两个 ID 标识相同的账户。

- 要向您的账户中的 IAM 用户授予权限，您必须提供 "AWS": "user-ARN" 名称值对。

```
"Principal": {"AWS": "arn:aws:iam::account-number-without-hyphens:user/username"}
```

- 要授予每个人权限，也称为匿名访问，请将 Principal 值设置为通配符 "\*"。例如，如果您将存储桶配置为网站，您将需要使该存储桶中的所有对象都可公开访问。以下项目具有相同的效果：

```
"Principal": "*"
```

```
"Principal": {"AWS": "*"}"
```

**Warning**

在授予对您的 S3 存储桶的匿名访问权限时应谨慎使用。如果您授予匿名访问权限，那么世界上任何人都可以访问您的存储桶。我们强烈建议您绝对不要授予对 S3 存储桶的任何类型的匿名写入权限。

- 您可要求您的用户使用 Amazon CloudFront URL (非 Amazon S3 URL) 访问您的 Amazon S3 内容。为此，创建一个 CloudFront 来源访问标识 (OAI)，然后更改对您的存储桶或存储桶中的对象的权限。在 Principal 语句中指定 OAI 的格式如下所示：

```
"Principal": {"CanonicalUser": "Amazon S3 Canonical User ID assigned to origin access identity"}
```

有关更多信息，请参阅[使用源访问身份限制](#)在Amazon CloudFront 开发人员指南中对您的 Amazon S3 内容的访问。

有关其他访问策略语言元素的更多信息，请参阅[访问策略语言概述 \(p. 279\)](#)。

## 在策略中指定权限

Amazon S3 定义了可在策略中指定的一组权限。这些是关键字，其中每一个均映射到特定 Amazon S3 操作 (请参阅 Amazon Simple Storage Service API Reference 中的[存储桶上的操作](#)和[对象上的操作](#))。

### 主题

- [对象操作的权限 \(p. 282\)](#)
- [与存储桶操作相关的权限 \(p. 284\)](#)
- [与存储桶子资源操作相关的权限 \(p. 284\)](#)

## 对象操作的权限

本节提供可在策略中指定的对象操作的权限列表。

## 对象操作的 Amazon S3 权限

权限	Amazon S3 操作
s3:AbortMultipartUpload	<a href="#">启动分段上传</a>
s3:DeleteObject	<a href="#">DELETE Object</a>
s3:DeleteObjectTagging	<a href="#">DELETE Object 标签</a>
s3:DeleteObjectVersion	<a href="#">DELETE Object (针对特定版本的对象)</a>
s3:DeleteObjectVersionTagging	<a href="#">DELETE Object 标签 (针对特定版本的对象)</a>
s3:GetObject	<a href="#">GET 对象、HEAD 对象、SELECT 对象内容</a>  如果在启用了版本控制的存储桶上授予此权限，则始终会获得最新版本的数据。
s3:GetObjectAcl	<a href="#">GET Object ACL</a>
s3:GetObjectTagging	<a href="#">GET Object 标签</a>
s3:GetObjectTorrent	<a href="#">GET Object torrent</a>
s3:GetObjectVersion	<a href="#">GET Object、HEAD Object</a>  要授予对版本特定对象数据的权限，必须授予此权限。即，如果您在发出其中任何请求时指定版本号，则需要此 Amazon S3 权限。
s3:GetObjectVersionAcl	<a href="#">GET ACL (针对特定版本的对象)</a>
s3:GetObjectVersionTagging	<a href="#">GET Object 标签 (针对特定版本的对象)</a>
s3:GetObjectTorrent	<a href="#">GET Object Torrent versioning</a>
s3>ListMultipartUploads	<a href="#">列出分段</a>
s3:PutObject	<a href="#">PUT Object、POST Object、启动分段上传、上传分段、完成分段上传、PUT Object - Copy</a>
s3:PutObjectAcl	<a href="#">PUT Object ACL</a>
s3:PutObjectTagging	<a href="#">PUT Object 标签</a>
s3:PutObjectVersionAcl	<a href="#">PUT 对象 ACL (针对特定版本的对象)</a>
s3:PutObjectVersionTagging	<a href="#">PUT Object 标签 (针对特定版本的对象)</a>
s3:RestoreObject	<a href="#">POST Object restore</a>

以下存储桶策略示例授予用户 (Dave) s3:PutObject 和 s3:PutObjectAcl 权限。如果移除 Principal 元素，则可将此策略附加到用户。这些是对象操作，因此 Resource ARN 的 relative-id 部分标识对象 (examplebucket/\*)。有关更多信息，请参阅 [在策略中指定资源 \(p. 280\)](#)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
            }
        }
    ]
}
```

```
        },
        "Action": ["s3:PutObject", "s3:PutObjectAcl"],
        "Resource": "arn:aws:s3:::examplebucket/*"
    }
}
```

您可使用通配符来授予对所有 Amazon S3 操作的权限。

```
"Action": "*"
```

## 与存储桶操作相关的权限

本节提供与可在策略中指定的对象操作相关的权限列表。

### 与存储桶操作相关的 Amazon S3 权限

权限关键字	涉及的 Amazon S3 操作
s3:CreateBucket	PUT Bucket
s3:DeleteBucket	DELETE Bucket
s3>ListBucket	GET Bucket ( 列出对象 ) , HEAD Bucket
s3>ListBucketVersions	GET Bucket Object versions
s3>ListAllMyBuckets	GET 服务
s3>ListBucketMultipleUploads	列出分段上传

以下用户策略示例授予用户 s3:CreateBucket、s3>ListAllMyBuckets 和 s3:GetBucketLocation 权限。注意，对于所有这些权限，均将 Resource ARN 的 relative-id 部分设置为“\*”。对于其他所有存储桶操作，必须指定存储桶名称。有关更多信息，请参阅 [在策略中指定资源 \(p. 280\)](#)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket", "s3>ListAllMyBuckets", "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::/*"
            ]
        }
    ]
}
```

如果您的用户要使用控制台来查看存储桶，并查看这些存储桶中的内容，用户必须具有 s3>ListAllMyBuckets 和 s3:GetBucketLocation 权限。有关示例，请参阅 [编写 IAM 策略：如何授予对 Amazon S3 存储桶的访问权限](#) 中的“控制台访问策略”。

## 与存储桶子资源操作相关的权限

本节提供与可在策略中指定的存储桶子资源操作相关的权限列表。

### 与存储桶子资源操作相关的 Amazon S3 权限

权限	涉及的 Amazon S3 操作
s3:DeleteBucketPolicy	<a href="#">DELETE Bucket policy</a>
s3:DeleteBucketWebsite	<a href="#">DELETE Bucket website</a>
s3:GetAccelerateConfiguration	<a href="#">GET Bucket 加速</a>
s3:GetAnalyticsConfiguration	<a href="#">GET Bucket 分析、List Bucket 分析配置</a>
s3:GetBucketAcl	<a href="#">GET Bucket acl</a>
s3:GetBucketCORS	<a href="#">GET Bucket cors</a>
s3:GetBucketLocation	<a href="#">GET Bucket location</a>
s3:GetBucketLogging	<a href="#">GET Bucket logging</a>
s3:GetBucketNotification	<a href="#">GET Bucket notification</a>
s3:GetBucketPolicy	<a href="#">GET Bucket policy</a>
s3:GetBucketRequestPayment	<a href="#">GET Bucket requestPayment</a>
s3:GetBucketTagging	<a href="#">GET Bucket 标记</a>
s3:GetBucketVersioning	<a href="#">GET Bucket versioning</a>
s3:GetBucketWebsite	<a href="#">GET Bucket website</a>
s3:GetEncryptionConfiguration	<a href="#">GET Bucket 加密</a>
s3:GetInventoryConfiguration	<a href="#">GET Bucket 清单、List Bucket 清单配置</a>
s3:GetLifecycleConfiguration	<a href="#">GET Bucket lifecycle</a>
s3:GetMetricsConfiguration	<a href="#">GET Bucket 指标、List Bucket 指标配置</a>
s3:GetReplicationConfiguration	<a href="#">GET Bucket 复制</a>
s3:PutAccelerateConfiguration	<a href="#">PUT Bucket 加速</a>
s3:PutAnalyticsConfiguration	<a href="#">PUT Bucket 分析、DELETE Bucket 分析</a>
s3:PutBucketAcl	<a href="#">PUT Bucket acl</a>
s3:PutBucketCORS	<a href="#">PUT Bucket cors、DELETE Bucket cors</a>
s3:PutBucketLogging	<a href="#">PUT Bucket logging</a>
s3:PutBucketNotification	<a href="#">PUT Bucket notification</a>
s3:PutBucketPolicy	<a href="#">PUT 存储桶策略</a>
s3:PutBucketRequestPayment	<a href="#">PUT Bucket requestPayment</a>
s3:PutBucketTagging	<a href="#">DELETE Bucket 标签、PUT Bucket 标签</a>
s3:PutBucketVersioning	<a href="#">PUT Bucket versioning</a>
s3:PutBucketWebsite	<a href="#">PUT Bucket website</a>

权限	涉及的 Amazon S3 操作
s3:PutEncryptionConfiguration	PUT 存储桶加密、 DELETE 存储桶加密
s3:PutInventoryConfiguration	PUT Bucket 清单、 DELETE Bucket 清单
s3:PutLifecycleConfiguration	PUT 存储桶生命周期、 DELETE 存储桶生命周期
s3:PutMetricsConfiguration	PUT Bucket 指标、 DELETE Bucket 指标
s3:PutReplicationConfiguration	PUT 存储桶复制、 DELETE 存储桶复制

以下用户策略授予用户 Dave 对 examplebucket 存储桶的 s3:GetBucketAcl 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::Account-ID:user/Dave"
            },
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetBucketAcl"
            ],
            "Resource": "arn:aws:s3:::examplebucket"
        }
    ]
}
```

可通过显式调用 DELETE Object API 或配置其生命周期来删除对象 (请参阅[对象生命周期管理 \(p. 104\)](#))，以便 Amazon S3 能够在对象生命周期已过时将它们删除。要显式阻止用户或账户删除对象，必须显式拒绝授予他们 s3:DeleteObject、s3:DeleteObjectVersion 和 s3:PutLifecycleConfiguration 权限。默认状态下，用户没有权限。但在创建用户，将用户添加到组以及授予他们权限时，用户有可能获得您并不打算授予的某些权限。这是可使用显式拒绝的情况，显式拒绝将取代用户可能拥有的其他所有权限，并且拒绝对特定操作的用户权限。

## 在策略中指定条件

访问策略语言可使您在授予权限时指定条件。在 Condition 元素 (或 Condition 块) 中，可以指定策略生效的条件。在可选的 Condition 元素中，您创建的表达式应使用布尔值运算符 (等于、小于等)，以使您指定的条件与请求中的值相匹配。例如，当授予用户上传对象的权限时，存储桶拥有者可通过添加 StringEquals 条件要求此对象可公开读取，如下所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ],
            "Condition": {

```

```
        "StringEquals": {
            "s3:x-amz-acl": [
                "public-read"
            ]
        }
    }
}
```

Condition 块指定应用于指定的键值对 "s3:x-amz-acl":["public-read"] 的 StringEquals 条件。有一组预定义键可用于表达条件。此示例使用 s3:x-amz-acl 条件键。此条件要求用户包含 x-amz-acl 标头，并且值 public-read 位于每个 PUT 对象请求中。

有关使用访问策略语言来指定条件的更多信息，请参阅 IAM 用户指南 中的[条件](#)。

以下主题描述 AWS 范围和 Amazon S3 特定的条件键并提供示例策略。

#### 主题

- [可用条件键 \(p. 287\)](#)
- [针对对象操作的 Amazon S3 条件键 \(p. 289\)](#)
- [针对存储桶操作的 Amazon S3 条件键 \(p. 298\)](#)

## 可用条件键

用于在 Amazon S3 访问策略中指定条件的预定义键可按以下方式分类：

- AWS 范围的键 – AWS 提供一组常用键，所有支持策略的 AWS 服务均支持这些键。所有服务常用的这些键称作 AWS 范围的键，它们使用前缀 aws:。有关 AWS 范围的键的列表，请参阅 IAM 用户指南 中的[可用的条件键](#)。还存在特定于 Amazon S3 的键，这些键使用前缀 s3:。下一个项目符号项中将讨论特定于 Amazon S3 的键。

新条件键 aws:sourceVpc 和 aws:sourceVpc 在 VPC 终端节点的存储桶策略中使用。有关使用这些条件键的示例，请参阅 [Amazon S3 的 VPC 终端节点的示例存储桶策略 \(p. 310\)](#)。

以下示例存储桶策略允许当请求来自特定 IP 地址范围 (192.168.143.\*) 时，经过身份验证的用户有权使用 s3:GetObject 操作，除非此 IP 地址为 192.168.143.188。在条件块中，IpAddress 和 NotIpAddress 为条件，每个条件均提供一个键-值对用于评估。此示例中的两个键-值对均使用 aws:SourceIp AWS 范围内的键。

#### Note

请注意，在条件中指定的 IPAddress 和 NotIpAddress 键值使用 RFC 4632 中描述的 CIDR 表示法。有关详细信息，请转到 <http://www.rfc-editor.org/rfc/rfc4632.txt>。

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action": ["s3:GetObject"],
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "IpAddress": {

```

```
        "aws:SourceIp": "192.168.143.0/24"
    },
    "NotIpAddress" : {
        "aws:SourceIp": "192.168.143.188/32"
    }
}
]
```

- 特定于 Amazon S3 的键 – 除了 AWS 范围的键之外，下面还有一些条件键，这些键仅适用于向 Amazon S3 授予特定权限的上下文。这些特定于 Amazon S3 的键使用前缀 s3:。

- s3:x-amz-acl
- s3:x-amz-copy-source
- s3:x-amz-metadata-directive
- s3:x-amz-server-side-encryption
- s3:VersionId
- s3:LocationConstraint
- s3:delimiter
- s3:max-keys
- s3:prefix
- s3:x-amz-server-side-encryption-aws-kms-key-id
- s3:ExistingObjectTag/<tag-key>

有关使用对象标签相关条件键的策略的示例，请参阅[对象标签和访问控制策略 \(p. 98\)](#)。

- s3:RequestObjectTagKeys
- s3:RequestObjectTag/<tag-key>

例如，如果请求包含可使对象公开可读的 x-amz-acl 标头，则以下存储桶策略允许两个 AWS 账户具有 s3:PutObject 权限。

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid":"AddCannedAcl",
            "Effect":"Allow",
            "Principal": {
                "AWS": [ "arn:aws:iam::account1-ID:root", "arn:aws:iam::account2-ID:root" ]
            },
            "Action":["s3:PutObject"],
            "Resource": [ "arn:aws:s3:::examplebucket/*" ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl":["public-read"]
                }
            }
        }
    ]
}
```

Condition 块使用 StringEquals 条件，并且提供键值对 "s3:x-amz-acl": ["public-read"]，以进行评估。在该键值对中，s3:x-amz-acl 是特定于 Amazon S3 的键，如前缀 s3: 所示。

### Important

并非所有条件对所有操作都有意义。例如，在授予 s3:CreateBucket Amazon S3 权限时，`s3:LocationConstraint` 条件是有意义的，但对于 s3:GetObject 权限没有意义。Amazon S3 可测试是否存在此类涉及 Amazon S3 特定条件的语义错误。但如果要创建针对 IAM 用户的策略，并且包含了语义无效的 Amazon S3 条件，则不报告错误，因为 IAM 无法验证 Amazon S3 条件。

下一节介绍可用于授予针对存储桶和对象操作的条件权限的条件键。此外，还有与 Amazon S3 签名版本 4 身份验证有关的条件键。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [Amazon S3 签名版本 4 身份验证特定的策略键](#)。

## 针对对象操作的 Amazon S3 条件键

下表显示了可对哪些 Amazon S3 操作使用哪些 Amazon S3 条件。在该表之后提供有策略示例。注意下面有关下表中所述的 Amazon S3 特定条件键的内容：

- 这些条件键名称的前缀为 `s3:`。例如：  
`s3:x-amz-acl`
  -
- 每个条件键均映射到可设置条件的 API 所允许的相同名称请求标头。即，这些条件键指定同名请求标头的行为。例如：
  - 条件键 `s3:x-amz-acl` 可用于对 `s3:PutObject` 权限授予条件权限，它定义 PUT Object API 支持的 `x-amz-acl` 请求标头的行为。
  - 条件键 `s3:VersionId` 可用于对 `s3:GetObjectVersion` 权限授予条件权限，它定义您在 GET Object 请求中设置的查询参数 `versionId` 的行为。

许可	适用的条件键 (或关键字)	描述
<code>s3:PutObject</code>	<ul style="list-style-type: none"><li>◦ <code>s3:x-amz-acl</code> (针对标准 ACL 权限)</li><li>◦ <code>s3:x-amz-grant-permission</code> (针对显式权限)，其中 <code>permission</code> 可以是： <code>read, write, read-acp, write-acp, full-control</code></li></ul>	<p><code>PUT Object</code> 操作允许在授予基于 ACL 的权限时可使用的访问控制列表 (ACL) 特定标头。通过使用这些键，存储桶拥有者可设置条件，要求用户上传对象时需具有特定访问权限。</p> <p>有关策略示例，请参阅 <a href="#">示例 1：授予 s3:PutObject 权限，指定了要求存储桶拥有者获得完全控制的条件 (p. 294)</a>。</p> <p>有关 ACL 的更多信息，请参阅 <a href="#">访问控制列表 (ACL) 概述 (p. 333)</a>。</p>
	<code>s3:x-amz-copy-source</code>	<p>要复制对象，请使用 PUT Object API (请参阅 <a href="#">PUT Object</a>) 并使用 <code>x-amz-copy-source</code> 标头指定源。通过使用此键，存储桶拥有者可将复制源限定到特定存储桶、存储桶中的特定文件夹，或者存储桶中的特定对象。</p> <p>有关策略示例，请参阅 <a href="#">示例 3：授予从限定复制源复制对象的 s3:PutObject 权限 (p. 296)</a>。</p>
	<code>s3:x-amz-server-side-encryption</code>	上传对象时，可以使用 <code>x-amz-server-side-encryption</code> 标头。

许可	适用的条件键 (或关键字)	描述
		<p>求 Amazon S3 在保存对象时使用 AWS Key Management Service (AWS KMS) 或 Amazon S3 托管的信封加密密钥加密对象 (请参阅<a href="#">使用服务器端加密保护数据 (p. 345)</a>)。</p> <p>当授予 <code>s3:PutObject</code> 权限时，存储桶拥有者可使用此键添加条件，要求用户在请求中指定此标头。存储桶拥有者可通过授予此类条件权限，确保在保存用户上传的对象时进行加密。</p> <p>有关策略示例，请参阅<a href="#">示例 1：授予 s3:PutObject 权限，指定了要求存储桶拥有者获得完全控制的条件 (p. 294)</a>。</p>
	<code>s3:x-amz-server-side-encryption-aws-kms-key-id</code>	<p>在上传对象时，可以使用 <code>x-amz-server-side-encryption-aws-kms-key-id</code> 标头请求 Amazon S3 使用指定的 AWS KMS 密钥加密对象 (参阅<a href="#">使用具有 AWS KMS 托管密钥的服务器端加密 (SSE-KMS) 保护数据 (p. 346)</a>)。</p> <p>在授予 <code>s3:PutObject</code> 权限时，存储桶拥有者可以使用此键添加一个条件，将用于对象加密的 AWS KMS 密钥 ID 限制为特定值。</p> <p>存储桶拥有者可通过授予此类条件权限，确保在保存用户上传的对象时使用特定密钥进行加密。</p> <p>您在策略中指定的 AWS KMS 密钥必须使用以下格式：</p> <pre>arn:aws:kms:<b>region:acct-id</b>:key/<b>key-id</b></pre>
	<code>s3:x-amz-metadata-directive</code>	<p>在使用 PUT Object API 复制对象时 (请参阅<a href="#">PUT Object</a>)，可以选择添加 <code>x-amz-metadata-directive</code> 标头，以便指定您是要从源对象复制对象元数据，还是要将该对象元数据替换为在请求中提供的元数据。</p> <p>通过使用此键值存储桶，拥有者可添加在上传对象时将执行某种行为的条件。</p> <p>有效值：<code>COPY   REPLACE</code>。默认为 <code>COPY</code>。</p>

许可	适用的条件键 (或关键字)	描述
	s3:x-amz-storage-class	<p>默认情况下，<code>s3:PutObject</code> 使用 STANDARD 存储类存储对象，但您可以使用 <code>x-amz-storage-class</code> 请求标头指定其他存储类。</p> <p>在授予 <code>s3:PutObject</code> 权限时，您可以使用 <code>s3:x-amz-storage-class</code> 条件键限制存储上传对象时使用的存储类。有关存储类的更多信息，请参阅<a href="#">存储类</a>。</p> <p>有关策略示例，请参阅<a href="#">示例 5：将上传对象限制为具有特定存储类的对象 (p. 298)</a>。</p> <p>有效值：STANDARD   STANDARD_IA   REDUCED_REDUNDANCY。默认为 STANDARD。</p>
	<ul style="list-style-type: none"> <li>• <code>s3:RequestObjectTagKeys</code></li> <li>• <code>s3:RequestObjectTag/&lt;tag-key&gt;</code></li> </ul>	<p>通过使用此条件键，您可以通过限制请求中允许使用的对象标签来限制 <code>s3:PutObject</code> 操作的权限。有关使用这些条件键的示例，请参阅<a href="#">对象标签和访问控制策略 (p. 98)</a>。</p>
<code>s3:PutObjectAcl</code>	<ul style="list-style-type: none"> <li>• <code>s3:x-amz-acl</code> (针对标准 ACL 权限)</li> <li>• <code>s3:x-amz-grant-permission</code> (针对显式权限)，其中 <code>permission</code> 可以是： <code>read, write, read-acp, write-acp, grant-full-control</code></li> </ul>	<p><a href="#">PUT Object ACL API</a> 在特定对象上设置访问控制列表 (ACL)。此操作支持与 ACL 相关的标头。当授予此权限时，存储桶拥有者可使用这些键添加将要求某些权限的条件。有关 ACL 的更多信息，请参阅<a href="#">访问控制列表 (ACL) 概述 (p. 333)</a>。</p> <p>例如，无论谁应对此对象，存储桶拥有者都可能需要保持对此对象的控制。要实现这一点，存储桶拥有者可使用其中一个键添加条件，要求用户包含对存储桶拥有者的特定权限。</p>
	<code>s3:ExistingObjectTag/&lt;tag-key&gt;</code>	通过使用此条件键，您可以将 <code>s3:PutObjectAcl</code> 的操作权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
<code>s3:PutObjectTagging</code>	<ul style="list-style-type: none"> <li>• <code>s3:RequestObjectTagKeys</code></li> <li>• <code>s3:RequestObjectTag/&lt;tag-key&gt;</code></li> </ul>	<p>通过使用此条件键，您可以通过限制请求中允许使用的对象标签来限制 <code>s3:PutObjectTagging</code> 操作的权限。有关使用这些条件键的示例，请参阅<a href="#">对象标签和访问控制策略 (p. 98)</a>。</p>

许可	适用的条件键 (或关键字)	描述
	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
s3:PutObjectVersionTagging	• s3:RequestObjectTagKeys • s3:RequestObjectTag/ <tag-key>	通过使用此条件键，您可以通过限制请求中允许使用的对象标签来限制 s3:PutObjectVersionTagging 操作的权限。有关使用这些条件键的示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
	s3:VersionId	通过使用此条件键，您可以将 s3:PutObjectVersionTagging 的操作权限限制在特定对象版本上。有关策略示例，请参阅 <a href="#">示例 4：授予对特定对象版本的访问权限 (p. 297)</a> 。
	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
s3:GetObjectVersion	s3:VersionId	该 Amazon S3 权限使用户能够执行一组 Amazon S3 API 操作 (请参阅 <a href="#">对象操作的 Amazon S3 权限 (p. 283)</a> )。对于启用了版本控制的存储桶，可指定检索数据所针对的对象版本。  通过使用此键添加条件，存储桶拥有者可将用户限定于仅访问特定对象版本的数据。有关策略示例，请参阅 <a href="#">示例 4：授予对特定对象版本的访问权限 (p. 297)</a> 。
	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
s3:GetObject	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
s3:GetObjectAcl	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。

许可	适用的条件键 (或关键字)	描述
s3:GetObjectVersionAcl	s3:VersionId	<p>您可以使用 <a href="#">GET Object acl API</a> 来检索特定对象版本的访问控制列表 (ACL)。用户必须具有执行 s3:GetObjectVersionAcl 操作的权限。对于启用了版本控制的存储桶，该 Amazon S3 权限使用户能够获得针对特定对象版本的 ACL。</p> <p>存储桶拥有者可使用此键值添加条件，将用户限定于对象的特定版本。</p>
	s3:ExistingObjectTag/<tag-key>	<p>通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a>。</p>
s3:PutObjectVersionAcl	s3:VersionId	<p>对于启用了版本控制的存储桶，可在 <a href="#">PUT Object acl</a> 请求中指定对象版本，以针对特定对象版本设置 ACL。通过使用此条件，存储桶拥有者可将用户限定于仅针对特定对象版本设置 ACL。</p>
	<ul style="list-style-type: none"> <li>• s3:x-amz-acl (针对标准 ACL 权限)</li> <li>• s3:x-amz-grant-permission (针对显式权限)，其中 <i>permission</i> 可以是： <code>read, write, read-acp, write-acp, grant-full-control</code></li> </ul>	<p>对于启用了版本控制的存储桶，此 Amazon S3 权限使您能够针对特定对象版本设置 ACL。</p> <p>有关这些条件键的描述，请参阅本表中的 s3:PutObjectACL 权限。</p>
	s3:ExistingObjectTag/<tag-key>	<p>通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a>。</p>
s3>DeleteObjectVersion	s3:VersionId	<p>对于启用了版本控制的存储桶，该 Amazon S3 权限使用户能够删除对象的特定版本。</p> <p>存储桶拥有者可使用此键值添加条件，限制用户仅删除对象的特定版本的能力。</p> <p>有关使用此条件键的示例，请参阅 <a href="#">示例 4：授予对特定对象版本的访问权限 (p. 297)</a>。该示例授权 s3:GetObjectVersion 操作，但是，该策略演示此条件键的使用。</p>

许可	适用的条件键 (或关键字)	描述
s3:DeleteObjectTagging	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
s3:DeleteObjectVersionTagging	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
	s3:VersionId	通过使用此条件键，您可以将s3:DeleteObjectVersionTagging的操作权限限制在特定对象版本上。有关策略示例，请参阅 <a href="#">示例 4：授予对特定对象版本的访问权限 (p. 297)</a> 。
s3:GetObjectTagging	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
s3:GetObjectVersionTagging	s3:ExistingObjectTag/ <tag-key>	通过使用此条件键，您可以将权限限制在有特定标签键和值的对象上。有关示例，请参阅 <a href="#">对象标签和访问控制策略 (p. 98)</a> 。
	s3:VersionId	通过使用此条件键，您可以将s3:GetObjectVersionTagging的操作权限限制在特定对象版本上。有关策略示例，请参阅 <a href="#">示例 4：授予对特定对象版本的访问权限 (p. 297)</a> 。

#### 示例 1：授予 s3:PutObject 权限，指定了要求存储桶拥有者获得完全控制的条件

假设账户 A 拥有一个存储桶，而账户管理员想要授予账户 B 中的用户 Dave 上传对象的权限。默认情况下，Dave 上传的对象由账户 B 拥有，而账户 A 对这些对象没有权限。由于存储桶拥有者要支付账单，需要对 Dave 上传的对象具有全部权限。要实现这一目的，账户 A 管理员可向 Dave 授予 s3:PutObject 权限，指定的条件要求请求包含 ACL 特定标头，以及显式授予全部权限或使用标准 ACL (请参阅 [PUT Object](#))。

- 要求在请求中具有 x-amz-full-control 标头，授予存储桶拥有者完全控制权限。

以下存储桶策略向用户 Dave 授予 s3:PutObject 权限，使用 s3:x-amz-grant-full-control 条件键指定了条件，要求此请求包含 x-amz-full-control 标头。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
        "x-amz-grant-full-control": "true"
      }
    }
  ]
}
```

```
        "StringEquals": {
            "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
        }
    }
]
```

#### Note

此示例是关于跨账户权限的。但如果将要获得此权限的 Dave 属于拥有此存储桶的 AWS 账户，则无需这种条件权限，因为 Dave 所属的父账户拥有用户上传的对象。

上述存储桶策略向账户 B 中的用户 Dave 授予条件权限。当该策略生效时，Dave 可通过其他某个策略获得没有任何条件的相同权限。例如，Dave 可属于一个组，并且您授予该组没有任何条件的 s3:PutObject 权限。为避免这些权限漏洞，可通过添加显式拒绝编写更严格的访问策略。在该示例中，如果用户 Dave 的请求没有包含必要标头向存储桶拥有者授予全部权限，则您显式拒绝他的上传权限。显式拒绝始终取代授予的其他任何权限。以下是修订的访问策略示例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
}
```

如果您有两个 AWS 账户，则可使用 AWS Command Line Interface (AWS CLI) 测试此策略。使用 Dave 的凭证附加此策略，通过以下 AWS CLI put-object 命令测试权限。通过添加 --profile 参数提供 Dave 的凭证。通过添加 --grant-full-control 参数可向存储桶拥有者授予完全控制权限。有关设置和使用 AWS CLI 的更多信息，请参阅 [设置用于示例演练的工具 \(p. 256\)](#)。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
--grant-full-control id="AccountA-CanonicalUserID" --profile AccountBUserProfile
```

- 要求 x-amz-acl 标头，带有向存储桶拥有者授予完全控制权限的标准 ACL。

若要求在请求中包含 x-amz-acl 标头，可替换 Condition 块中的键值对，并指定 s3:x-amz-acl 条件键，如下所示。

```
"Condition": {  
    "StringNotEquals": {  
        "s3:x-amz-acl": "bucket-owner-full-control"  
    }  
}
```

要使用 AWS CLI 测试权限，则指定 --acl 参数。然后 AWS CLI 在其发送此请求时添加 x-amz-acl 标头。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg  
--acl "bucket-owner-full-control" --profile AccountBadmin
```

### 示例 2：授予要求使用服务器端加密存储对象的 s3:PutObject 权限

假设账户 A 拥有一个存储桶。账户管理员想要授予账户 A 中的用户 Jane 上传对象的权限，条件是 Jane 始终请求服务器端加密，使 Amazon S3 保存加密的对象。账户 A 管理员可使用所示的 s3:x-amz-server-side-encryption 条件键来完成。Condition 块中的键值对指定 s3:x-amz-server-side-encryption 键。

```
"Condition": {  
    "StringNotEquals": {  
        "s3:x-amz-server-side-encryption": "AES256"  
    }  
}
```

当使用 AWS CLI 测试此权限时，必须使用 --server-side-encryption 参数添加所需的参数。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg --  
server-side-encryption "AES256" --profile AccountBadmin
```

### 示例 3：授予从限定复制源复制对象的 s3:PutObject 权限

在 PUT Object 请求中，如果指定了源对象，则为一个复制操作（请参阅 [PUT Object - Copy](#)）。因此，存储桶拥有者可授予用户复制限定于此源上的对象的权限。例如：

- 允许仅从 sourcebucket 存储桶复制对象。
- 允许从 sourcebucket 存储桶复制对象，并仅复制键名称前缀开头为 public/f 的对象。例如 sourcebucket/public/\*
- 只允许从 sourcebucket 复制特定对象；例如 sourcebucket/example.jpg。

以下存储桶策略授予用户 Dave s3:PutObject 权限，此权限要求他复制对象时需满足以下条件：请求中包含 s3:x-amz-copy-source 标头且标头值指定 /examplebucket/public/\* 键名称前缀。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "cross-account permission to user in your own account",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"  
            },  
            "Action": ["s3:PutObject"],  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        }  
    ]  
}
```

```
        "Resource": "arn:aws:s3:::examplebucket/*"
    },
    {
        "Sid": "Deny your user permission to upload object if copy source is not / bucket/folder",
        "Effect": "Deny",
        "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
        },
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::examplebucket/*",
        "Condition": {
            "StringNotLike": {
                "s3:x-amz-copy-source": "examplebucket/public/*"
            }
        }
    }
]
```

可使用 AWS CLI `copy-object` 命令测试此权限。可通过添加 `--copy-source` 参数指定源，键名称前缀必须与策略中允许的前缀相匹配。您需要使用 `--profile` 参数为用户 Dave 提供凭证。有关设置 AWS CLI 的更多信息，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

```
aws s3api copy-object --bucket examplebucket --key HappyFace.jpg
--copy-source examplebucket/public/PublicHappyFace1.jpg --profile AccountADave
```

注意，上述策略使用 `StringNotLike` 条件。要授予仅复制特定对象的权限，您必须将条件从 `StringNotLike` 更改为 `StringNotEquals`，然后指定所示的对象键。

```
"Condition": {
    "StringNotEquals": {
        "s3:x-amz-copy-source": "examplebucket/public/PublicHappyFace1.jpg"
    }
}
```

#### 示例 4：授予对特定对象版本的访问权限

假设账户 A 拥有启用版本控制的存储桶。该存储桶具有 `HappyFace.jpg` 对象的多个版本。账户管理员现在想要授予用户 (Dave) 仅获得特定对象版本的权限。账户管理员可通过有条件地授予 Dave 所示的 `s3:GetObjectVersion` 权限来实现这一点。Condition 块中的键值对指定 `s3:VersionId` 条件键。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": ["s3:GetObjectVersion"],
            "Resource": "arn:aws:s3:::examplebucket/*/HappyFace.jpg"
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": ["s3:GetObjectVersion"],
            "Resource": "arn:aws:s3:::examplebucket/*/HappyFace.jpg",
            "Condition": {
                "StringNotLike": {
                    "s3:VersionId": "2012-10-17"
                }
            }
        }
    ]
}
```

```

        "Condition": {
            "StringNotEquals": {
                "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
            }
        }
    ]
}

```

在这种情况下，Dave 需要知道确切的对象版本 ID 才能检索该对象。

可使用 AWS CLI `get-object` 命令以及标识特定对象版本的 `--version-id` 参数来测试这些权限。此命令会检索该对象，并将其保存到 `OutputFile.jpg` 文件。

```
aws s3api get-object --bucket examplebucketversionenabled --key HappyFace.jpg
OutputFile.jpg --version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

### 示例 5：将上传对象限制为具有特定存储类的对象

假设账户 A 拥有一个存储桶。账户管理员想要限制 Dave (账户 A 中的一名用户) 只能将使用 STANDARD\_IA 存储类存储的对象上传到此存储桶。账户 A 管理员可以借助 `s3:x-amz-storage-class` 条件键达到这一目的，如下面的示例存储桶策略所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-storage-class": [
                        "STANDARD_IA"
                    ]
                }
            }
        }
    ]
}
```

### 针对存储桶操作的 Amazon S3 条件键

下表显示了可在策略中授予的存储桶操作特定权限列表，并且对于其中的每个权限，该表还显示了可用于指定条件的可用键。

许可	适用的条件键	描述
<code>s3:CreateBucket</code>	<ul style="list-style-type: none"> <li><code>s3:x-amz-acl</code> (针对标准 ACL 权限)</li> <li><code>s3:x-amz-grant-permission</code> (针对显式权限)，其中 <code>permission</code> 可以是：</li> </ul>	创建存储桶 API (请参阅 <a href="#">PUT Bucket</a> ) 支持 ACL 特定的标头。通过使用这些条件键，可要求用户在授予特定权限的请求中设置这些标头。

许可	适用的条件键	描述
	<code>read, write, read-acp, write-acp, full-control</code>	
	<code>s3:LocationConstraint</code>	通过使用此条件键，可将用户限定于在特定 AWS 区域创建存储桶。有关策略示例，请参阅 <a href="#">示例 1：允许用户创建存储桶，但仅在特定区域内创建 (p. 301)</a> 。
<code>s3&gt;ListBucket</code>	<code>s3:prefix</code>	<p>通过使用此条件键，可将 Get Bucket (列出对象) API 的响应限定于具有特定前缀的键名称，请参阅<a href="#">GET Bucket (列出对象)</a>。</p> <p>Get Bucket (列出对象) API 会返回指定存储桶中的对象键列表。此 API 支持 <code>prefix</code> 标头，仅检索具有特定前缀的对象键。此条件键与 <code>prefix</code> 标头相关。</p> <p>例如，Amazon S3 控制台支持使用键名称前缀的文件夹概念。因此，如果您有键名称为 <code>public/object1.jpg</code> 和 <code>public/object2.jpg</code> 的两个对象，则该控制台会在 <code>public</code> 文件夹下显示这些对象。如果使用这些前缀组织对象键，则可授予 <code>s3&gt;ListBucket</code> 权限，条件为允许用户获得具有特定前缀的键名称的列表。</p> <p>有关策略示例，请参阅<a href="#">示例 2：允许用户根据特定前缀获取存储桶中的对象的列表 (p. 302)</a>。</p>
	<code>s3:delimiter</code>	如果使用前缀和分隔符组织对象键名称，则可使用此条件键来要求用户在 Get Bucket (列出对象) 请求中指定 <code>delimiter</code> 参数。在这种情况下，Amazon S3 响应返回的对象键列表将按通用前缀分组。有关使用前缀和分隔符的示例，请参阅 <a href="#">Get Bucket (列出对象)</a> 。
	<code>s3:max-keys</code>	<p>通过使用此条件，要求用户指定 <code>max-keys</code> 参数可限制 Amazon S3 响应 Get Bucket (List Objects) 请求所返回的键数。默认情况下，API 返回最多 1000 个键名称。</p> <p>有关可使用的数字条件的列表，请参阅 IAM 用户指南 中的<a href="#">数字条件运算符</a>。</p>

许可	适用的条件键	描述
s3>ListBucketVersions	s3:prefix	<p>如果您的存储桶启用了版本控制，则可使用 GET Bucket Object versions API (请参阅 <a href="#">GET Bucket Object versions</a>) 来检索所有对象版本的元数据。对于此 API，存储桶拥有者必须在策略中授予 s3&gt;ListBucketVersions 权限。</p> <p>使用此条件键，可要求用户在请求中指定具有特定值的 prefix 参数，将 API 的响应限定于具有特定前缀的键名称。</p> <p>例如，Amazon S3 控制台支持使用键名称前缀的文件夹概念。如果您有键名称为 public/object1.jpg 和 public/object2.jpg 的两个对象，则该控制台会在 public 文件夹下显示这些对象。如果使用这些前缀组织对象键，则可授予 s3&gt;ListBucket 权限，条件为允许用户获得具有特定前缀的键名称的列表。</p> <p>有关策略示例，请参阅 <a href="#">示例 2：允许用户根据特定前缀获取存储桶中的对象的列表 (p. 302)</a>。</p>
	s3:delimiter	如果使用前缀和分隔符组织对象键名称，则可使用此条件键要求用户在 GET Bucket Object versions 请求中指定 delimiter 参数。在这种情况下，Amazon S3 响应返回的对象键列表将按通用前缀分组。
	s3:max-keys	通过使用此条件，要求用户指定 max-keys 参数，可限制 Amazon S3 响应 GET Bucket Object versions 请求所返回的键数。默认情况下，API 返回最多 1000 个键名称。有关可使用的数字条件的列表，请参阅 IAM 用户指南 中的 <a href="#">数字条件运算符</a> 。
s3>PutBucketAcl	<ul style="list-style-type: none"> <li>• s3:x-amz-acl (针对标准 ACL 权限)</li> <li>• s3:x-amz-grant-permission (针对显式权限)，其中 permission 可以是： <code>read, write, read-acp, write-acp, full-control</code></li> </ul>	PUT Bucket acl API (请参阅 <a href="#">PUT Bucket</a> ) 支持 ACL 特定的标头。可使用这些条件键来要求用户在请求中设置这些标头。

### 示例 1：允许用户创建存储桶，但仅在特定区域内创建

假定 AWS 账户管理员想要授予其用户 (Dave) 仅在 南美洲 ( 圣保罗 ) 区域创建存储桶的权限。账户管理员可附加以下用户策略，授予附带条件的 s3:CreateBucket 权限，如下所示。Condition 块中的键值对指定 s3:LocationConstraint 键，以 sa-east-1 区域作为值。

#### Note

在该示例中，存储桶拥有者为其用户之一授予权限，因此可以使用存储桶策略或用户策略。此示例显示了用户策略。

有关 Amazon S3 区域的列表，请参阅 Amazon Web Services 一般参考 中的[区域和终端节点](#)。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "statement1",  
            "Effect": "Allow",  
            "Action": [  
                "s3:CreateBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::*"  
            ],  
            "Condition": {  
                "StringLike": {  
                    "s3:LocationConstraint": "sa-east-1"  
                }  
            }  
        }  
    ]  
}
```

此策略限制用户只能在 sa-east-1 区域创建存储桶。但有可能别的策略授予了此用户在其他区域创建存储桶的权限。例如，如果用户属于一个组，该组可能被附加了一个策略，使得该组中的所有用户都有权在其他某个区域创建存储桶。要确保此用户不会获得在其他任何区域创建存储桶的权限，可在该策略中添加一个显式拒绝语句。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "statement1",  
            "Effect": "Allow",  
            "Action": [  
                "s3:CreateBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::*"  
            ],  
            "Condition": {  
                "StringLike": {  
                    "s3:LocationConstraint": "sa-east-1"  
                }  
            }  
        },  
        {  
            "Sid": "statement2",  
            "Effect": "Deny",  
            "Action": [  
                "s3:CreateBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::*"  
            ]  
        }  
    ]  
}
```

```
    "Resource": [
        "arn:aws:s3::::*"
    ],
    "Condition": {
        "StringNotLike": {
            "s3:LocationConstraint": "sa-east-1"
        }
    }
]
```

Deny 语句使用 StringNotLike 条件。即，如果位置约束不是 "sa-east-1"，则创建存储桶的请求将被拒绝。显式拒绝不允许用户在其他任何区域创建存储桶，无论该用户获得了哪种其他权限。

可使用以下 `create-bucket` AWS CLI 命令测试此策略。此示例使用 `bucketconfig.txt` 文件来指定位置约束。记下此 Windows 文件路径。您需要更新相应的存储桶名称和路径。必须使用 `--profile` 参数提供用户凭证。有关设置和使用 AWS CLI 的更多信息，请参阅 [设置用于示例演练的工具 \(p. 256\)](#)。

```
aws s3api create-bucket --bucket examplebucket --profile AccountADave --create-bucket-configuration file:///c:/Users/someUser/bucketconfig.txt
```

`bucketconfig.txt` 文件指定配置，如下所示：

```
{"LocationConstraint": "sa-east-1"}
```

### 示例 2：允许用户根据特定前缀获取存储桶中的对象的列表

存储桶拥有者可限定用户仅列出存储桶中特定文件夹的内容。如果存储桶中的对象按键名称前缀组织，这非常有用。Amazon S3 控制台可使用这些前缀显示文件夹的层级结构（仅控制台支持文件夹概念；Amazon S3 API 仅支持存储桶和对象）。

在该示例中，存储桶拥有者和用户所属的父账户相同。因此存储桶拥有者可使用存储桶策略或用户策略。首先，我们显示用户策略。

以下用户策略授予 `s3>ListBucket` 权限（请参阅 [GET Bucket \(列出对象\)](#)），条件为要求用户在请求中指定值为 `projects` 的 `prefix`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Action": [
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": "projects"
                }
            }
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Action": [

```

```
        "s3>ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::examplebucket"
    ],
    "Condition" : {
        "StringNotEquals" : {
            "s3:prefix": "projects"
        }
    }
}
]
```

此条件将用户限定于列出具有 `projects` 前缀的对象键。添加的显式拒绝将拒绝用户列出具有其他任何前缀的键，无论该用户可能具有其他什么权限。例如，该用户有可能获得列出没有任何限制的对象键的权限，例如通过更新先前用户策略或通过存储桶策略。但由于显式拒绝始终会取代其他任何权限，因此列出非 `project` 前缀的键的用户请求会被拒绝。

上述策略为用户策略。如果将 `Principal` 元素添加到该策略，从而标识用户，则现在您拥有了所示的存储桶策略。

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"statement1",
            "Effect":"Allow",
            "Principal": {
                "AWS": "arn:aws:iam::BucketOwner-accountID:user/user-name"
            },
            "Action":[
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket"
            ],
            "Condition" : {
                "StringEquals" : {
                    "s3:prefix": "examplefolder"
                }
            }
        },
        {
            "Sid":"statement2",
            "Effect":"Deny",
            "Principal": {
                "AWS": "arn:aws:iam::BucketOwner-AccountID:user/user-name"
            },
            "Action":[
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket"
            ],
            "Condition" : {
                "StringNotEquals" : {
                    "s3:prefix": "examplefolder"
                }
            }
        }
    ]
}
```

可使用以下 list-object AWS CLI 命令测试此策略。在该命令中，使用 --profile 参数提供用户凭证。有关设置和使用 AWS CLI 的更多信息，请参阅 [设置用于示例演练的工具 \(p. 256\)](#)。

```
aws s3api list-objects --bucket examplebucket --prefix examplefolder --profile AccountADave
```

现在，如果该存储桶启用了版本控制，要列出该存储桶中的对象，必须在上述策略中授予 s3>ListBucketVersions 权限，而不是 s3>ListBucket 权限。此权限还支持 s3:prefix 条件键。

## 存储桶策略示例

本节介绍几个关于存储桶策略的典型使用案例示例。策略在资源值中使用 `bucket` 和 `examplebucket` 字符串。要测试这些策略，您需要将这些字符串替换为您的存储桶名称。有关访问策略语言的更多信息，请参阅 [访问策略语言概述 \(p. 279\)](#)。

### Note

存储桶策略的大小限制为 20 KB。

您可以使用 [AWS 策略生成器](#) 为您的 Amazon S3 存储桶创建存储桶策略。然后，您可以使用 [Amazon S3 控制台](#)、众多第三方工具或通过您的应用程序，使用生成的文档设置您的存储桶策略。

### Important

使用 Amazon S3 控制台测试权限时，您需要授予控制台所需的其他权限 (s3>ListAllMyBuckets、s3:GetBucketLocation 和 s3>ListBucket 权限)。有关向用户授予权限并使用控制台测试这些权限的示例演练，请参阅 [演练示例：使用用户策略控制对存储桶的访问 \(p. 317\)](#)。

### 主题

- [在添加条件的情况下向多个账户授予权限 \(p. 304\)](#)
- [向匿名用户授予只读权限 \(p. 305\)](#)
- [限制对特定 IP 地址的访问权限 \(p. 305\)](#)
- [限制对特定 HTTP 引用站点的访问 \(p. 306\)](#)
- [向 Amazon CloudFront Origin Identity 授予权限 \(p. 307\)](#)
- [添加存储桶策略以请求 MFA \(p. 308\)](#)
- [在授予上传对象的交叉账户许可的同时，确保存储桶拥有者拥有完全控制 \(p. 309\)](#)
- [向 Amazon S3 清单和 Amazon S3 分析功能授予权限 \(p. 310\)](#)
- [Amazon S3 的 VPC 终端节点的示例存储桶策略 \(p. 310\)](#)

## 在添加条件的情况下向多个账户授予权限

以下示例策略向多个 AWS 账户授予 s3:PutObject 和 s3:PutObjectAcl 权限，并要求针对这些操作的任何请求都包含 public-read 标准 ACL。有关更多信息，请参阅 [在策略中指定权限 \(p. 282\)](#) 和 [在策略中指定条件 \(p. 286\)](#)。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AddCannedAcl",  
            "Effect": "Allow",  
            "Principal": {"AWS": ["arn:aws:iam::111122223333:root", "arn:aws:iam::444455556666:root"]},  
            "Action": ["s3:PutObject", "s3:PutObjectAcl"],  
            "Condition": {"StringLike": {"aws:SourceVpc": "10.0.0.0/16"}, "StringNotLike": {"aws:SourceVpc": "10.0.0.1/32"}},  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        }  
    ]  
}
```

```
        "Resource": ["arn:aws:s3:::examplebucket/*"],
        "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
    }
]
```

## 向匿名用户授予只读权限

下面的示例策略向任何公用匿名用户授予 s3:GetObject 权限。(有关权限以及它们允许执行的操作的列表，请参阅[在策略中指定权限 \(p. 282\)](#)。)此权限允许任何人读取对象数据，当您将存储桶配置为网站并且希望每个人都能读取存储桶中的对象时，这十分有用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPerm",
            "Effect": "Allow",
            "Principal": "*",
            "Action": ["s3:GetObject"],
            "Resource": ["arn:aws:s3:::examplebucket/*"]
        }
    ]
}
```

### Warning

在授予对您的 S3 存储桶的匿名访问权限时应谨慎使用。如果您授予匿名访问权限，那么世界上的任何人都可以访问您的存储桶。我们强烈建议您绝对不要授予对 S3 存储桶的任何类型的匿名写入权限。

## 限制对特定 IP 地址的访问权限

以下示例向任何用户授予对指定存储桶中的对象执行任何 Amazon S3 操作的权限。但是，请求必须来自条件中指定的 IP 地址范围。

此语句的条件确定允许的 Internet 协议版本 4 (IPv4) IP 地址范围为 54.240.143.\*，有一种例外情况：54.240.143.188。

Condition 块使用 IPAddress 和 NotIpAddress 条件以及 aws:SourceIp 条件键 (这是 AWS 范围的条件键)。有关这些条件键的更多信息，请参阅[在策略中指定条件 \(p. 286\)](#)。aws:sourceIp IPv4 值使用标准 CIDR 表示法。有关更多信息，请参阅 IAM 用户指南 中的 [IP 地址条件运算符](#)。

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "IpAddress": {"aws:SourceIp": "54.240.143.0/24"},
                "NotIpAddress": {"aws:SourceIp": "54.240.143.188/32"}
            }
        }
    ]
}
```

}

## 允许 IPv4 和 IPv6 地址

在您开始使用 IPv6 地址时，建议您使用您的 IPv6 地址范围以及现有的 IPv4 范围来更新组织的策略，以确保策略在您过渡到 IPv6 时继续有效。

以下示例存储桶策略说明如何结合使用 IPv4 和 IPv6 地址范围来覆盖组织的所有有效 IP 地址。该示例策略允许对示例 IP 地址 54.240.143.1 和 2001:DB8:1234:5678::1 的访问，拒绝对地址 54.240.143.129 和 2001:DB8:1234:5678:ABCD::1 的访问。

`aws:sourceIp` 的 IPv6 值必须采用标准的 CIDR 格式。对于 IPv6，我们支持使用 :: 表示 0s 范围，例如，`2032001:DB8:1234:5678::/64`。有关更多信息，请参阅 IAM 用户指南 中的 [IP 地址条件运算符](#)。

```
{  
    "Id": "PolicyId2",  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowIPmix",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::examplebucket/*",  
            "Condition": {  
                "IpAddress": {  
                    "aws:SourceIp": [  
                        "54.240.143.0/24",  
                        "2001:DB8:1234:5678::/64"  
                    ]  
                },  
                "NotIpAddress": {  
                    "aws:SourceIp": [  
                        "54.240.143.129/30",  
                        "2001:DB8:1234:5678:ABCD::/80"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

## 限制对特定 HTTP 引用站点的访问

假设您拥有一个网站，其域名为 (`www.example.com` 或 `example.com`)，并且带有指向存储在 S3 存储桶中的照片和视频的链接 `examplebucket`。默认情况下，所有 S3 资源都是私有的，因此只有创建资源的 AWS 账户才能访问它们。要允许从您的网站对这些对象进行读取访问，您可以添加一个存储桶策略允许 `s3:GetObject` 权限，并附带使用 `aws:referer` 键的条件，即获取请求必须来自特定的网页。以下策略指定带有 `aws:Referer` 条件键的 `StringLike` 条件。

```
{  
    "Version": "2012-10-17",  
    "Id": "http referer policy example",  
    "Statement": [  
        {  
            "Sid": "Allow get requests originating from www.example.com and example.com.",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "s3:GetObject",  
            "Resource": "arn:aws:s3:::examplebucket/*",  
            "Condition": {  
                "aws:Referer": "StringLike",  
                "aws:Referer": ["http://www.example.com", "http://example.com"]  
            }  
        }  
    ]  
}
```

```
        "Condition": {
            "StringLike": {"aws:Referer": ["http://www.example.com/*", "http://example.com/*"]}
        }
    ]
}
```

确保您使用的浏览器在请求中包含 http referer 标头。

您可以通过向存储桶策略添加显式拒绝，更好地保护对 examplebucket 存储桶中的对象的访问，如下面的示例所示。显式拒绝将取代您使用其他方法（如 ACL 或用户策略）可能已授予 examplebucket 存储桶中的对象的任何许可。

```
{
    "Version": "2012-10-17",
    "Id": "http referer policy example",
    "Statement": [
        {
            "Sid": "Allow get requests referred by www.example.com and example.com.",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringLike": {"aws:Referer": ["http://www.example.com/*", "http://example.com/*"]}
            }
        },
        {
            "Sid": "Explicit deny to ensure requests are allowed only from specific referer.",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringNotLike": {"aws:Referer": ["http://www.example.com/*", "http://example.com/*"]}
            }
        }
    ]
}
```

## 向 Amazon CloudFront Origin Identity 授予权限

下面的示例中，存储桶策略向 CloudFront Origin Identity 授予获取（列出）Amazon S3 存储桶中所有对象的权限。CloudFront Origin Identity 用于启用 CloudFront 的私有内容功能。该策略使用 CanonicalUser 前缀而不是 AWS 来指定规范用户 ID。要了解有关提供私有内容的 CloudFront 支持的更多信息，请参阅 Amazon CloudFront 开发人员指南 中的 [提供私有内容](#) 主题。您必须为您的 CloudFront 分配的原始访问标识指定规范用户 ID。有关查找规范用户 ID 的说明，请参阅[在策略中指定委托人 \(p. 281\)](#)。

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": " Grant a CloudFront Origin Identity access to support private content",
            "Effect": "Allow",
            "Principal": {"CanonicalUser": "CloudFront Origin Identity Canonical User ID"},
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::examplebucket/*"
        }
    ]
}
```

}

## 添加存储桶策略以请求 MFA

Amazon S3 支持受 MFA 保护的 API 访问，这是一项可在访问您的 Amazon S3 资源时强制进行多重验证 (MFA) 的功能。Multi-Factor Authentication 提供了额外的安全级别，可以应用于您的 AWS 环境。它是一项安全功能，要求用户通过提供有效 MFA 代码来证明实际拥有 MFA 设备。有关详细信息，请参阅 [AWS Multi-Factor Authentication](#)。您可以要求对任何访问 Amazon S3 资源的请求进行 MFA 身份验证。

您可以使用存储桶策略中的 `aws:MultiFactorAuthAge` 密钥强制 MFA 身份验证请求。IAM 用户可以使用由 AWS Security Token Service (STS) 颁发的临时凭证访问 Amazon S3 资源。您可以在 STS 请求时提供 MFA 代码。

Amazon S3 收到带 MFA 身份验证的请求时，`aws:MultiFactorAuthAge` 键将提供一个数值，指示临时凭证是在多久以前创建的 (以秒为单位)。如果请求中提供的临时凭证不是使用 MFA 设备创建的，则此键值为空 (缺失)。您可以在存储桶策略中添加一个条件来检查此值，如下面的示例存储桶策略所示。如果请求未经 MFA 身份验证，策略将拒绝对 `examplebucket` 存储桶中的 `/taxdocuments` 文件夹执行的任何 Amazon S3 操作。要了解有关 MFA 身份验证的更多信息，请参阅 IAM 用户指南 中的 [在 AWS 中使用 Multi-Factor Authentication \(MFA\)](#)。

```
{  
    "Version": "2012-10-17",  
    "Id": "123",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",  
            "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }  
        }  
    ]  
}
```

如果 `aws:MultiFactorAuthAge` 键值为空，即指示请求中创建的临时安全凭证不具有 MFA 密钥，则 Condition 块中的 Null 条件的计算结果为 `true`。

下面的存储桶策略是上述存储桶策略的扩展。它包含两个策略声明。一条语句允许所有人对存储桶 (`examplebucket`) 拥有 `s3:GetObject` 权限，另一条语句通过要求进行 MFA 身份验证进一步限制对存储桶中 `examplebucket/taxdocuments` 文件夹的访问。

```
{  
    "Version": "2012-10-17",  
    "Id": "123",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",  
            "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }  
        },  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": ["s3:GetObject"],  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        }  
    ]  
}
```

```
    ]  
}
```

您可以选择使用数值条件限制 aws:MultiFactorAuthAge 密钥的有效期，该期限独立于对请求进行身份验证时使用的临时安全凭证的生存期。例如，除了要求 MFA 身份验证外，下面的存储桶策略还会查看临时会话是在多久以前创建的。如果 aws:MultiFactorAuthAge 键值指示临时会话是在一个小时 (3600 秒) 之前创建的，则策略将拒绝任何操作。

```
{  
    "Version": "2012-10-17",  
    "Id": "123",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",  
            "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}  
        },  
        {  
            "Sid": "",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",  
            "Condition": {"NumericGreaterThanOrEqual": {"aws:MultiFactorAuthAge": 3600 }}  
        },  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": ["s3:GetObject"],  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        }  
    ]  
}
```

## 在授予上传对象的交叉账户许可的同时，确保存储桶拥有者拥有完全控制

您可以允许其他 AWS 账户将对象上传到您的存储桶。然而，您可能会决定作为存储桶拥有者，您必须完全控制已上传到您的存储桶的对象。下面的策略将强制拒绝特定的 AWS 账户 (111111111111) 上传对象的能力，除非该账户向由电子邮件地址 (xyz@amazon.com) 标识的存储桶拥有者授予完全控制访问权限。该策略中的 StringNotEquals 条件指定 s3:x-amz-grant-full-control 条件键以表示要求 (请参阅[在策略中指定条件 \(p. 286\)](#))。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "111",  
            "Effect": "Allow",  
            "Principal": {"AWS": "111111111111"},  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::examplebucket/*"  
        },  
        {  
            "Sid": "112",  
            "Effect": "Deny",  
            "Principal": {"AWS": "111111111111"},  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::examplebucket/*",  
            "Condition": {"StringNotEquals": {"s3:x-amz-grant-full-control": "xyz@amazon.com"} }  
        }  
    ]  
}
```

```
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::examplebucket/*",
    "Condition": {
        "StringNotEquals": {"s3:x-amz-grant-full-control": ["emailAddress=xyz@amazon.com"]}}
    }
}
```

## 向 Amazon S3 清单和 Amazon S3 分析功能授予权限

Amazon S3 清单功能在 S3 存储桶中创建对象列表，而 Amazon S3 分析导出功能创建分析中使用的数据的输出文件。由清单列出其对象的存储桶称为源存储桶。清单文件将写入到的存储桶和分析导出文件将写入到的存储桶称为目标存储桶。您在为 S3 存储桶设置清单以及设置分析导出时，必须为目标存储桶创建存储桶策略。有关更多信息，请参阅 [Amazon S3 清单 \(p. 235\)](#) 和 [Amazon S3 分析 - 存储类分析 \(p. 230\)](#)。

以下示例存储桶策略向 Amazon S3 授予将来自源存储桶账户的对象写入 (PUT) 到目标存储桶的权限。您在设置 Amazon S3 清单和 Amazon S3 分析导出功能时，将对目标存储桶使用像这样的存储桶策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InventoryAndAnalyticsExamplePolicy",
            "Effect": "Allow",
            "Principal": {"Service": "s3.amazonaws.com"},
            "Action": ["s3:PutObject"],
            "Resource": ["arn:aws:s3:::destination-bucket/*"],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "1234567890",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

## Amazon S3 的 VPC 终端节点的示例存储桶策略

您可以使用 Amazon S3 存储桶策略控制从特定 Amazon Virtual Private Cloud (Amazon VPC) 终端节点或特定 VPC 对存储桶的访问。本部分包含可用于从 VPC 终端节点控制 S3 存储桶访问的示例存储桶策略。要了解如何设置 VPC 终端节点，请参阅 Amazon VPC 用户指南中的 [VPC 终端节点](#)。

Amazon VPC 允许您在定义的虚拟网络内启动 Amazon Web Services (AWS) 资源。使用 VPC 终端节点可以在您的 VPC 和其他 AWS 服务之间创建私有连接，无需通过 Internet、VPN 连接、NAT 实例或 AWS Direct Connect 进行访问。

Amazon S3 的 VPC 终端节点是 VPC 内的一种逻辑实体，只允许连接到 Amazon S3。VPC 终端节点将请求路由到 Amazon S3 并将响应路由回 VPC。VPC 终端节点仅更改请求的路由方式。Amazon S3 公有终端节点和 DNS 名称将继续使用 VPC 终端节点。有关在 Amazon S3 中使用 Amazon VPC 终端节点的重要信息，请参阅 Amazon VPC 用户指南中的 [网关 VPC 终端节点](#) 和 [Amazon S3 的终端节点](#)。

Amazon S3 的 VPC 终端节点提供两种方式来控制对 Amazon S3 数据的访问：

- 您可以控制允许通过特定 VPC 终端节点访问的请求、用户或组。有关此类型的访问控制的信息，请参阅 Amazon VPC 用户指南中的 [使用 VPC 终端节点控制对服务的访问](#)。

- 可以使用 S3 存储桶策略控制哪些 VPC 或 VPC 终端节点有权访问 S3 存储桶。有关此类型存储桶策略访问控制的示例，请参阅以下有关限制访问的主题。

#### 主题

- [限制对特定 VPC 终端节点的访问 \(p. 311\)](#)
- [限制对特定 VPC 的访问 \(p. 311\)](#)
- [相关资源 \(p. 312\)](#)

#### Important

如本节中所述对 VPC 终端节点应用 S3 存储桶策略时，您可能会无意中阻止对存储桶的访问权限。存储桶权限旨在专门限制存储桶访问源自 VPC 终端节点的连接，而这可能会阻止到存储桶的所有连接。有关如何解决此问题的信息，请参阅 AWS Support 知识中心中的[在对 Amazon S3 存储桶应用策略以限制对 VPC 终端节点的访问后，如何重新获得对存储桶的访问权限？](#)

## 限制对特定 VPC 终端节点的访问

下面是限制仅从 ID 为 vpce-1a2b3c4d 的 VPC 终端节点访问特定存储桶 examplebucket 的 S3 存储桶策略示例。如果未使用指定的终端节点，则该策略拒绝对存储桶的所有访问。`aws:sourceVpce` 条件用于指定终端节点。`aws:sourceVpc` 条件不需要 VPC 终端节点资源的 ARN，而只需要 VPC 终端节点 ID。有关在策略中使用条件的更多信息，请参阅[在策略中指定条件 \(p. 286\)](#)。

```
{  
    "Version": "2012-10-17",  
    "Id": "Policy1415115909152",  
    "Statement": [  
        {  
            "Sid": "Access-to-specific-VPCE-only",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Effect": "Deny",  
            "Resource": ["arn:aws:s3::::examplebucket",  
                        "arn:aws:s3::::examplebucket/*"],  
            "Condition": {  
                "StringNotEquals": {  
                    "aws:sourceVpce": "vpce-1a2b3c4d"  
                }  
            }  
        }  
    ]  
}
```

## 限制对特定 VPC 的访问

可以使用 `aws:sourceVpc` 条件来创建用于限制对特定 VPC 的访问的存储桶策略。如果在同一 VPC 中配置了多个 VPC 终端节点，并且要管理对所有终端节点的 S3 存储桶的访问，这一点非常有用。下面是允许 VPC vpc-111bbb22 访问 examplebucket 及其对象的策略的示例。如果未使用指定的 VPC，则该策略拒绝对存储桶的所有访问。`vpc-111bbb22` 条件键不需要 VPC 资源的 ARN，仅需要 VPC ID。

```
{  
    "Version": "2012-10-17",  
    "Id": "Policy1415115909153",  
    "Statement": [  
        {  
            "Sid": "Access-to-specific-VPC-only",  
            "Principal": "*",  
            "Action": "s3:*
```

```
    "Effect": "Deny",
    "Resource": ["arn:aws:s3:::examplebucket",
                 "arn:aws:s3:::examplebucket/*"],
    "Condition": {
        "StringNotEquals": {
            "aws:sourceVpc": "vpc-111bbb22"
        }
    }
}
```

## 相关资源

- [存储桶策略示例 \(p. 304\)](#)
- [Amazon VPC 用户指南 中的 VPC 终端节点](#)
- [在对 Amazon S3 存储桶应用策略以限制对 VPC 终端节点的访问后，如何重新获得对存储桶的访问权限？](#)

## 用户策略示例

本节介绍用于控制用户访问 Amazon S3 的几个 IAM 用户策略。有关访问策略语言的更多信息，请参阅[访问策略语言概述 \(p. 279\)](#)。

如果您以编程方式测试以下示例策略，则它们将起作用；但是，为了使它们与 Amazon S3 控制台一起使用，您将需要授予控制台所需的其他权限。有关使用策略（例如与 Amazon S3 控制台一起使用的策略）的信息，请参阅[演练示例：使用用户策略控制对存储桶的访问 \(p. 317\)](#)。

### 主题

- [示例：允许 IAM 用户访问您的一个存储桶 \(p. 312\)](#)
- [示例：允许每个 IAM 用户访问存储桶中的一个文件夹 \(p. 313\)](#)
- [示例：允许组在 Amazon S3 中拥有共享文件夹 \(p. 315\)](#)
- [示例：允许所有用户读取企业存储桶某部分中的对象 \(p. 316\)](#)
- [示例：允许合作伙伴将文件放入企业存储桶的特定部分中 \(p. 316\)](#)
- [演练示例：使用用户策略控制对存储桶的访问 \(p. 317\)](#)

## 示例：允许 IAM 用户访问您的一个存储桶

在本示例中，您需要授予您的 AWS 账户中的 IAM 用户访问一个存储桶 examplebucket 的权限，以便该用户能够添加、更新和删除对象。

除了授予该用户 s3:PutObject、s3:GetObject 和 s3:DeleteObject 权限外，此策略还授予 s3>ListAllMyBuckets、s3:GetBucketLocation 和 s3>ListBucket 权限。这些是控制台所需的其他权限。此外，s3:PutObjectAcl 和 s3:GetObjectAcl 操作需要能够在控制台中复制、剪切和粘贴对象。有关向用户授予权限并使用控制台测试这些权限的示例演练，请参阅[演练示例：使用用户策略控制对存储桶的访问 \(p. 317\)](#)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListAllMyBuckets"
            ],
            ...
        }
    ]
}
```

```
        "Resource": "arn:aws:s3:::examplebucket/*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3>ListBucket",
            "s3:GetBucketLocation"
        ],
        "Resource": "arn:aws:s3:::examplebucket"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3>DeleteObject"
        ],
        "Resource": "arn:aws:s3:::examplebucket/*"
    }
}
```

## 示例：允许每个 IAM 用户访问存储桶中的一个文件夹

在本示例中，您需要两个 IAM 用户 (Alice 和 Bob) 具有访问存储桶 examplebucket 的权限，以便他们可以添加、更新和删除对象。但是，您想要限制每个用户对存储桶中单个文件夹的访问权限。您可以使用与用户名匹配的名称创建文件夹。

```
examplebucket
  Alice/
  Bob/
```

要授予每个用户仅可以访问他们的文件夹的权限，您可以为每个用户编写策略，然后分别附加它。例如，您可以将以下策略附加到用户 Alice，以允许她对 examplebucket/Alice 文件夹的特定 Amazon S3 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3>DeleteObject",
                "s3>DeleteObjectVersion"
            ],
            "Resource": "arn:aws:s3:::examplebucket/Alice/*"
        }
    ]
}
```

然后，您可以将类似策略附加到用户 Bob，以在 Resource 值中识别 Bob 文件夹。

不要将策略附加到单个用户，而是编写使用策略变量的单个策略并将该策略附加到组。首先，您需要创建一个组，然后将 Alice 和 Bob 添加到该组。以下示例策略允许在 examplebucket/\${aws:username} 文件夹中具有一组 Amazon S3 权限。评估策略后，策略变量 \${aws:username} 将替换为请求者的用户名。例如，如果 Alice 发送了一个请求以放置对象，只有当 Alice 将对象上传到 examplebucket/Alice 文件夹后，才允许该操作。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject",  
                "s3:GetObject",  
                "s3:GetObjectVersion",  
                "s3:DeleteObject",  
                "s3:DeleteObjectVersion"  
            ],  
            "Resource": "arn:aws:s3:::examplebucket/${aws:username}/*"  
        }  
    ]  
}
```

#### Note

当使用策略变量时，您必须在策略中明确指定版本 2012-10-17。访问策略语言的默认版本 2008-10-17 不支持策略变量。

如果需要在 Amazon S3 控制台上测试之前的策略，控制台需要其他 Amazon S3 权限，如以下策略所示。有关控制台如何使用这些权限的信息，请参阅 [演练示例：使用用户策略控制对存储桶的访问 \(p. 317\)](#)。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowGroupToSeeBucketListInTheConsole",  
            "Action": [ "s3>ListAllMyBuckets", "s3:GetBucketLocation" ],  
            "Effect": "Allow",  
            "Resource": [ "arn:aws:s3:::*" ]  
        },  
        {  
            "Sid": "AllowRootLevelListingOfTheBucket",  
            "Action": [ "s3>ListBucket" ],  
            "Effect": "Allow",  
            "Resource": [ "arn:aws:s3:::examplebucket" ],  
            "Condition": {  
                "StringEquals": {  
                    "s3:prefix": [ "" ], "s3:delimiter": [ "/" ]  
                }  
            }  
        },  
        {  
            "Sid": "AllowListBucketOfASpecificUserPrefix",  
            "Action": [ "s3>ListBucket" ],  
            "Effect": "Allow",  
            "Resource": [ "arn:aws:s3:::examplebucket" ],  
            "Condition": { "StringLike": { "s3:prefix": [ "${aws:username}/*" ] } }  
        },  
        {  
            "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject",  
                "s3:GetObject",  
                "s3:GetObjectVersion",  
                "s3:DeleteObject",  
                "s3:DeleteObjectVersion"  
            ],  
            "Resource": "arn:aws:s3:::examplebucket/${aws:username}/*"  
        }  
    ]  
}
```

```
        "Resource": "arn:aws:s3:::examplebucket/${aws:username}/*"
    ]
}
```

#### Note

在 2012-10-17 版本的策略中，策略变量以 \$ 开始。如果您的对象键包括 \$，则语法中的此更改可能会产生冲突。例如，要在策略中包括对象键 my\$file，可以使用 \$、\${\$} 指定 my\${\$}file 字符。

尽管 IAM 用户名称是友好、用户可读的标识符，但是它们无需全局唯一。例如，如果用户 Bob 离开了组织，且另一个 Bob 加入进来，则新 Bob 可以访问旧 Bob 的信息。您可以基于用户 ID 创建文件夹，而不是使用用户名。每个用户 ID 都是唯一的。在这种情况下，您必须修改之前的策略，以使用 \${aws:userid} 策略变量。有关用户标识符的更多信息，请参阅 IAM 用户指南 中的 [IAM 标识符](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
      ],
      "Resource": "arn:aws:s3:::my_corporate_bucket/home/${aws:userid}/*"
    }
  ]
}
```

## 允许非 IAM 用户 (移动应用用户) 访问存储桶中的文件夹

假设您要开发一个移动应用，一个将用户数据存储在 S3 存储桶中的游戏。对于每个应用用户，您都需要在您的存储桶中创建一个文件夹。您还需要限制每个用户对其文件夹的访问权限。但是，在有人下载您的应用、开始玩此游戏之前，您不能创建文件夹，因为您没有用户 ID。

这种情况下，您可以要求用户使用公共身份提供商 (如 Login with Amazon、Facebook、或 Google) 登录到您的应用程序。用户通过这些提供商之一登录应用后，他们就有了用户 ID，可用来在运行时创建用户特定文件夹。

然后，您可以使用 AWS Security Token Service 中的 Web 联合身份验证将来自身份提供商的信息与您的应用相集成，为每个用户获取临时安全凭证。然后，您可以创建 IAM 策略，以便允许该应用访问存储桶和执行操作，如创建用户特定文件夹和上传数据。有关 Web 联合身份的更多信息，请参阅 IAM 用户指南 中的 [关于 Web 联合身份](#)。

## 示例：允许组在 Amazon S3 中拥有共享文件夹

将以下策略附加到组时，会授予组中每个成员访问 Amazon S3 中以下文件夹的权限：my\_corporate\_bucket/share/marketing。组成员仅允许访问策略中显示的特定 Amazon S3 权限，仅适用于指定文件夹中的对象。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {

```

```
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
    ],
    "Resource": "arn:aws:s3:::my_corporate_bucket/share/marketing/*"
}
]
```

## 示例：允许所有用户读取企业存储桶某部分中的对象

在本示例中，您将创建名为 AllUsers 的组，该组包含 AWS 账户拥有的所有 IAM 用户。然后，您将附加向该组提供对 GetObject 和 GetObjectVersion 的访问权限的策略，但仅适用于 my\_corporate\_bucket/readonly 文件夹中的对象。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::my_corporate_bucket/readonly/*"
        }
    ]
}
```

## 示例：允许合作伙伴将文件放入企业存储桶的特定部分中

在本示例中，您将创建代表合作伙伴公司的名为 WidgetCo 的组。您将为需要访问权限的合作伙伴公司中的特定人员或应用程序创建 IAM 用户，然后将该用户放入组中。

然后，您将附加向组 PutObject 提供对企业存储桶中的以下文件夹的访问权限的策略：my\_corporate\_bucket/uploads/widgetco。

您需要阻止 WidgetCo 组对存储桶执行任何其他操作，因此添加了一条语句，除了对 AWS 账户中的任何 Amazon S3 资源的 PutObject 权限外，该语句将显式拒绝对所有 Amazon S3 权限的权限。如果在您的 AWS 账户的其他位置使用了广泛策略，该策略向用户提供对 Amazon S3 资源的广泛访问，则此步骤是必要的。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my_corporate_bucket/uploads/widgetco/*"
        },
        {
            "Effect": "Deny",
            "NotAction": "s3:PutObject",
            "Resource": "arn:aws:s3:::my_corporate_bucket/uploads/widgetco/*"
        }
    ]
}
```

```
        "Effect": "Deny",
        "Action": "s3:*",
        "NotResource": "arn:aws:s3:::my_corporate_bucket/uploads/widgetco/*"
    }
]
```

## 演练示例：使用用户策略控制对存储桶的访问

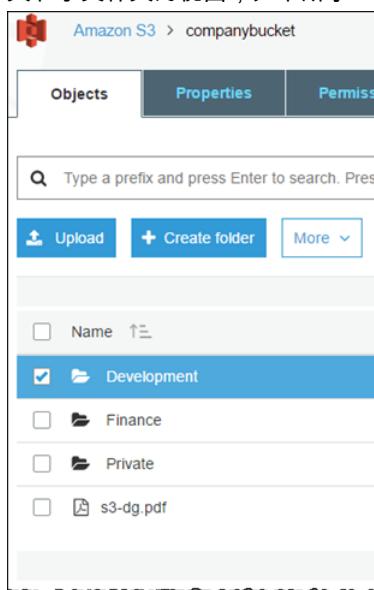
本演练说明如何将用户许可用于 Amazon S3。我们将创建一个包含文件夹的存储桶，然后在您的 AWS 账户中创建 AWS Identity and Access Management 用户并向这些用户授予对 Amazon S3 存储桶及其中的文件夹的增量许可。

### 主题

- [背景：存储桶和文件夹基础知识 \(p. 317\)](#)
- [演练示例 \(p. 319\)](#)
- [步骤 0：准备演练 \(p. 319\)](#)
- [步骤 1：创建存储桶 \(p. 320\)](#)
- [步骤 2：创建 IAM 用户和组 \(p. 320\)](#)
- [步骤 3：验证 IAM 用户没有任何权限 \(p. 321\)](#)
- [步骤 4：授予组级许可 \(p. 321\)](#)
- [步骤 5：授予 IAM 用户 Alice 特定的权限 \(p. 327\)](#)
- [步骤 6：授予 IAM 用户 Bob 特定的权限 \(p. 330\)](#)
- [步骤 7：确保 Private 文件夹的安全 \(p. 330\)](#)
- [清除 \(p. 332\)](#)
- [相关资源 \(p. 332\)](#)

### 背景：存储桶和文件夹基础知识

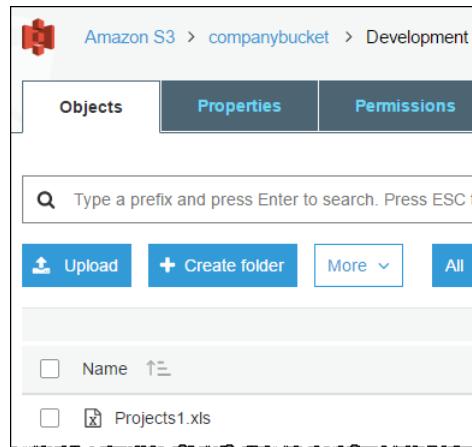
Amazon S3 数据模型是一种扁平结构：您创建存储桶，存储桶存储对象。不存在子存储桶或子文件夹的层级结构，但您可以效仿文件夹的层级结构。诸如 Amazon S3 控制台等工具可以显示存储桶中的这些逻辑文件夹和子文件夹的视图，如下所示：



控制台显示名为 companybucket 的存储桶拥有三个文件夹：Private、Development 和 Finance；以及一个对象：s3-dg.pdf。控制台使用对象名称（键）来创建文件夹和子文件夹的逻辑层级结构。考虑以下示例：

- 当您创建 Development 文件夹时，控制台将创建带有键值 Development/ 的对象。请注意尾部的“/”分隔符。
- 当您将名为 Projects1.xls 的对象上传到 Development 文件夹时，控制台会上传该对象并授予它 Development/Projects1.xls 键。

在此键值中，Development 是前缀，‘/’是分隔符。Amazon S3 API 在其操作中支持前缀和分隔符。例如，您可以使用特定的前缀和分隔符，从存储桶获取所有对象的列表。在控制台中，若您双击 Development 文件夹，控制台会列出该文件夹内的对象。在下例中，Development 文件夹包含一个对象。



当控制台列出 companybucket 存储桶中的 Development 文件夹时，它会向 Amazon S3 发送请求并在请求中指定前缀 Development 和分隔符 ‘/’。控制台的反应方式与计算机文件系统内的文件夹列表的反应方式相似。上一个示例显示，存储桶 companybucket 拥有一个带键值 Development/Projects1.xls 的对象。

控制台使用对象键来推断逻辑层级结构；Amazon S3 没有物理层级结构，它只有采用平面文件结构包含对象的存储桶。若您使用 Amazon S3 API 来创建对象，可以使用暗示逻辑层级结构的对象键。

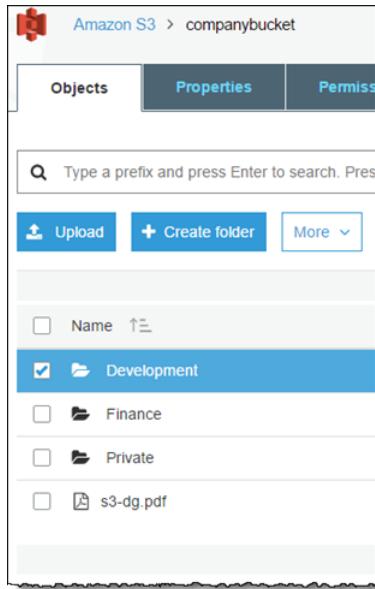
当您创建对象的逻辑层级结构时，您可以管理对单个文件夹的访问权限，在本演练中我们将进行这样的操作。

在浏览演练之前，您还需要熟悉另一个概念，即“根级”存储桶内容。假设您的 companybucket 存储桶具有以下对象：

Private/privDoc1.txt  
Private/privDoc2.zip  
Development/project1.xls  
Development/project2.xls  
Finance/Tax2011/document1.pdf  
Finance/Tax2011/document2.pdf  
s3-dg.pdf

这些对象键可以创建一个逻辑层级结构，将 Private、Development 和 Finance 作为根级文件夹并将 s3-dg.pdf 作为根级对象。当您在 Amazon S3 控制台中单击存储桶名称时，根级项目如下所示。控制台会

将顶级前缀 (Private/、Development/ 和 Finance/) 显示为根级文件夹。对象键 s3-dg.pdf 没有前缀，因此它显示为根级项目。



## 演练示例

本演练的示例如下：

- 创建一个存储桶，然后向其添加三个文件夹 (Private、Development 和 Finance)。
- 您拥有两个用户，Alice 和 Bob。您希望 Alice 只能访问 Development 文件夹；Bob 只能访问 Finance 文件夹；并且您想要将 Private 文件夹内容保持为私有。在演练中，您通过创建 AWS Identity and Access Management (IAM) 用户 (我们将使用相同用户名 Alice 和 Bob) 来管理访问，并向这些用户授予所需许可。

IAM 还支持创建用户组以及授予适用于组中所有用户的组级许可。这将帮助您更好地管理许可。就本演练而言，Alice 和 Bob 都需要一些常规的许可。因此，您也需要创建名为 Consultants 的组，并将 Alice 和 Bob 添加到该组。首先，通过将组策略附加到该组来授予许可。然后，通过将策略附加到特定用户来添加用户特定的许可。

### Note

本演练使用 companybucket 作为存储桶名称，使用 Alice 和 Bob 作为 IAM 用户，并使用 Consultants 作为组名称。由于 Amazon S3 要求存储桶名称全局唯一，因此您需要创建一个名称来替代存储桶名称。

## 步骤 0：准备演练

在本示例中，您将使用 AWS 账户凭证来创建 IAM 用户。最初，这些用户没有权限。您可以采用递增的方式授予这些用户许可，让他们执行特定的 Amazon S3 操作。要测试这些许可，您需要使用每个用户的凭证登录控制台。由于您以 AWS 账户拥有者身份递增地授予许可，并以 IAM 用户身份测试这些许可，因此每次使用不同的凭证时需要登录和注销。您可以使用一个浏览器进行此项测试，但如果采用两个不同的浏览器，测试过程会更快：使用 AWS 账户凭证将一个浏览器连接到 AWS 管理控制台，并使用 IAM 用户凭证连接另一个浏览器。

要使用 AWS 账户凭证登录 AWS 管理控制台，请转到 <https://console.aws.amazon.com/>。IAM 用户无法使用同一链接进行登录。IAM 用户必须使用支持 IAM 的登录页面。作为账户所有者，您可以向您的用户提供此链接。

有关 IAM 的更多信息，请转到 IAM 用户指南 中的 [AWS 管理控制台登录页](#)。

### 向 IAM 用户提供登录链接

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。
2. 在 Navigation 窗格中，单击 IAM Dashboard。
3. 记下 IAM users sign in link: 下的 URL。向 IAM 用户提供此链接，以便其使用 IAM 用户名称和密码登录控制台。

### 步骤 1：创建存储桶

在此步骤中，您可以使用 AWS 账户凭证登录到 Amazon S3 控制台，创建一个存储桶，将文件夹 (Development、Finance 和 Private) 添加到该存储桶，然后在每个文件夹中上传一个或两个示例文档。

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 创建存储桶。

有关分步说明，请参阅[如何创建 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南中)。

3. 将一个文档上传到存储桶。

本演练假设您有一个 s3-dg.pdf 文档位于此存储桶的根级。如果您上传其他文档，请使用其文件名替代 s3-dg.pdf。

4. 将名为 Private、Finance 和 Development 的三个文件夹添加到存储桶。
5. 将一个或两个文档上传到每个文件夹。

在本演练中，我们将假设您已在每个文件夹中上传了一些文档，并使存储桶包含带有以下密钥的对象：

Private/privDoc1.txt

Private/privDoc2.zip

Development/project1.xls

Development/project2.xls

Finance/Tax2011/document1.pdf

Finance/Tax2011/document2.pdf

s3-dg.pdf

有关分步说明，请参阅[如何将文件和文件夹上传至 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南中)。

### 步骤 2：创建 IAM 用户和组

现在使用 IAM 控制台将两个 IAM 用户 (Alice 和 Bob) 添加到您的 AWS 账户。还需要创建名为 Consultants 的管理组，然后将这两个用户添加到该组。

#### Warning

添加用户和组时，请不要为这些用户附加任何授予许可的策略。一开始，这些用户没有任何许可。在下面的部分中，您会采用递增的方式授予许可。您首先必须确保已为这些 IAM 用户分配密码。您将使用这些用户凭证来测试 Amazon S3 操作并验证这些许可按预期工作。

有关创建新 IAM 用户的分步说明，请参阅 IAM 用户指南 中的[在您的 AWS 账户中创建 IAM 用户](#)。为此演练创建用户之后，选中“AWS 管理控制台访问”并将“编程访问”保留为未选中。

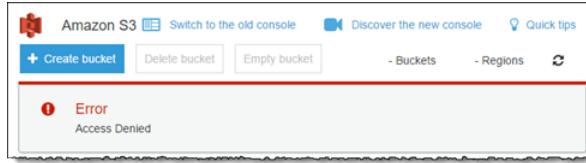
有关创建管理组的分步指导，请参阅 IAM 用户指南 中的[创建您的第一个 IAM 用户和管理员组](#)。

## 步骤 3：验证 IAM 用户没有任何权限

如果您使用两个浏览器，现在可以使用第二个浏览器通过 IAM 用户凭证之一登录控制台。

1. 使用 IAM 用户登录链接 (请参阅[向 IAM 用户提供登录链接 \(p. 320\)](#))，通过任一 IAM 用户证书登录 AWS 控制台。
2. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

验证是否有以下告知您访问已被拒绝的控制台消息。



现在开始授予用户增量许可。首先，附加组策略以授予两个用户都必须拥有的许可。

## 步骤 4：授予组级许可

我们希望所有的用户都可以执行以下操作：

- 列出父账户拥有的所有存储桶。要执行此操作，Bob 和 Alice 必须拥有执行 s3>ListAllMyBuckets 操作的许可。
- 在 companybucket 存储桶中，列出根级项目、文件夹和对象。要执行此操作，Bob 和 Alice 必须拥有在 companybucket 存储桶上执行 s3>ListBucket 操作的许可。

现在，我们将创建可授予这些许可的策略，然后将其附加到 Consultants 组。

### 步骤 4.1：授予列出所有存储桶的权限

在此步骤中，您将创建一个托管策略，该策略授予用户最低权限，使他们可以列出父账户拥有的所有存储桶，然后您将此策略附加到 Consultants 组。如果您将该托管策略附加到某个用户或组，您即授予该用户或组许可，允许他们获取父 AWS 账户拥有的存储桶的列表。

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。

#### Note

因为您将授予用户权限，所以应使用您的 AWS 账户证书而不是以 IAM 用户身份登录。

2. 创建托管策略。
  - a. 在左侧的导航窗格中，单击 Policies (策略)，然后单击 Create Policy (创建策略)。
  - b. 单击 JSON 选项卡。
  - c. 复制下面的访问策略，然后将其粘贴到策略文本字段中：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowGroupToSeeBucketListInTheConsole",  
            "Effect": "Allow",  
            "Action": "s3:ListBucket",  
            "Resource": "arn:aws:s3:::companybucket"  
        }  
    ]  
}
```

```
        "Action": ["s3>ListAllMyBuckets"],
        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::*"]
    }
]
```

策略是一个 JSON 文档。在文档中，Statement 是一个对象数组，其中每个对象均使用名称/值对的集合来描述许可。之前的策略描述了一个特定的许可。Action 指定访问权限的类型。在策略中，s3>ListAllMyBuckets 是预定义的 Amazon S3 操作。此操作涵盖了 Amazon S3 GET Service 操作，它可以返回已经过身份验证的发件人拥有的所有存储桶的列表。Effect 元素值决定是允许还是拒绝特定的许可。

- d. 单击 Review Policy (查看策略)。在下一页上，在 Name (名称) 字段中输入 AllowGroupToSeeBucketListInTheConsole，然后单击 Create policy (创建策略)。

#### Note

Summary (摘要) 条目将显示一条消息，说明策略未授予任何权限。对于本演练，您可以安全地忽略此消息。

3. 将您创建的 AllowGroupToSeeBucketListInTheConsole 托管策略附加到 Consultants 组。

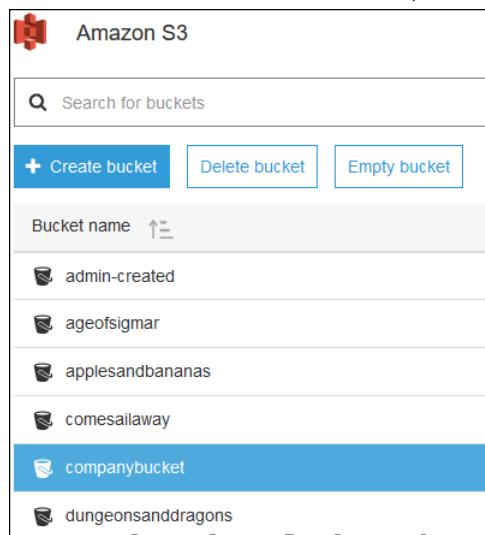
有关附加托管策略的分步说明，请参阅 IAM 用户指南 中的[添加和删除 IAM 策略 \( 控制台 \)](#)。

您可以在 IAM 控制台中将策略文档附加到 IAM 用户和组。因为我们希望两个用户都能够列出存储桶，所以我们将该策略附加到该组。

4. 测试许可。

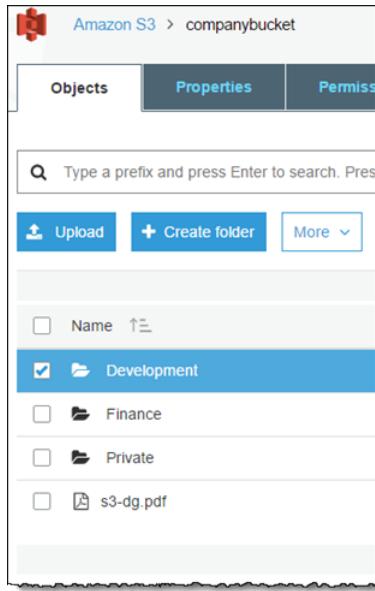
- a. 使用 IAM 用户登录链接 (请参阅[向 IAM 用户提供登录链接 \(p. 320\)](#))，通过任何 IAM 用户证书登录 AWS 控制台。
- b. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

控制台现在应该会列出所有的存储桶，但不会列出任何存储桶中的对象。



#### 步骤 4.2：允许用户列出存储桶的根级内容

现在，我们允许 Consultants 组中的所有用户列出根级 companybucket 存储桶项目。当用户在 Amazon S3 控制台中单击公司存储桶时，该用户将能够看到此存储桶中的根级项目。



请记住，我们在使用 companybucket 进行说明。您必须使用您为此演练创建的存储桶的名称。

要了解在您单击存储桶名称时控制台向 Amazon S3 发送的请求、Amazon S3 返回的响应以及控制台如何解释响应，我们有必要进行深入的了解。

当您单击存储桶名称时，控制台会向 Amazon S3 发送 [GET Bucket \(列出对象\)](#) 请求。该请求包括以下参数：

- `prefix` 参数，使用空字符串作为其值。
- `delimiter` 参数，使用 `/` 作为其值。

以下是一个请求示例：

```
GET ?prefix=&delimiter=/ HTTP/1.1
Host: companybucket.s3.amazonaws.com
Date: Wed, 01 Aug 2012 12:00:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

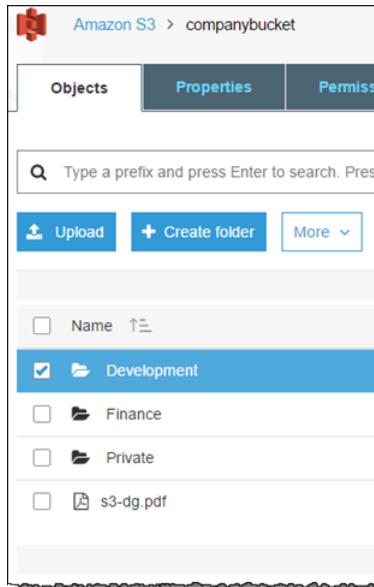
Amazon S3 将返回包含以下 `<ListBucketResult>` 元素的响应：

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Name>companybucket</Name>
<Prefix></Prefix>
<Delimiter>/<Delimiter>
...
<Contents>
<Key>s3-dg.pdf</Key>
...
</Contents>
<CommonPrefixes>
<Prefix>Development/<Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Finance/<Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Private/<Prefix>
</CommonPrefixes>
```

```
</ListBucketResult>
```

键值 s3-dg.pdf 不包含 '/' 分隔符，并且 Amazon S3 在 <Contents> 元素中返回键值。但是，我们的示例存储桶中的所有其他键值都包含 '/' 分隔符。Amazon S3 将组合这些键值，并在指定的 <CommonPrefixes> 分隔符首次出现时，为每个不同前缀值 Development/、Finance/ 和 Private/（它们是这些键值开头的子字符串）返回 '/' 元素。

控制台将解释该结果并将根级项目显示为三个文件夹和一个对象键。



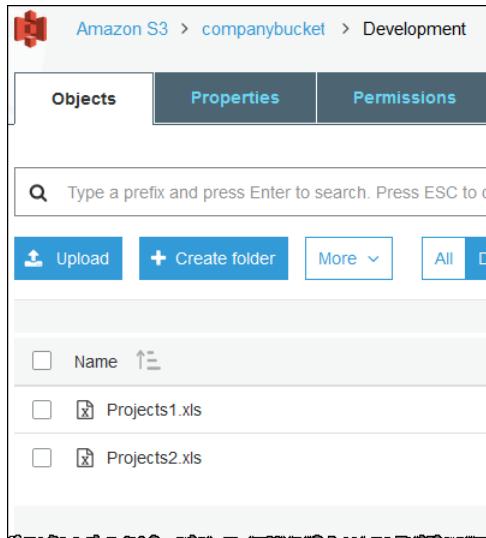
现在，如果 Bob 或 Alice 双击 Development 文件夹，控制台会向 Amazon S3 发送 GET Bucket (列出对象) 请求并将 prefix 和 delimiter 参数设置为以下值：

- prefix 参数，使用值 Development/。
- delimiter 参数，使用值 '/'。

作为响应，Amazon S3 将返回对象键，该键以指定的前缀开头。

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Name>companybucket</Name>
<Prefix>Development</Prefix>
<Delimiter>/<Delimiter>
...
<Contents>
<Key>Project1.xls</Key>
...
</Contents>
<Contents>
<Key>Project2.xls</Key>
...
</Contents>
</ListBucketResult>
```

控制台将显示以下对象键：



现在，我们返回到授予用户许可，以便列出根级存储桶项目。要列出存储桶内容，用户需要调用 `s3>ListBucket` 操作的许可，如下面的策略声明中所述。为确保他们只能看见根级内容，我们将添加一个条件：用户必须在请求中指定一个空的 `prefix` – 即不允许他们双击任意根级文件夹。最后，我们将通过要求用户请求包含使用值 ‘/’ 的 `delimiter` 参数，添加一个条件来要求文件夹样式的访问。

```
{  
    "Sid": "AllowRootLevelListingOfCompanyBucket",  
    "Action": ["s3>ListBucket"],  
    "Effect": "Allow",  
    "Resource": ["arn:aws:s3:::companybucket"],  
    "Condition": {  
        "StringEquals": {  
            "s3:prefix": [""], "s3:delimiter": ["/"]  
        }  
    }  
}
```

使用 Amazon S3 控制台时，请注意当您单击一个存储桶时，控制台首先会发送 `GET Bucket location` 请求，以查找部署存储桶的 AWS 区域。然后，控制台会将该区域特定的终端节点用于存储桶，以便发送 `GET Bucket (List Objects)` 请求。因此，如果用户要使用控制台，您必须授予他们进行 `s3:GetBucketLocation` 操作的许可，如下面的策略声明所述：

```
{  
    "Sid": "RequiredByS3Console",  
    "Action": ["s3:GetBucketLocation"],  
    "Effect": "Allow",  
    "Resource": ["arn:aws:s3::*"]  
}
```

#### 允许用户列出根级存储桶内容的步骤

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

使用 AWS 账户证书，而不是 IAM 用户的证书登录控制台。

2. 将附加到 Consultants 组的现有 `AllowGroupToSeeBucketListInTheConsole` 托管策略替换为以下策略，该策略还允许 `s3>ListBucket` 操作。记得将策略 Resource 中的 `companybucket` 替换为您存储桶的名称。

有关分步说明，请参阅 IAM 用户指南 中的[编辑客户托管策略](#)。按照分步说明操作时，请确保按照指示操作，以便将您的更改应用到策略附加到的所有委托人实体。

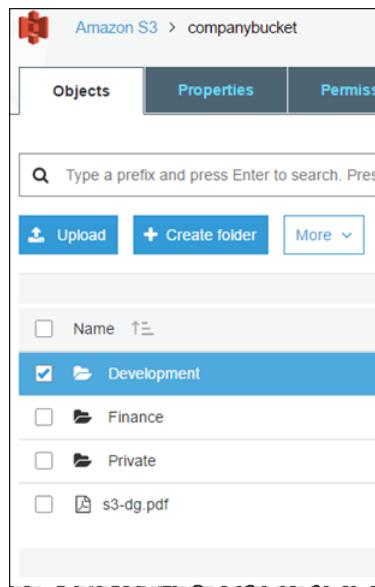
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",  
            "Action": [ "s3>ListAllMyBuckets", "s3:GetBucketLocation" ],  
            "Effect": "Allow",  
            "Resource": [ "arn:aws:s3:::*" ]  
        },  
        {  
            "Sid": "AllowRootLevelListingOfCompanyBucket",  
            "Action": [ "s3>ListBucket" ],  
            "Effect": "Allow",  
            "Resource": [ "arn:aws:s3:::companybucket" ],  
            "Condition":{  
                "StringEquals":{  
                    "s3:prefix":[""], "s3:delimiter":["/"]  
                }  
            }  
        }  
    ]  
}
```

3. 测试更新的许可。

- a. 使用 IAM 用户登录链接 (请参阅[向 IAM 用户提供登录链接 \(p. 320\)](#)) 登录 AWS 管理控制台。

通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

- b. 单击您为本演练创建的存储桶，控制台现在将显示根级存储桶项目。若您单击存储桶中的任意文件夹，您将无法查看文件夹内容，因为您尚未被授予执行这些操作的许可。



在用户使用 Amazon S3 控制台时，此项测试即成功完成，因为当您在控制台中单击存储桶时，控制台实施将发送一份请求，其中包含使用空字符串作为其值的 `prefix` 以及使用 '/' 作为其值的 `delimiter` 参数。

## 步骤 4.3：组策略总结

您所添加的组策略的实际结果是授予 IAM 用户 Alice 和 Bob 以下最低许可：

- 列出父账户拥有的所有存储桶。
- 查看 companybucket 存储桶中的根级项目。

然而，许多操作用户仍然无法执行。让我们授予用户特定的许可，如下所述：

- 允许 Alice 在 Development 文件夹中获取和放置对象。
- 允许 Bob 在 Finance 文件夹中获取和放置对象。

对于用户特定的许可，您需要将策略附加到特定的用户，而不是附加到组。在以下部分中，您将授予 Alice 在 Development 文件夹中工作的许可。您可以重复这些步骤，以便授予 Bob 相似的许可，使其可在 Finance 文件夹中工作。

## 步骤 5：授予 IAM 用户 Alice 特定的权限

现在，我们将其他许可授予 Alice，以使她能够查看 Development 文件夹中的内容，并可以在该文件夹中获取和放置对象。

### 步骤 5.1：授予 IAM 用户 Alice 列出 Development 文件夹内容的权限

要允许 Alice 列出 Development 文件夹内容，您必须对 Alice 应用一个策略，以授予她对 companybucket 存储桶执行 s3>ListBucket 操作的权限，但请求应包含前缀 Development/。因为我们希望只对用户 Alice 应用此策略，所以我们将使用内联策略。有关内联策略的更多信息，请参阅 IAM 用户指南 中的 [托管策略与内联策略](#)。

1. 登录 AWS 管理控制台并通过 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。

使用 AWS 账户证书，而不是 IAM 用户的证书登录控制台。

2. 创建内联策略以授予用户 Alice 列出 Development 文件夹内容的权限。
  - a. 在左侧导航窗格中，单击 Users。
  - b. 单击用户名 Alice。
  - c. 在用户详细信息页面上，选择 Permissions (权限) 选项卡，然后单击 Add inline policy (添加内联策略) 部分。
  - d. 单击 JSON 选项卡。
  - e. 将以下策略复制到策略文本字段中：

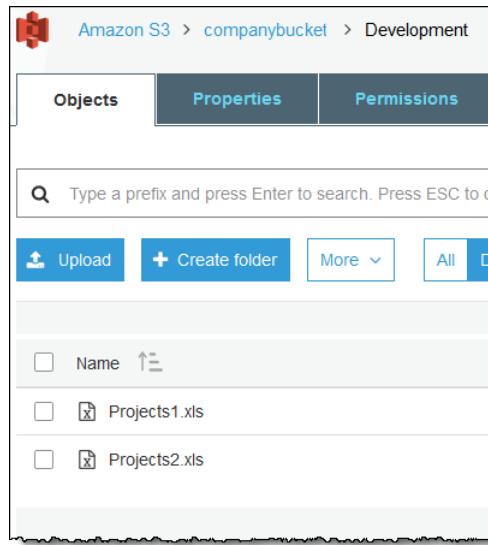
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",  
            "Action": ["s3>ListBucket"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::companybucket"],  
            "Condition": { "StringLike": {"s3:prefix": ["Development/*"] } }  
        }  
    ]  
}
```

- f. 单击 Review Policy (查看策略)。在下一页上，在 Name (名称) 字段中输入名称，然后单击 Create policy (创建策略)。

3. 测试对 Alice 的许可的更改：

- 使用 IAM 用户登录链接 (请参阅 [向 IAM 用户提供登录链接 \(p. 320\)](#)) 登录 AWS 管理控制台。
- 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
- 在 Amazon S3 控制台中，验证 Alice 能否可以查看存储桶中 Development/ 文件夹中的对象列表。

当用户单击 /Development 文件夹以查看其中的对象列表时，Amazon S3 控制台会向 Amazon S3 发送前缀为 /Development 的 ListObjects 请求。由于用户已被授予许可，可以查看使用前缀 Development 和分隔符 '/' 的对象列表，因此 Amazon S3 将返回使用键前缀 Development/ 的对象列表，并且控制台将显示该列表。



[步骤 5.2：授予 IAM 用户 Alice 在 Development 文件夹中获取和放置对象的权限](#)

要使 Alice 能够在 Development 文件夹中获取和放置对象，她需要用于调用 s3:GetObject 和 s3:PutObject 操作的许可。如果请求包含使用值 Development/ 的 prefix 参数，可使用下面的策略声明授予这些许可。

```
{  
    "Sid": "AllowUserToReadWriteObjectData",  
    "Action": ["s3:GetObject", "s3:PutObject"],  
    "Effect": "Allow",  
    "Resource": ["arn:aws:s3:::companybucket/Development/*"]  
}
```

- 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

使用 AWS 账户证书，而不是 IAM 用户的证书登录控制台。

- 编辑您在上一步中创建的内联策略。

- 在左侧导航窗格中，单击 Users。
- 单击用户名 Alice。
- 在用户详细信息页面上，选择 Permissions 选项卡，然后展开 Inline Policies 部分。
- 单击上一步所建策略的名称旁的 Edit Policy。
- 将下面的策略复制到策略文本字段中，替换现有策略：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",  
            "Action": ["s3>ListBucket"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::companybucket"],  
            "Condition": {  
                "StringLike": {"s3:prefix": ["Development/*"]} }  
        },  
        {  
            "Sid": "AllowUserToReadWriteObjectDataInDevelopmentFolder",  
            "Action": ["s3GetObject", "s3PutObject"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::companybucket/Development/*"]  
        }  
    ]  
}
```

3. 测试更新的策略：

- 使用 IAM 用户登录链接 (请参阅 [向 IAM 用户提供登录链接 \(p. 320\)](#)) 登录 AWS 管理控制台。
- 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
- 在 Amazon S3 控制台中，验证 Alice 现在是否可在 Development 文件夹中添加和下载对象。

### 步骤 5.3：显式拒绝 IAM 用户 Alice 对存储桶中其他任何文件夹的权限

用户 Alice 现在可以列出 companybucket 存储桶中的根级内容。她还可以在 Development 文件夹中获取和放置对象。如果您确实想要严格控制访问许可，可以显式拒绝 Alice 对存储桶中其他任何文件夹的访问。若存在任何其他策略 (存储桶策略或 ACL)，而且它们授予了 Alice 访问存储桶中任何其他文件夹的权限，此显式拒绝将覆盖这些许可。

您可以向用户 Alice 策略添加以下声明：要求 Alice 发送到 Amazon S3 的所有请求均包含 prefix 参数，该参数的值可为 Development/\* 或空字符串。

```
{  
    "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",  
    "Action": ["s3>ListBucket"],  
    "Effect": "Deny",  
    "Resource": ["arn:aws:s3:::companybucket"],  
    "Condition": { "StringNotLike": {"s3:prefix": ["Development/*", ""]},  
                  "Null" : {"s3:prefix": false} }  
}
```

请注意，Condition 数据块中存在两个条件表达式。使用逻辑 AND 组合这些条件表达式的结果。如果两种条件均为真，组合条件的结果才为真。由于此策略中的 Effect 为 Deny，在 Condition 求值为 true 时，用户将无法执行指定的 Action。

- Null 条件表达式确保来自 Alice 的请求包含 prefix 参数。

prefix 参数要求类似文件夹的访问权限。如果您发送的请求没有包含 prefix 参数，Amazon S3 将返回所有的对象键。

如果请求包括值为空的 prefix 参数，该表达式的值将计算为真，因此整个 Condition 的值也将计算为真。您必须允许将空字符串作为 prefix 参数的值。在前面讨论的内容中，回顾了采用与前面讨论中控制

台相同的方式，通过允许空字符串使 Alice 能够检索根级存储桶项目。有关更多信息，请参阅 [步骤 4.2：允许用户列出存储桶的根级内容 \(p. 322\)](#)。

- StringNotLike 条件表达式确保当指定了 prefix 参数的值并且不是 Development/\* 时，请求将失败。

根据上一节中的步骤，再次更新您为用户 Alice 创建的内联策略。

将下面的策略复制到策略文本字段中，替换现有策略：

```
{  
    "Statement": [  
        {  
            "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",  
            "Action": ["s3>ListBucket"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::companybucket"],  
            "Condition": {  
                "StringLike": {"s3:prefix": ["Development/*"]}  
            }  
        },  
        {  
            "Sid": "AllowUserToReadWriteObjectDataInDevelopmentFolder",  
            "Action": ["s3>GetObject", "s3>PutObject"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::companybucket/Development/*"]  
        },  
        {  
            "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",  
            "Action": ["s3>ListBucket"],  
            "Effect": "Deny",  
            "Resource": ["arn:aws:s3:::companybucket"],  
            "Condition": {  
                "StringNotLike": {"s3:prefix": ["Development/*", ""]},  
                "Null": {"s3:prefix": false}  
            }  
        }  
    ]  
}
```

## 步骤 6：授予 IAM 用户 Bob 特定的权限

现在，您想要授予 Bob 访问 Finance 文件夹的许可。遵循您之前在授予 Alice 许可时所采用的步骤，但是将 Development 文件夹替换成 Finance 文件夹。如需分步指导，请参阅 [步骤 5：授予 IAM 用户 Alice 特定的权限 \(p. 327\)](#)。

## 步骤 7：确保 Private 文件夹的安全

在此示例中，您有两个用户。您已授予了所需最低组级许可，仅当确实需要单个用户级别的许可时才授予用户级许可。此方法有助于最大程度地降低管理许可的工作量。随着用户数的增加，管理许可工作将变得极为繁重。例如，在本示例中您不希望任意用户访问 Private 文件夹内的内容。如何确保您不会在无意中授予用户访问该文件夹的许可？您可以添加一个策略，显式拒绝对该文件夹的访问。显式拒绝将覆盖任何其他许可。要确保 Private 文件夹仍保持为私有，您可以向组策略添加以下两个拒绝声明：

- 添加以下声明以显式拒绝对 Private 文件夹 (companybucket/Private/\*) 中的资源执行任何操作。

```
{  
    "Sid": "ExplicitDenyAccessToPrivateFolderToEveryoneInTheGroup",  
    "Action": ["s3:*"],  
    "Effect": "Deny",  
    "Resource": ["arn:aws:s3:::companybucket/Private/*"]
```

}

- 若请求指定了 Private/ 前缀，您也可以拒绝列出对象操作所需的许可。在控制台中，如果 Bob 或 Alice 双击 Private 文件夹，此策略将导致 Amazon S3 返回错误响应。

```
{  
    "Sid": "DenyListBucketOnPrivateFolder",  
    "Action": ["s3>ListBucket"],  
    "Effect": "Deny",  
    "Resource": ["arn:aws:s3:::*"],  
    "Condition":{  
        "StringLike": {"s3:prefix": ["Private/*"]}  
    }  
}
```

将 Consultants 组策略替换为包含了上述拒绝声明的更新策略。应用更新的策略后，组中的用户均无法访问存储桶中的 Private 文件夹。

- 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

使用 AWS 账户证书，而不是 IAM 用户的证书登录控制台。

- 将附加到 Consultants 组的现有 AllowGroupToSeeBucketListInTheConsole 托管策略替换为下面的策略。记得将策略中的 companybucket 替换为您的存储桶的名称。

有关说明，请参阅 IAM 用户指南 中的[编辑客户托管策略](#)。按照说明操作时，请确保按照指示操作，以便将您的更改应用到策略附加到的所有委托人实体。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",  
            "Action": ["s3>ListAllMyBuckets", "s3:GetBucketLocation"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::*"]  
        },  
        {  
            "Sid": "AllowRootLevelListingOfCompanyBucket",  
            "Action": ["s3>ListBucket"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::companybucket"],  
            "Condition":{  
                "StringEquals": {"s3:prefix": [""]}  
            }  
        },  
        {  
            "Sid": "RequireFolderPathStyleList",  
            "Action": ["s3>ListBucket"],  
            "Effect": "Deny",  
            "Resource": ["arn:aws:s3:::*"],  
            "Condition":{  
                "StringNotEquals": {"s3:delimiter": "/"}  
            }  
        },  
        {  
            "Sid": "ExplicitDenyAccessToPrivateFolderToEveryoneInTheGroup",  
            "Action": ["s3:*"],  
            "Effect": "Deny",  
            "Resource": ["arn:aws:s3:::companybucket/Private/*"]  
        }  
    ]  
}
```

```
},
{
    "Sid": "DenyListBucketOnPrivateFolder",
    "Action": ["s3>ListBucket"],
    "Effect": "Deny",
    "Resource": ["arn:aws:s3::::*"],
    "Condition": {
        "StringLike": {"s3:prefix": ["Private/*"]}
    }
}
]
```

## 清除

要进行清除，请转到 IAM 控制台并删除用户 Alice 和 Bob。有关分步说明，请转到 IAM 用户指南 中的[删除 IAM 用户](#)。

为了确保不再继续占用存储空间，您还应该删除您为本演练创建的对象和存储桶。

## 相关资源

- IAM 用户指南 中的[使用策略](#)。

# 使用 ACL 管理访问

## 主题

- [访问控制列表 \(ACL\) 概述 \(p. 333\)](#)
- [管理 ACL \(p. 338\)](#)

访问控制列表 (ACL) 是基于资源的访问策略选项之一 (请参阅[访问管理概述 \(p. 243\)](#))，可用于管理对存储桶和对象的访问。使用 ACL 可向其他 AWS 账户授予基本的读/写权限。使用 ACL 管理权限有一些限制。例如，您可以仅向其他 AWS 账户授予权限；您无法向您账户中的用户授予权限。无法授予条件性权限，也无法显式拒绝权限。ACL 适用于特定情景。例如，如果存储桶拥有者允许其他 AWS 账户上传对象，则对这些对象的权限只能由拥有此对象的 AWS 账户使用对象 ACL 加以管理。

以下介绍性主题说明了可供您管理对 Amazon S3 资源的访问权限的基本概念和选项，并提供有关何时使用哪些访问策略选项的指南。

- [Amazon S3 资源访问权限管理介绍 \(p. 242\)](#)
- [有关使用可用访问策略选项的准则 \(p. 252\)](#)

## 访问控制列表 (ACL) 概述

## 主题

- [谁是被授权者？ \(p. 334\)](#)
- [我能授予哪些许可？ \(p. 335\)](#)
- [示例 ACL \(p. 336\)](#)
- [标准 ACL \(p. 337\)](#)
- [如何指定 ACL \(p. 338\)](#)

Amazon S3 访问控制列表 (ACL) 使您可以管理对存储桶和对象的访问权限。每个存储桶和对象都有一个作为子资源而附加的 ACL。它定义了哪些 AWS 账户或组将被授予访问权限以及访问的类型。收到针对某个资源的请求后，Amazon S3 将检查相应的 ACL 以验证请求者是否拥有所需的访问权限。

创建存储桶或对象时，Amazon S3 将创建一个默认 ACL 以授予资源拥有者对资源的完全控制权限。这显示在下面的示例存储桶 ACL 中 (默认对象 ACL 具有相同的结构)：

### Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
```

```
</AccessControlPolicy>
```

示例 ACL 包含一个可通过 AWS 账户的规范用户 ID 识别所有者的 `Owner` 元素。有关查找规范用户 ID 的说明，请参阅[查找 AWS 账户的规范用户 ID \(p. 334\)](#)。`Grant` 元素将识别被授权者 (AWS 账户或预定义的组) 和所授予的权限。此默认的 ACL 拥有一个适用于所有者的 `Grant` 元素。您可以通过添加 `Grant` 元素授予权限，每个授权都将识别被授权者和权限。

**Note**

ACL 可以拥有最多 100 个授权。

## 谁是被授权者？

被授权者可以是 AWS 账户或某个预定义的 Amazon S3 组。您可以使用电子邮件地址或规范用户 ID 向 AWS 账户授予权限。但是，如果您在授权请求中提供电子邮件地址，Amazon S3 将为该账户查找规范用户 ID 并将它添加到 ACL。生成的 ACL 将始终包含 AWS 账户的规范用户 ID，而不是 AWS 账户的电子邮件地址。

**Important**

仅以下 AWS 区域中支持使用电子邮件地址指定被授权者：

- 美国东部 ( 弗吉尼亚北部 )
- 美国西部 ( 加利福尼亚北部 )
- 美国西部 ( 俄勒冈 )
- 亚太区域 ( 新加坡 )
- 亚太区域 ( 悉尼 )
- 亚太区域 ( 东京 )
- 欧洲 ( 爱尔兰 )
- 南美洲 ( 圣保罗 )

有关所有 Amazon S3 支持的区域和终端节点的列表，请参阅 AWS 一般参考 中的[区域和终端节点](#)。

**Warning**

当您需要向其他 AWS 账户授权访问您的资源时，注意 AWS 账户可以向其账户下的用户授予权限。这称为跨账户访问。有关使用跨账户访问的信息，请参阅 IAM 用户指南 中的[创建角色以向 IAM 用户委派权限](#)。

## 查找 AWS 账户的规范用户 ID

规范用户 ID 与您的 AWS 账户关联。它是一个长字符串，例如 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be。有关如何查找您的账户的规范用户 ID 的信息，请参阅[查找账户的规范用户 ID](#)。

您也可以通过读取 AWS 账户有权访问的存储桶或对象的 ACL，查找 AWS 账户的规范用户 ID。如果某个 AWS 账户通过授权请求被授予了许可，ACL 中将新增一个授权条目和 AWS 账户的规范用户 ID。

**Note**

如果您公开存储桶（不建议），则任何未经身份验证的用户都可以将对象上传到存储桶。这些匿名用户没有 AWS 账户。当匿名用户将对象上传到您的存储桶时，Amazon S3 会在 ACL 中添加特殊的规范用户 ID (65a011a29cdf8ec533ec3d1ccaae921c) 作为对象所有者。

## Amazon S3 预定义的组

Amazon S3 拥有一系列预定义的组。将账户访问权限授予某个组时，您可以指定我们的一个 URI，而不是规范用户 ID。我们提供以下预定义的组：

- 经身份验证的用户组 – 由 `http://acs.amazonaws.com/groups/global/AuthenticatedUsers` 表示。

该组代表了所有的 AWS 账户。该组的访问许可允许任何 AWS 账户访问资源。但是，所有的请求必须是已签名的 (经身份验证)。

### Warning

在您为 Authenticated Users 组授予访问权限后，世界上的任何经过身份验证的 AWS 用户都可以访问您的资源。

- 所有的用户组 – 由 `http://acs.amazonaws.com/groups/global/AllUsers` 表示。

授予此组的访问权限将允许世界上的任何人访问资源。请求可以是已签名的 (经身份验证)，也可以是未签名的 (匿名)。未签名的请求将省略请求中的 Authentication 标头。

### Warning

强烈建议您绝不要向 All Users 组授予 WRITE、WRITE\_ACP 或 FULL\_CONTROL 权限。例如，WRITE 权限将允许任何人在您的存储桶中存储对象，而您需要为其付费。它还会允许其他人删除您可能希望保留的对象。有关这些权限的详细信息，请参阅下一节[我能授予哪些许可？\(p. 335\)](#)。

- 日志传输组 – 由 `http://acs.amazonaws.com/groups/s3/LogDelivery` 表示。

存储桶上的 WRITE 许可使该组可以将服务器访问日志 (参阅 [Amazon S3 服务器访问日志记录 \(p. 506\)](#)) 写入存储桶。

### Note

使用 ACL 时，被授权者可以是 AWS 账户或是某个预定义的 Amazon S3 组。但是，被授权者不能是 IAM 用户。有关 IAM 中 AWS 用户和许可的更多信息，请参阅[使用 AWS Identity and Access Management](#)。

## 我能授予哪些许可？

下表列出了 Amazon S3 在 ACL 中支持的权限集。对于对象 ACL 和存储桶 ACL，ACL 权限集相同。但是，根据上下文 (存储桶 ACL 或对象 ACL)，这些 ACL 权限将授予针对特定存储桶或对象操作的权限。下表列出了权限并描述这些权限在对象和存储桶上下文中的意义。

许可	在存储桶上授权的时间	在对象上授权的时间
READ	允许被授权者列出存储桶中的对象	允许被授权者读取对象数据及其元数据
WRITE	允许被授权者创建、覆盖和删除存储桶中的任意对象	不适用
READ_ACP	允许被授权者读取存储桶 ACL	允许被授权者读取对象 ACL
WRITE_ACP	允许被授权者为适用的存储桶编写 ACL	允许被授权者为适用的对象编写 ACL
FULL_CONTROL	允许被授权者在存储桶上的 READ、WRITE、READ_ACP 和 WRITE_ACP 许可	允许被授权者在对象上的 READ、READ_ACP 和 WRITE_ACP 许可

### Warning

在授予对您的 S3 存储桶和对象的访问权限时应谨慎使用。例如，授予对存储桶的 WRITE 权限将允许被授权者创建、覆盖和删除存储桶中的任何对象。我们强烈建议您在授予权限之前通读整个[访问控制列表 \(ACL\) 概述 \(p. 333\)](#)部分。

## ACL 权限和访问策略权限的映射

如先前表中所示，相比于在访问策略中可设置的权限数目（请参阅[在策略中指定权限 \(p. 282\)](#)），ACL 仅允许授予有限数量的权限。其中每个权限允许一个或多个 Amazon S3 操作。

下表显示每个 ACL 权限如何映射到相应的访问策略权限。正如您所见，与 ACL 相比，访问策略允许的权限更多。您可以将 ACL 主要用于授予基本的读/写权限（类似于文件系统权限）。有关何时使用 ACL 的更多信息，请参阅[有关使用可用访问策略选项的准则 \(p. 252\)](#)。

ACL 权限	当在存储桶上授予 ACL 权限时的相应访问策略	当在对象上授予 ACL 权限时的相应访问策略
READ	s3>ListBucket、s3>ListBucketVersions 和 s3>ListBucketMultipartUploads	s3GetObject、s3GetObjectVersion 和 s3GetObjectTorrent
WRITE	s3PutObject 和 s3DeleteObject。  此外，如果被授权者是存储桶拥有者，使用存储桶 ACL 来授予 WRITE 许可将允许对该存储桶中的任意版本执行 s3DeleteObjectVersion 操作。	不适用
READ_ACP	s3GetBucketAcl	s3GetObjectAcl 和 s3GetObjectVersionAcl
WRITE_ACP	s3PutBucketAcl	s3PutObjectAcl 和 s3PutObjectVersionAcl
FULL_CONTROL	等同于授予 READ、WRITE、READ_ACP 和 WRITE_ACP ACL 权限。因此，此 ACL 权限将映射到相应访问策略权限的组合。	等同于授予 READ、READ_ACP 和 WRITE_ACP ACL 权限。因此，此 ACL 权限将映射到相应访问策略权限的组合。

## 示例 ACL

下面的存储桶上的示例 ACL 可识别资源所有者和一系列的授权。格式是 Amazon S3 REST API 中的 ACL 的 XML 表示形式。存储桶拥有者拥有资源的 FULL\_CONTROL。此外，ACL 演示了如何将某个资源的权限授予两个 AWS 账户、规范用户 ID 如何识别权限，以及上一节讨论的两个预定义 Amazon S3 组。

### Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>Owner-canonical-user-ID</ID>
    <DisplayName>display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
```

```

<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
    <ID>Owner-canonical-user-ID</ID>
    <DisplayName>display-name</DisplayName>
</Grantee>
<Permission>FULL_CONTROL</Permission>
</Grant>

<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>user1-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>

<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>user2-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
    </Grantee>
    <Permission>READ</Permission>
</Grant>

<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
</Grant>
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>

</AccessControlList>
</AccessControlPolicy>

```

## 标准 ACL

Amazon S3 支持一系列预定义的授权，称为标准 ACL。每个标准 ACL 都有一组预定义的被授权者和许可。下表列出了一系列标准 ACL 和相关联的预定义授权。

标准 ACL	适用于	添加到 ACL 的权限
private	存储桶和对象	所有者将获得 FULL_CONTROL。其他人没有访问权限（默认）。
public-read	存储桶和对象	所有者将获得 FULL_CONTROL。AllUsers 组（参阅 <a href="#">谁是被授权者？(p. 334)</a> ）将获得 READ 访问权限。
public-read-write	存储桶和对象	所有者将获得 FULL_CONTROL。AllUsers 组将获得 READ 和 WRITE 访问权限。通常不建议在存储桶上授予该权限。
aws-exec-read	存储桶和对象	所有者将获得 FULL_CONTROL。Amazon EC2 从 Amazon S3 获取对 GET Amazon 系统映像 (AMI) 缀绑的 READ 访问权限。

标准 ACL	适用于	添加到 ACL 的权限
authenticated-read	存储桶和对象	所有者将获得 FULL_CONTROL。AuthenticatedUsers 组将获得 READ 访问权限。
bucket-owner-read	对象	对象所有者将获得 FULL_CONTROL。存储桶拥有者将获得 READ 访问权限。如果您在创建存储桶时指定此标准 ACL , Amazon S3 将忽略它。
bucket-owner-full-control	对象	对象所有者和存储桶拥有者均可获得对对象的 FULL_CONTROL。如果您在创建存储桶时指定此标准 ACL , Amazon S3 将忽略它。
log-delivery-write	存储桶	LogDelivery 组将获得针对存储桶的 WRITE 和 READ_ACP 许可。有关日志的更多信息 , 请参阅 ( <a href="#">Amazon S3 服务器访问日志记录 (p. 506)</a> )。

#### Note

您可以在请求中仅指定这些标准 ACL 中的一个。

您可以使用 `x-amz-acl` 请求标头在请求中指定标准的 ACL。当 Amazon S3 收到包含标准 ACL 的请求时 , 它会向资源的 ACL 添加预定义的授权。

## 如何指定 ACL

Amazon S3 API 使您可以在创建存储桶或对象时设置 ACL。Amazon S3 还提供 API 以在现有存储桶或对象上设置 ACL。这些 API 提供了以下方法来设置 ACL :

- 使用请求标头设置 ACL – 发送创建资源 (存储桶或对象) 的请求时 , 您可以使用请求标头设置 ACL。使用这些标头时 , 您可以指定一个标准 ACL 或者显式指定授权 (显式识别被授权者和许可)。
- 使用请求正文设置 ACL – 当您发送在现有资源上设置 ACL 的请求时 , 您可以在请求标头或正文中设置 ACL。

有关更多信息 , 请参阅 [管理 ACL \(p. 338\)](#)。

## 管理 ACL

### 主题

- [在 AWS 管理控制台中管理 ACL \(p. 338\)](#)
- [使用 AWS SDK for Java 管理 ACL \(p. 339\)](#)
- [使用适用于 .NET 的 AWS 开发工具包管理 ACL \(p. 341\)](#)
- [使用 REST API 管理 ACL \(p. 344\)](#)

您可以使用多种方法来向您的资源 ACL 添加授权。您可以使用 AWS 管理控制台 , 该控制台提供了一个用于管理权限的 UI 且无需编写任何代码。您可以使用 REST API 或 AWS 开发工具包之一。这些数据库将进一步简化您的编程任务。

## 在 AWS 管理控制台中管理 ACL

AWS 管理控制台提供了一个 UI , 您可以使用它授予对存储桶和对象的基于 ACL 的访问权限。有关在控制台中设置基于 ACL 的访问权限的信息 , 请参阅 [如何设置 ACL 存储桶策略 ?](#) 和 [如何在对象上设置权限 ?](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

## 使用AWS SDK for Java管理 ACL

本节提供了有关如何在存储桶和对象上配置访问控制列表 (ACL) 授予的示例。第一个示例将创建具有标准 ACL (请参阅[标准 ACL \(p. 337\)](#)) 的存储桶，创建自定义权限授予列表，然后将标准 ACL 替换为包含自定义授予的 ACL。第二个示例演示如何使用 `AccessControlList.grantPermission()` 方法修改 ACL。

### 设置 ACL 授予

#### Example

此示例将创建一个存储桶。在请求中，此示例指定了一个标准 ACL，该 ACL 向日志传输组授予将日志写入到存储桶的权限。

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collection;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CannedAccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.Grant;
import com.amazonaws.services.s3.model.GroupGrantee;
import com.amazonaws.services.s3.model.Permission;

public class CreateBucketWithACL {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Create a bucket with a canned ACL. This ACL will be deleted by the
            // getGrantsAsList().clear() call below. It is here for demonstration
            // purposes.
            CreateBucketRequest createBucketRequest = new CreateBucketRequest(bucketName,
clientRegion)
                .withCannedAcl(CannedAccessControlList.LogDeliveryWrite);
            s3Client.createBucket(createBucketRequest);

            // Create a collection of grants to add to the bucket.
            Collection<Grant> grantCollection = new ArrayList<Grant>();

            // Grant the account owner full control.
            Grant grant1 = new Grant(new
CanonicalGrantee(s3Client.getS3AccountOwner().getId()), Permission.FullControl);
            grantCollection.add(grant1);

            // Grant the LogDelivery group permission to write to the bucket.
            Grant grant2 = new Grant(GroupGrantee.LogDelivery, Permission.Write);
            grantCollection.add(grant2);

            // Save (replace) grants by deleting all current ACL grants and replacing
        }
    }
}
```

```
// them with the two we just created.  
AccessControlList bucketAcl = s3Client.getBucketAcl(bucketName);  
bucketAcl.getGrantsAsList().clear();  
bucketAcl.getGrantsAsList().addAll(grantCollection);  
s3Client.setBucketAcl(bucketName, bucketAcl);  
}  
catch(AmazonServiceException e) {  
    // The call was transmitted successfully, but Amazon S3 couldn't process  
    // it and 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();  
}  
}  
}  
}
```

## 在现有对象上配置 ACL 授予

### Example

此示例将更新对象上的 ACL。该示例执行以下任务：

- 检索对象的 ACL
- 通过删除所有现有权限来清除该 ACL
- 添加两个权限：对所有者的完全访问权限以及对通过电子邮件地址标识的用户的 WRITE\_ACP 权限（请参阅[我能授予哪些许可？\(p. 335\)](#)）。
- 将 ACL 保存到对象

```
import java.io.IOException;  
  
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.SdkClientException;  
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.services.s3.AmazonS3;  
import com.amazonaws.services.s3.AmazonS3ClientBuilder;  
import com.amazonaws.services.s3.model.AccessControlList;  
import com.amazonaws.services.s3.model.CanonicalGrantee;  
import com.amazonaws.services.s3.model.EmailAddressGrantee;  
import com.amazonaws.services.s3.model.Permission;  
  
public class ModifyACLExistingObject {  
  
    public static void main(String[] args) throws IOException {  
        String clientRegion = "**** Client region ****";  
        String bucketName = "**** Bucket name ****";  
        String keyName = "**** Key name ****";  
        String emailGrantee = "**** user@example.com ****";  
  
        try {  
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()  
                .withCredentials(new ProfileCredentialsProvider())  
                .withRegion(clientRegion)  
                .build();  
  
            // Get the existing object ACL that we want to modify.  
            AccessControlList acl = s3Client.getObjectAcl(bucketName, keyName);  
        }  
    }  
}
```

```
// Clear the existing list of grants.  
acl.getGrantsAsList().clear();  
  
    // Grant a sample set of permissions, using the existing ACL owner for Full  
    Control permissions.  
    acl.grantPermission(new CanonicalGrantee(acl.getOwner().getId()),  
Permission.FullControl);  
    acl.grantPermission(new EmailAddressGrantee(emailGrantee),  
Permission.WriteAcp);  
  
    // Save the modified ACL back to the object.  
    s3Client.setObjectAcl(bucketName, keyName, acl);  
}  
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();  
}  
}  
}  
}
```

## 使用适用于 .NET 的 AWS 开发工具包管理 ACL

本节提供在 Amazon S3 存储桶和对象上配置 ACL 授予的示例。

### 示例 1：创建存储桶并使用标准 ACL 设置权限

此 C# 示例将创建一个存储桶。在请求中，代码还将指定一个标准 ACL，该 ACL 向日志传输组授予将日志写入到存储桶的权限。

有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-  
developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using Amazon.S3;  
using Amazon.S3.Model;  
using System;  
using System.Threading.Tasks;  
  
namespace Amazon.DocSamples.S3  
{  
    class ManagingBucketACLTTest  
    {  
        private const string newBucketName = "*** 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()  
        {  
            client = new AmazonS3Client(bucketRegion);  
            CreateBucketUseCannedACLAsync().Wait();  
        }  
  
        private static async Task CreateBucketUseCannedACLAsync()
```

```
{  
    try  
    {  
        // Add bucket (specify canned ACL).  
        PutBucketRequest putBucketRequest = new PutBucketRequest()  
        {  
            BucketName = newBucketName,  
            BucketRegion = S3Region.EUW1, // S3Region.US,  
            // Add canned ACL.  
            CannedACL = S3CannedACL.LogDeliveryWrite  
        };  
        PutBucketResponse putBucketResponse = await  
client.PutBucketAsync(putBucketRequest);  
  
        // Retrieve bucket ACL.  
        GetACLResponse getACLResponse = await client.GetACLAsync(new GetACLRequest  
        {  
            BucketName = newBucketName  
        });  
    }  
    catch (AmazonS3Exception amazonS3Exception)  
    {  
        Console.WriteLine("S3 error occurred. Exception: " +  
amazonS3Exception.ToString());  
    }  
    catch (Exception e)  
    {  
        Console.WriteLine("Exception: " + e.ToString());  
    }  
}  
}  
}
```

## 示例 2：在现有对象上配置 ACL 授予

此 C# 示例将更新现有对象上的 ACL。该示例执行以下任务：

- 检索对象的 ACL。
- 通过删除所有现有权限来清除该 ACL。
- 添加两个权限：对所有者的完全访问权限以及对通过电子邮件地址标识的用户的 WRITE\_ACP 权限。
- 通过发送 PutAcl 请求来保存该 ACL。

有关创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-  
developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using Amazon.S3;  
using Amazon.S3.Model;  
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
  
namespace Amazon.DocSamples.S3  
{  
    class ManagingObjectACLTTest  
    {  
        private const string bucketName = "*** bucket name ***";  
        private const string keyName = "*** object key name ***";  
        private const string emailAddress = "*** email address ***";
```

```
// Specify your bucket region (an example region is shown).
private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
private static IAmazonS3 client;
public static void Main()
{
    client = new AmazonS3Client(bucketRegion);
    TestObjectACLTestAsync().Wait();
}

private static async Task TestObjectACLTestAsync()
{
    try
    {
        // Retrieve the ACL for the object.
        GetACLResponse aclResponse = await client.GetACLAsync(new GetACLRequest
        {
            BucketName = bucketName,
            Key = keyName
        });

        S3AccessControlList acl = aclResponse.AccessControlList;

        // Retrieve the owner (we use this to re-add permissions after we clear
        the ACL).
        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).
        S3Grant fullControlGrant = new S3Grant
        {
            Grantee = new S3Grantee { CanonicalUser = owner.Id },
            Permission = S3Permission.FULL_CONTROL
        };

        // Describe the grant for the permission using an email address.
        S3Grant grantUsingEmail = new S3Grant
        {
            Grantee = new S3Grantee { EmailAddress = emailAddress },
            Permission = S3Permission.WRITE_ACP
        };
        acl.Grants.AddRange(new List<S3Grant> { fullControlGrant,
        grantUsingEmail });

        // Set a new ACL.
        PutACLResponse response = await client.PutACLAsync(new PutACLRequest
        {
            BucketName = bucketName,
            Key = keyName,
            AccessControlList = acl
        });
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
amazonS3Exception.ToString());
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.ToString());
    }
}
}
```

}

## 使用 REST API 管理 ACL

有关针对 ACL 管理的 REST API 支持的信息，请参阅 Amazon Simple Storage Service API Reference 中的以下各部分：

- [GET Bucket acl](#)
- [PUT Bucket acl](#)
- [GET Object acl](#)
- [PUT Object acl](#)
- [PUT Object](#)
- [PUT Bucket](#)
- [PUT Object – Copy](#)
- [开始分段上传](#)

# 在 Amazon S3 中保护数据

## 主题

- [使用加密保护数据 \(p. 345\)](#)
- [使用版本控制 \(p. 380\)](#)

Amazon S3 为任务关键型和主数据存储提供了高度耐用的存储基础设施。在 Amazon S3 区域，对象以冗余方式存储在多个设施间的多个设备中。为帮助确保数据持久性，Amazon S3 PUT 和 PUT Object copy 操作会在多个设施间同步存储您的数据，然后才会返回 SUCCESS。存储对象后，Amazon S3 将通过快速检测和修复任何丢失的冗余数据，保持对象的持久性。

Amazon S3 还会使用校验和定期验证所存储数据的完整性。如果 Amazon S3 检测到数据损坏，它将使用冗余的数据进行修复。此外，Amazon S3 还会在存储或检索数据时对所有网络流量计算校验和，以检测数据包是否损坏。

Amazon S3 的标准存储：

- 以 [Amazon S3 服务等级协议](#) 作为后盾
- 设计目的是在指定年度内为对象提供 99.99999999% 的持久性和 99.99% 的可用性
- 能够承受同时两个设施中的数据丢失

Amazon S3 使用版本控制功能进一步保护您的数据。对于 Amazon S3 存储桶中存储的每个对象，您可以使用版本控制功能来保留、检索和还原它们的各个版本。使用版本控制能够轻松从用户意外操作和应用程序故障中恢复数据。默认情况下，请求会检索最新写入的版本。您可以通过指定请求中对象的版本，来检索该对象的旧版本。

# 使用加密保护数据

## 主题

- [使用服务器端加密保护数据 \(p. 345\)](#)
- [使用客户端加密保护数据 \(p. 372\)](#)

数据保护指在数据传输（发往和离开 Amazon S3 时）和处于静态（存储在 Amazon S3 数据中心的磁盘上时）期间保护数据。可以使用 SSL 或使用客户端加密保护传输中的数据。您可以通过以下选项在 Amazon S3 中保护静态数据。

- 使用服务器端加密 – 请求 Amazon S3 在将对象保存到数据中心的磁盘上之前加密对象，并在下载对象时进行解密。
- 使用客户端加密 – 可以在客户端加密数据并将加密的数据上传到 Amazon S3。在这种情况下，您需要管理加密过程、加密密钥和相关的工具。

## 使用服务器端加密保护数据

服务器端加密关乎静态数据加密，即 Amazon S3 在将您的数据写入数据中心内的磁盘时会在对象级别上加密这些数据，并在您访问这些数据时为您解密这些数据。只要您验证了您的请求并且拥有访问权限，您访问

加密和未加密对象的方式就没有区别。例如，如果您使用预签名的 URL 来共享您的对象，那么对于加密和解密对象，该 URL 的工作方式是相同的。

**Note**

您不能对同一个对象应用不同类型的服务器端加密。

您有三个互斥选项，具体取决于您选择如何管理加密密钥：

- 使用具有 Amazon S3 托管密钥的服务器端加密 (SSE-S3) – 利用多因素强加密来使用唯一密钥加密每个对象。作为额外的保护，它将使用定期轮换的主密钥加密密钥本身。Amazon S3 服务器端加密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。有关更多信息，请参阅 [使用具有 Amazon S3 托管加密密钥的服务器端加密 \(SSE-S3\) 保护数据 \(p. 351\)](#)。
- 使用具有 AWS KMS 托管密钥的服务器端加密 (SSE-KMS) – 与 SSE-S3 类似，但使用此服务有一些额外好处以及一些额外费用。使用信封密钥 (即，保护数据的加密密钥的密钥) 需要单独的权限，信封密钥可进一步防止未经授权地访问 S3 中的对象。SSE-KMS 还提供您的密钥的使用时间和使用者的审核跟踪。此外，您可以选择自己创建和管理加密密钥，或使用对您所使用的服务和您的工作区域来说具有唯一性的默认密钥。有关更多信息，请参阅 [使用具有 AWS KMS 托管密钥的服务器端加密 \(SSE-KMS\) 保护数据 \(p. 346\)](#)。
- 将服务器端加密与客户提供的密钥一起使用 (SSE-C) – 您管理加密密钥，而 Amazon S3 管理加密 (在对磁盘进行写入时) 和解密 (在您访问您的对象时)。有关更多信息，请参阅 [通过使用客户提供的加密密钥的服务器端加密 \(SSE-C\) 保护数据 \(p. 359\)](#)。

**Note**

在您列出存储桶中的对象时，列表 API 会返回所有对象的列表 (无论对象是否加密)。

## 使用具有 AWS KMS 托管密钥的服务器端加密 (SSE-KMS) 保护数据

服务器端加密是为了保护静态数据。AWS Key Management Service (AWS KMS) 是一项将安全、高度可用的硬件和软件结合起来，提供可扩展到云的密钥管理系统的服务。AWS KMS 使用客户主密钥 (CMK) 加密您的 Amazon S3 对象。可以通过 IAM 控制台中的“Encryption Keys”部分或通过 AWS KMS API 使用 AWS KMS 来集中创建加密密钥，定义策略以控制密钥的使用方法，以及审核密钥使用情况来证明它们使用得当。您可以利用这些密钥来保护您在 Amazon S3 存储桶中的数据。

首次向区域中的存储桶添加 SSE-KMS 加密的对象时，将自动为您创建一个默认 CMK。除非您选择了使用 AWS Key Management Service 单独创建的 CMK，否则此密钥将用于 SSE-KMS 加密。创建您自己的 CMK 可为您提供更大灵活性，包括创建、轮换、禁用和定义访问控制，以及审核用于保护数据的加密密钥的能力。

有关更多信息，请参阅 [什么是 AWS Key Management Service？\(在 AWS Key Management Service Developer Guide 中\)](#)。如果您使用 AWS KMS，则使用 AWS-KMS 密钥会产生额外费用。有关更多信息，请参阅 [AWS Key Management Service 定价](#)。

**Note**

如果您要上传或访问使用 SSE-KMS 加密的对象，则需使用 AWS 签名版本 4 来提高安全性。有关如何使用 AWS SDK 执行此操作的更多信息，请参阅 [在请求身份验证中指定签名版本](#)。

SSE-KMS 的要点是：

- 您可以选择自行创建和管理加密密钥，也可以选择使用由服务按区域级别为某个客户生成的唯一的默认服务密钥。
- 响应中的 ETag 不是对象数据的 MD5。
- 用于加密您的数据的数据密钥也会被加密并与它们保护的数据一起存储。
- 可以从 IAM 控制台创建、轮换或禁用可审核的主密钥。

- AWS KMS 中的安全控制可帮助您满足与加密相关的合规性要求。

Amazon S3 支持存储桶策略，如果您要对所有存储在存储桶中的对象执行服务器端加密，则可以使用这些策略。例如，如果请求不包含用于请求服务器端加密 (SSE-KMS) 的 s3:PutObject 标头，则下面的存储桶策略将拒绝所有人的上传对象 (x-amz-server-side-encryption) 权限。

```
{  
    "Version": "2012-10-17",  
    "Id": "PutObjPolicy",  
    "Statement": [  
        {  
            "Sid": "DenyUnEncryptedObjectUploads",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::YourBucket/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "aws:kms"  
                }  
            }  
        }  
    ]  
}
```

Amazon S3 也支持 s3:x-amz-server-side-encryption-aws-kms-key-id 条件键，您可以使用该条件键来获取特定的对象加密 KMS 密钥。您在策略中指定的 KMS 密钥必须使用 arn:aws:kms:[region:acct-id:key/key-id](#) 格式。

**Note**

在上传对象时，您可以使用 x-amz-server-side-encryption-aws-kms-key-id 标头指定 KMS 密钥。如果请求中没有该标头，则 Amazon S3 采用默认 KMS 密钥。无论如何，Amazon S3 用于对象加密的 KMS 密钥 ID 必须与策略中的 KMS 密钥 ID 匹配，否则 Amazon S3 会拒绝请求。

**Important**

如果针对受 AWS KMS 保护的对象的所有 GET 和 PUT 请求不是通过 SSL 或通过使用 SigV4 发出的，则这些请求将会失败。

SSE-KMS 仅加密对象数据。不会加密任何对象元数据。

## 在 Amazon S3 管理控制台中使用 AWS Key Management Service

有关在 Amazon S3 管理控制台中使用 KMS 托管加密密钥的更多信息，请参阅 Amazon Simple Storage Service 用户指南 中的[上传 S3 对象](#)。

## Amazon S3 中针对 AWS Key Management Service 的 API 支持

对象创建 REST API (请参阅 [使用 REST API 在 Amazon S3 中指定 AWS Key Management Service \(p. 350\)](#)) 提供了一个请求标头 x-amz-server-side-encryption，您可以使用该标头和值 aws:kms 来请求 SSE-KMS。此外还有 x-amz-server-side-encryption-aws-kms-key-id，它指定用于对象的 AWS KMS 主加密密钥的 ID。Amazon S3 API 还使用 x-amz-server-side-encryption-context 标头支持加密上下文。

加密上下文可以是您希望的任何值，前提是该标头遵循 Base64 编码的 JSON 格式。但是，由于加密上下文不会加密并且在 AWS CloudTrail 记录打开时会被记录，因此加密上下文不应包含敏感信息。我们进一步建议您的上下文描述所加密或解密的数据，以便让您可以更好地了解 AWS KMS 生成的 CloudTrail 事件。有关更多信息，请参阅 AWS Key Management Service Developer Guide 中的[加密上下文](#)。

此外，Amazon S3 还可为预定义密钥 aws:s3:arn 追加等于对象的 ARN (用于您提供的加密上下文) 的值。这仅在密钥 aws:s3:arn 尚不存在于您提供的加密上下文中时发生，在这种情况下，此预定义密钥在 Amazon

S3 处理您的 Put 请求时追加。如果此 aws:s3:arn 密钥已存在于您的加密上下文中，则不会再次将它追加到您的加密上下文。

加密上下文包含此预定义密钥则意味着，您可以在 CloudTrail 中跟踪相关请求，这样您始终能够查看哪个 S3 对象的 ARN 与哪个加密密钥一起使用。此外，作为您的加密上下文的一部分的此预定义密钥保证了不同的 S3 对象之间拥有不同的加密上下文，从而为您的对象提供更高的安全性。系统将验证您的完全加密上下文是否有等于对象的 ARN 的值。

以下 Amazon S3 API 支持这些请求标头。

- PUT 操作 - 使用 PUT API 上传数据 (请参阅 [PUT Object](#)) 时，可以指定这些请求标头。
- Initiate Multipart Upload - 使用分段上传 API 上传大型对象时，可以指定这些标头。可以在启动请求中指定这些标头 (请参阅 [Initiate Multipart Upload](#))。
- POST 操作 - 使用 POST 操作上传对象 (请参阅 [POST Object](#)) 时，可在表单字段而不是请求标头中提供相同的信息。
- COPY 操作 - 复制对象 (请参阅 [PUT Object - Copy](#)) 时，您同时具有源对象和目标对象。如果使用 COPY 操作传递 SSE-KMS 标头，它们将仅应用于目标对象。

AWS 软件开发工具包还提供了一个包装程序 API，您可以使用它在 Amazon S3 中请求 SSE-KMS。

## 使用 AWS 软件开发工具包在 Amazon S3 中指定 AWS Key Management Service

### 主题

- [适用于 Java 的 AWS 开发工具包 \(p. 348\)](#)
- [适用于 .NET 的 AWS 开发工具包 \(p. 349\)](#)

当使用 AWS 开发工具包时，您可以请求 Amazon S3 使用 AWS Key Management Service (AWS KMS) 托管加密密钥。此部分提供的示例演示了如何使用适用于 Java 和 .NET 的 AWS 开发工具包。有关其他开发工具包的信息，请转到[示例代码和库](#)。

### 适用于 Java 的 AWS 开发工具包

本节介绍使用适用于 Java 的 AWS 开发工具包进行的各种 Amazon S3 操作以及如何使用 AWS KMS 托管加密密钥。

#### Put 操作

当使用适用于 Java 的 AWS 开发工具包上传对象时，您可以通过添加 SSEAwsKeyManagementParams 属性请求 Amazon S3 使用 AWS KMS 托管加密密钥，如以下请求所示：

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,  
    keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

这种情况下，Amazon S3 使用默认主密钥 (参阅 [使用具有 AWS KMS 托管密钥的服务器端加密 \(SSE-KMS\) 保护数据 \(p. 346\)](#))。您可以选择创建自己的密钥并在请求中指定该密钥。

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,  
    keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams(keyID));
```

有关创建密钥的更多信息，请转到 AWS Key Management Service Developer Guide 中的[编程 AWS KMS API](#)。

有关上传对象的工作代码示例，请参阅以下主题。您将需要更新这些代码示例并提供加密信息，如上述代码片段所示。

- 有关在单个操作中上传对象，请参阅 [使用 AWS SDK for Java 上传对象 \(p. 151\)](#)
- 有关分段上传，请参阅以下主题：
  - 有关使用高级别分段上传 API，请参阅 [上传文件 \(p. 160\)](#)
  - 如果您使用低级别分段上传 API，请参阅 [上传文件 \(p. 164\)](#)

## 复制操作

在复制对象时，您将添加相同的请求属性 (`ServerSideEncryptionMethod` 和 `ServerSideEncryptionKeyManagementServiceKeyId`) 来请求 Amazon S3 使用 AWS KMS 托管加密密钥。有关复制对象的更多信息，请参阅 [复制对象 \(p. 187\)](#)。

## 预签名 URL

在为使用 AWS KMS 托管加密密钥加密的对象创建预签名 URL 时，您必须显式指定签名版本 4：

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
    new ProfileCredentialsProvider(), clientConfiguration);
...
```

有关代码示例，请参阅 [使用 AWS SDK for Java 生成预签名对象 URL \(p. 148\)](#)。

## 适用于 .NET 的 AWS 开发工具包

本节介绍使用适用于 .NET 的 AWS 开发工具包进行的各种 Amazon S3 操作以及如何使用 AWS KMS 托管加密密钥。

### Put 操作

当使用适用于 .NET 的 AWS 开发工具包上传对象时，您可以通过添加 `ServerSideEncryptionMethod` 属性请求 Amazon S3 使用 AWS KMS 托管加密密钥，如以下请求所示：

```
PutObjectRequest putRequest = new PutObjectRequest
{
    BucketName = bucketName,
    Key = keyName,
    // other properties.
    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
};
```

这种情况下，Amazon S3 使用默认主密钥（参阅 [使用具有 AWS KMS 托管密钥的服务器端加密 \(SSE-KMS\) 保护数据 \(p. 346\)](#)）。您可以选择创建自己的密钥并在请求中指定该密钥。

```
PutObjectRequest putRequest1 = new PutObjectRequest
{
    BucketName = bucketName,
    Key = keyName,
    // other properties.
    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
    ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

有关创建密钥的更多信息，请参阅 AWS Key Management Service Developer Guide 中的 [AWS KMS API 编程](#)。

有关上传对象的工作代码示例，请参阅以下主题。您将需要更新这些代码示例并提供加密信息，如上述代码片段所示。

- 有关在单个操作中上传对象，请参阅 [使用适用于 .NET 的 AWS 开发工具包 上传对象 \(p. 152\)](#)
- 有关分段上传，请参阅以下主题：
  - 有关使用高级别分段上传 API，请参阅 [使用适用于 .NET 的 AWS 开发工具包 \(高级别 API\) 将文件上传到 S3 存储桶 \(p. 169\)](#)
  - 有关使用低级别分段上传 API，请参阅 [使用适用于 .NET 的 AWS 开发工具包 \(低级别 API\) 将文件上传到 S3 存储桶 \(p. 175\)](#)

## 复制操作

在复制对象时，您将添加相同的请求属性 (`ServerSideEncryptionMethod` 和 `ServerSideEncryptionKeyManagementServiceKeyId`) 来请求 Amazon S3 使用 AWS KMS 托管加密密钥。有关复制对象的更多信息，请参阅 [复制对象 \(p. 187\)](#)。

## 预签名 URL

在为使用 AWS KMS 托管加密密钥加密的对象创建预签名 URL 时，您必须显式指定签名版本 4：

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

有关代码示例，请参阅 [使用适用于 .NET 的 AWS 开发工具包生成预签名对象 URL \(p. 149\)](#)。

## 使用 REST API 在 Amazon S3 中指定 AWS Key Management Service

在创建对象时，即，在上传新对象或复制现有对象时，可以通过向请求中添加 `x-amz-server-side-encryption` 标头，来指定使用具有 AWS KMS 托管加密密钥的服务器端加密 (SSE-KMS) 对您的数据进行加密。将标头的值设置为加密算法 `aws:kms`。Amazon S3 通过返回响应标头 `x-amz-server-side-encryption`，来确认您的对象已使用 SSE-KMS 存储。

以下 REST 上传 API 接受 `x-amz-server-side-encryption` 请求标题。

- [PUT Object](#)
- [PUT Object – Copy](#)
- [POST Object](#)
- [开始分段上传](#)

使用分段上传 API 上传大型对象时，可以通过使用 `aws:kms` 的值向启动分段上传请求添加 `x-amz-server-side-encryption` 标头来指定 SSE-KMS。复制现有的对象时，不论源对象是否已经加密，都不会加密目标对象，除非您显式请求了服务器端加密。

使用服务器端加密存储对象后，以下 REST API 的响应标头将返回 `x-amz-server-side-encryption` 标头。

- [PUT Object](#)
- [PUT Object – Copy](#)
- [POST Object](#)
- [开始分段上传](#)
- [上传分段](#)
- [上传分段 – 复制](#)
- [完成分段上传](#)
- [GET Object](#)
- [HEAD Object](#)

#### Note

如果对象使用 SSE-KMS，则不应对 GET 请求和 HEAD 请求发送加密请求标头，否则将出现 HTTP 400 BadRequest 错误。

## 使用具有 Amazon S3 托管加密密钥的服务器端加密 (SSE-S3) 保护数据

服务器端加密可保护静态数据。使用 Amazon S3 托管加密密钥的服务器端加密 (SSE-S3) 采用了多因素强加密。Amazon S3 使用唯一的密钥来加密每个对象。作为额外的保护，它将使用定期轮换的主密钥对密钥本身进行加密。Amazon S3 服务器端加密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。

如果需要对存储在存储桶中的所有对象执行服务器端加密，请使用存储桶策略。例如，以下存储桶策略拒绝上传对象的权限，除非请求包含用于请求服务器端加密的 `x-amz-server-side-encryption` 标头：

```
{
    "Version": "2012-10-17",
    "Id": "PutObjPolicy",
    "Statement": [
        {
            "Sid": "DenyIncorrectEncryptionHeader",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::YourBucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "AES256"
                }
            }
        },
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::YourBucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption": "true"
                }
            }
        }
    ]
}
```

服务器端加密仅加密对象数据而不是对象元数据。

## 服务器端加密的 API 支持

要使用对象创建 REST API 请求服务器端加密，请提供 `x-amz-server-side-encryption` 请求标头。有关 REST API 的信息，请参阅[使用 REST API 指定服务器端加密 \(p. 358\)](#)。

以下 Amazon S3 API 支持此标头：

- PUT 操作 - 在使用 PUT API 上传数据时指定请求标头。有关更多信息，请参阅[PUT Object](#)。
- 启动分段上传 - 当使用分段上传 API 上传大型对象时，在启动请求中指定标头。有关更多信息，请参阅[启动分段上传](#)。
- COPY 操作 - 在复制对象时，您同时具有源对象和目标对象。有关更多信息，请参阅[PUT 对象 - Copy](#)。

Note

当使用 POST 操作上传对象时，请在表单字段中提供相同的信息，而不是提供请求标头。有关更多信息，请参阅 [POST Object](#)。

AWS 开发工具包还提供了一个可用于请求服务器端加密的包装程序 API。您还可以使用 AWS 管理控制台来上传对象并请求服务器端加密。

Note

对于使用预签名 URL 上传的对象，无法强制执行 SSE-S3 加密。只能使用 AWS 管理控制台或 HTTP 请求标头指定服务器端加密。有关更多信息，请参阅 [在策略中指定条件 \(p. 286\)](#)。

## 使用AWS SDK for Java指定服务器端加密

当您使用AWS SDK for Java上传对象时，可以使用服务器端加密为对象加密。要请求服务器端加密，请使用 `PutObjectRequest` 的 `ObjectMetadata` 属性设置 `x-amz-server-side-encryption` 请求标头。当您调用 `AmazonS3Client` 的 `putObject()` 方法时，Amazon S3 将加密并保存数据。

当使用分段上传 API 上传对象时，还可以请求服务器端加密：

- 使用高级别分段上传 API 时，请在上传对象时使用 `TransferManager` 方法将服务器端加密应用于对象。可以使用将 `ObjectMetadata` 视为参数的任一上传方法。有关更多信息，请参阅 [使用适用于分段上传的 AWS Java 开发工具包 \(高级别 API\) \(p. 160\)](#)。
- 当使用低级别分段上传 API 时，应在启动分段上传时指定服务器端加密。您通过调用 `InitiateMultipartUploadRequest.setObjectMetadata()` 方法添加 `ObjectMetadata` 属性。有关更多信息，请参阅 [上传文件 \(p. 164\)](#)。

不能直接更改对象的加密状态(加密未加密的对象或解密已加密的对象)。要更改对象的加密状态，请为对象创建一个副本，从而为副本指定所需的加密状态，然后删除原始对象。仅当您显式请求服务器端加密时，Amazon S3 才会加密复制的对象。要通过 Java API 请求加密复制的对象，请使用 `ObjectMetadata` 属性在 `CopyObjectRequest` 中指定服务器端加密。

### Example 示例

以下示例演示如何使用AWS SDK for Java设置服务器端加密。它展示了如何执行以下任务：

- 使用服务器端加密上传新对象
- 通过为对象创建副本更改对象的加密状态(本示例中为加密之前未加密的对象)
- 检查对象的加密状态

有关服务器端加密的更多信息，请参阅[使用 REST API 指定服务器端加密 \(p. 358\)](#)。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.ByteArrayInputStream;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.internal.SSEResultBase;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.CopyObjectResult;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

public class SpecifyServerSideEncryption {
```

```
public static void main(String[] args) {
    String clientRegion = "*** Client region ***";
    String bucketName = "*** Bucket name ***";
    String keyNameToEncrypt = "*** Key name for an object to upload and encrypt
***";
    String keyNameToCopyAndEncrypt = "*** Key name for an unencrypted object to be
encrypted by copying ***";
    String copiedObjectKeyName = "*** Key name for the encrypted copy of the
unencrypted object ***";

    try {
        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
            .withRegion(clientRegion)
            .withCredentials(new ProfileCredentialsProvider())
            .build();

        // Upload an object and encrypt it with SSE.
        uploadObjectWithSSEEncryption(s3Client, bucketName, keyNameToEncrypt);

        // Upload a new unencrypted object, then change its encryption state
        // to encrypted by making a copy.
        changeSSEEncryptionStatusByCopying(s3Client,
            bucketName,
            keyNameToCopyAndEncrypt,
            copiedObjectKeyName);
    }
    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();
    }
}

private static void uploadObjectWithSSEEncryption(AmazonS3 s3Client, String bucketName,
String keyName) {
    String objectContent = "Test object encrypted with SSE";

    // Specify server-side encryption.
    ObjectMetadata objectMetadata = new ObjectMetadata();
    objectMetadata.setContentLength(objectContent.length());
    objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
    PutObjectRequest putRequest = new PutObjectRequest(bucketName,
        keyName,
        new
ByteArrayInputStream(objectContent.getBytes()),
        objectMetadata);

    // Upload the object and check its encryption status.
    PutObjectResult putResult = s3Client.putObject(putRequest);
    System.out.println("Object \"" + keyName + "\" uploaded with SSE.");
    printEncryptionStatus(putResult);
}

private static void changeSSEEncryptionStatusByCopying(AmazonS3 s3Client,
    String bucketName,
    String sourceKey,
    String destKey) {
    // Upload a new, unencrypted object.
    PutObjectResult putResult = s3Client.putObject(bucketName, sourceKey, "Object
example to encrypt by copying");
}
```

```
System.out.println("Unencrypted object \'" + sourceKey + "\' uploaded.");
printEncryptionStatus(putResult);

// Make a copy of the object and use server-side encryption when storing the copy.
CopyObjectRequest request = new CopyObjectRequest(bucketName,
                                                    sourceKey,
                                                    bucketName,
                                                    destKey);
ObjectMetadata objectMetadata = new ObjectMetadata();
objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
request.setNewObjectMetadata(objectMetadata);

// Perform the copy operation and display the copy's encryption status.
CopyObjectResult response = s3Client.copyObject(request);
System.out.println("Object \'" + destKey + "\' uploaded with SSE.");
printEncryptionStatus(response);

// Delete the original, unencrypted object, leaving only the encrypted copy in
Amazon S3.
s3Client.deleteObject(bucketName, sourceKey);
System.out.println("Unencrypted object \'" + sourceKey + "\' deleted.");
}

private static void printEncryptionStatus(SSEResultBase response) {
    String encryptionStatus = response.getSSEAlgorithm();
    if(encryptionStatus == null) {
        encryptionStatus = "Not encrypted with SSE";
    }
    System.out.println("Object encryption status is: " + encryptionStatus);
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包指定服务器端加密

在上传对象时，可指示 Amazon S3 加密对象。要更改现有对象的加密状态，请复制该对象并删除源对象。默认情况下，仅当您显式请求目标对象的服务器端加密时，复制操作才会加密目标。要在 `CopyObjectRequest` 中指定服务器端加密，请添加以下命令：

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

有关如何复制对象的有效示例，请参阅 [使用适用于 .NET 的 AWS 开发工具包在单个操作中复制 Amazon S3 对象 \(p. 189\)](#)。

以下示例将上传对象。在请求中，该示例指示 Amazon S3 加密对象。该示例随后检索对象元数据并验证使用的加密方法。有关创建和测试有效示例的信息，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class SpecifyServerSideEncryptionTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for object created ***";
```

```
// Specify your bucket region (an example region is shown).
private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
private static IAmazonS3 client;

public static void Main()
{
    client = new AmazonS3Client(bucketRegion);
    WritingAnObjectAsync().Wait();
}

static async Task WritingAnObjectAsync()
{
    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 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);
    }
}
}
```

### 更改现有对象的服务器端加密 (复制操作)

### 使用适用于 PHP 的 AWS 开发工具包指定服务器端加密

本主题介绍如何使用版本 3 的适用于 PHP 的 AWS 开发工具包中的类来将服务器端加密添加到您上传到 Amazon Simple Storage Service (Amazon S3) 的对象。此主题假定您已按照[使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)的说明执行操作，并且已正确安装适用于 PHP 的 AWS 开发工具包。

要将对象上传到 Amazon S3，请使用 [Aws\S3\S3Client::putObject\(\)](#) 方法。要将 `x-amz-server-side-encryption` 请求标头添加到您的上传请求，请使用值 AES256 指定 `ServerSideEncryption` 参数，如以下代码示例中所示。有关服务器端加密请求的信息，请参阅[使用 REST API 指定服务器端加密 \(p. 358\)](#)。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

// $filepath should be an absolute path to a file on disk.
$filepath = '*** Your File Path ***';

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

// Upload a file with server-side encryption.
$result = $s3->putObject([
    'Bucket'          => $bucket,
    'Key'             => $keyname,
    'SourceFile'      => $filepath,
    'ServerSideEncryption' => 'AES256',
]);

```

作为响应，Amazon S3 将返回 `x-amz-server-side-encryption` 标头以及已用于加密对象数据的加密算法的值。

在使用分段上传 API 上传大型对象时，您可以为正在上传的对象指定服务器端加密，如下所示：

- 如果使用低级别分段上传 API，请在调用 `Aws\S3\S3Client::createMultipartUpload()` 方法时指定服务器端加密。要向请求添加 `x-amz-server-side-encryption` 请求标头，请使用值 `array` 指定 `ServerSideEncryption` 参数的 AES256 密钥。有关低级别分段上传 API 的更多信息，请参阅[使用适用于分段上传的 AWS PHP 开发工具包 \(低级别 API\) \(p. 180\)](#)。
- 在使用高级别分段上传 API 时，请使用 `CreateMultipartUpload` 方法的 `ServerSideEncryption` 参数来指定服务器端加密。有关将 `setOption()` 方法与高级别分段上传 API 结合使用的示例，请参阅[使用 AWS PHP 开发工具包执行分段上传 \(p. 179\)](#)。

## 确定使用的加密算法

要确定现有对象的加密状态，请通过调用 `Aws\S3\S3Client::headObject()` 方法检索对象元数据，如下面的 PHP 代码示例所示。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// Check which server-side encryption algorithm is used.
$result = $s3->headObject([
    'Bucket' => $bucket,
    'Key'     => $keyname,
```

```
]);  
echo $result['ServerSideEncryption'];
```

### 更改现有对象的服务器端加密 (复制操作)

要更改现有对象的加密状态，请使用 [Aws\S3\S3Client::copyObject\(\)](#) 方法复制对象并删除源对象。请注意，默认情况下，`copyObject()` 不会加密目标，除非您通过将值 AES256 用于 `ServerSideEncryption` 参数，显式请求对目标对象进行服务器端加密。以下 PHP 代码示例将复制对象并向复制的对象添加服务器端加密。

```
<?php  
  
require 'vendor/autoload.php';  
  
use Aws\S3\S3Client;  
  
$sourceBucket = '*** Your Source Bucket Name ***';  
$sourceKeyname = '*** Your Source Object Key ***';  
  
$targetBucket = '*** Your Target Bucket Name ***';  
$targetKeyname = '*** Your Target Object Key ***';  
  
$s3 = new S3Client([  
    'version' => 'latest',  
    'region'  => 'us-east-1'  
]);  
  
// Copy an object and add server-side encryption.  
$s3->copyObject([  
    'Bucket'          => $targetBucket,  
    'Key'             => $targetKeyname,  
    'CopySource'      => "{$sourceBucket}/{$sourceKeyname}",  
    'ServerSideEncryption' => 'AES256',  
]);
```

### 相关资源

- [用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包 `Aws\S3\S3Client` 类](#)
- [适用于 PHP 的 AWS 开发工具包文档](#)

## 使用适用于 Ruby 的 AWS 开发工具包指定服务器端加密

在使用适用于 Ruby 的 AWS 开发工具包上传对象时，您可以指定使用服务器端加密 (SSE) 对存储的对象进行静态加密。在您读回对象时，它将自动解密。

下面的适用于 Ruby 的 AWS 开发工具包版本 3 示例演示了如何指定对上传到 Amazon S3 的文件进行静态加密。

```
require 'aws-sdk-s3'  
  
s3 = Aws::S3::Resource.new(region:'us-west-2')  
obj = s3.bucket('my-bucket').object('key')  
obj.upload_file('local/path/to/file', :server_side_encryption => 'AES256')
```

有关演示在不使用 SSE 的情况下上传对象的示例，请参阅[使用适用于 Ruby 的 AWS 开发工具包上传对象 \(p. 154\)](#)。

### 确定使用的加密算法

下面的代码示例演示了如何确定现有对象的加密状态。

```
# Determine server-side encryption of an object.
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(region:'us-west-2')
enc = s3.bucket('bucket-name').object('key').server_side_encryption
enc_state = (enc != nil) ? enc : "not set"
puts "Encryption state is #{enc_state}."
```

如果存储在 Amazon S3 中的对象没有使用服务器端加密，则该方法将返回空值。

### 更改现有对象的服务器端加密 (复制操作)

要更改现有对象的加密状态，请复制该对象并删除源对象。默认情况下，复制方法不会加密目标，除非您显式请求服务器端加密。您可以通过在选项哈希参数中指定 `server_side_encryption` 值来请求加密目标对象，如以下 Ruby 代码示例所示。此代码示例演示如何复制对象和加密副本。

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(region:'us-west-2')
bucket1 = s3.bucket('source-bucket-name')
bucket2 = s3.bucket('target-bucket-name')
obj1 = bucket1.object('key')
obj2 = bucket2.object('key')

obj1.copy_to(obj2, :server_side_encryption => 'AES256')
```

有关如何复制未加密对象的示例，请参阅 [使用适用于 Ruby 的 AWS 开发工具包 复制对象 \(p. 191\)](#)。

### 使用 REST API 指定服务器端加密

创建对象时，即，上传新对象或复制现有对象时，您可以通过为请求添加 `x-amz-server-side-encryption` 标头来指定您是否希望 Amazon S3 加密您的数据。将标头的值设置为 Amazon S3 支持的加密算法 AES256。通过返回响应标头 `x-amz-server-side-encryption`，Amazon S3 确认已使用服务器端加密存储了您的对象。

以下 REST 上传 API 接受 `x-amz-server-side-encryption` 请求标题。

- [PUT Object](#)
- [PUT Object – Copy](#)
- [POST Object](#)
- [开始分段上传](#)

使用分段上传 API 上传大型对象时，您可以通过为启动分段上传请求添加 `x-amz-server-side-encryption` 标头来指定服务器端加密。复制现有对象时，不论源对象是否已经加密，都不会加密目标对象，除非您显式请求服务器端加密。

使用服务器端加密存储对象后，以下 REST API 的响应标头将返回 `x-amz-server-side-encryption` 标头。

- [PUT Object](#)
- [PUT Object – Copy](#)
- [POST Object](#)
- [开始分段上传](#)
- [上传分段](#)
- [上传分段 – 复制](#)
- [完成分段上传](#)

- GET Object
- HEAD Object

#### Note

如果对象使用 SSE-S3，则不应向 GET 请求和 HEAD 请求发送加密请求标头，否则将出现 HTTP 400 BadRequest 错误。

## 使用 AWS 管理控制台指定服务器端加密

使用 AWS 管理控制台上传对象时，可以指定服务器端加密。有关如何上传对象的示例，请参阅[上传 S3 对象](#)。

使用 AWS 管理控制台复制对象时，控制台将按原样复制对象。也就是说，如果复制源加密，那么目标对象也将加密。控制台还允许您为对象添加加密。有关更多信息，请参阅[如何为 S3 对象添加加密？](#)

## 通过使用客户提供的加密密钥的服务器端加密 (SSE-C) 保护数据

服务器端加密是为了保护静态数据。使用客户提供的加密密钥的服务器端加密 (SSE-C) 允许您设置自己的加密密钥。使用您作为请求的一部分提供的加密密钥，Amazon S3 在写入磁盘时管理加密并在您访问对象时管理解密。因此，您不需要维护任何代码来执行数据加密和解密。您只需管理您提供的加密密钥。

在您上传对象时，Amazon S3 将使用您提供的加密密钥对您的数据应用 AES-256 加密并从内存中移除加密密钥。

#### Important

Amazon S3 不存储您提供的加密密钥，而是存储加密密钥添加了随机数据的 HMAC 值来验证未来的请求。无法使用添加了随机数据的 HMAC 值来推导出加密密钥的值或解密加密对象的内容。这意味着，如果您丢失加密密钥，则会失去该对象。

在检索对象时，必须提供相同的加密密钥作为您请求的一部分。Amazon S3 在将对象数据返回给您之前，会首先验证您提供的加密密钥是否匹配，然后再解密对象。

SSE-C 的要点是：

- 您必须使用 https。

#### Important

在使用 SSE-C 时，Amazon S3 会拒绝通过 http 提出的所有请求。出于安全原因，我们建议您考虑您错误地使用 http 发送的任何密钥都将被外泄。您应丢弃该密钥，并根据需要轮换密钥。

- 响应中的 ETag 不是对象数据的 MD5。
- 您管理哪个加密密钥用于加密哪个对象的映射。Amazon S3 不存储加密密钥。您负责跟踪为哪个对象提供了哪个加密密钥。
  - 如果您的存储桶启用了版本控制，则您使用此功能上传的每个对象版本可能都具有自己的加密密钥。您负责跟踪哪个加密密钥用于哪个对象版本。
  - 因为您在客户端管理加密密钥，所以也要在客户端管理所有额外的保护措施，例如密钥轮换。

#### Warning

如果您丢失加密密钥，则针对某个对象的没有其加密密钥的任何 GET 请求都将失败，并且您将失去该对象。

## 使用 SSE-C

在使用具有客户提供的加密密钥的服务器端加密 (SSE-C) 时，您必须使用以下请求标头提供加密密钥信息。

名称	说明
<code>x-amz-server-side-encryption-customer-algorithm</code>	使用此标头来指定加密算法。标头值必须为“AES256”。
<code>x-amz-server-side-encryption-customer-key</code>	使用此标头来提供 256 位的 base64 编码的加密密钥以供 Amazon S3 用于加密或解密您的数据。
<code>x-amz-server-side-encryption-customer-key-MD5</code>	使用此标头根据 <a href="#">RFC 1321</a> 提供加密密钥的 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的传输无误。

您可以使用 AWS 开发工具包包装库将这些标头添加到您的请求中。如果需要，您可以直接在应用程序中调用 Amazon S3 REST API。

#### Note

不能使用 Amazon S3 控制台来上传对象和请求 SSE-C。也不能使用控制台来更新使用 SSE-C 存储的现有对象（例如，更改存储类别或添加元数据）。

以下 Amazon S3 API 支持这些标头。

- GET 操作 - 在使用 GET API 检索对象（请参阅 [GET Object](#)）时，您可以指定请求标头。使用 SSE-C 加密的对象不支持 Torrent。
- HEAD 操作 - 要使用 HEAD API 检索对象元数据（请参阅 [HEAD Object](#)），可以指定这些请求标头。
- PUT 操作 - 使用 PUT API 上传数据（请参阅 [PUT Object](#)）时，可以指定这些请求标头。
- Multipart Upload - 在使用分段上传 API 上传大对象时，可以指定这些标头。您在启动请求（请参阅 [Initiate Multipart Upload](#)）和每个后续分段上传请求（[Upload Part](#)）中指定这些标头。对于每个分段上传请求，加密信息必须与您在启动分段上传请求中提供的信息相同。
- POST 操作 - 使用 POST 操作上传对象（请参阅 [POST Object](#)）时，可在表单字段而不是请求标头中提供相同的信息。
- Copy 操作 - 复制对象（请参阅 [PUT Object - Copy](#)）时，您同时具有源对象和目标对象。因此，您需要考虑以下方面：
  - 如果您希望使用具有 AWS 托管密钥的服务器端加密对目标对象加密，则必须提供 `x-amz-server-side-encryption` 请求标头。
  - 如果您希望使用 SSE-C 对目标对象加密，则必须使用上表中描述的三个标头提供加密信息。
  - 如果源对象是使用 SSE-C 加密的，则您必须使用以下标头提供加密密钥信息，以便 Amazon S3 可以解密对象以进行复制。

名称	说明
<code>x-amz-copy-source-server-side-encryption-customer-algorithm</code>	包括此标头以指定 Amazon S3 用于解密源对象的算法。此值必须是 AES256。
<code>x-amz-copy-source-server-side-encryption-customer-key</code>	包括此标头以提供 base64 编码的加密密钥，供 Amazon S3 用于解密源对象。此加密密钥必须是您在创建源对象时为 Amazon S3 提供的加密密钥；否则，Amazon S3 将无法解密对象。
<code>x-amz-copy-source-server-</code>	包括此标头以根据 <a href="#">RFC 1321</a> 提供加密密钥的 base64 编码的 128 位 MD5 摘要。

名称	说明
side-encryption-customer-key-MD5	

## 预签名 URL 和 SSE-C

您可以生成可用于上传新对象、检索现有对象或对象元数据等操作的预签名 URL。预签名 URL 支持 SSE-C，如下所示：

- 在创建预签名 URL 时，您必须在签名计算中使用 `x-amz-server-side-encryption-customer-algorithm` 指定算法。
- 在使用预签名 URL 上传新对象、检索现有对象或仅检索对象元数据时，您必须在您的客户端应用程序中提供所有加密标头。

有关更多信息，请参阅以下主题：

- [使用 AWS SDK for Java 指定使用客户提供的加密密钥的服务器端加密 \(p. 361\)](#)
- [使用适用于 .NET 的 AWS 开发工具包指定具有客户提供的加密密钥的服务器端加密 \(p. 365\)](#)
- [使用 REST API 指定具有客户提供的加密密钥的服务器端加密 \(p. 372\)](#)

## 使用 AWS SDK for Java 指定使用客户提供的加密密钥的服务器端加密

以下示例演示如何为对象请求使用客户提供的密钥的服务器端加密 (SSE-C)。该示例执行将以下操作。每个操作均演示了如何在请求中指定 SSE-C 相关标头：

- 放置对象 — 上传对象并请求使用客户提供的加密密钥的服务器端加密。
- 获取对象 — 下载上一步中上传的对象。在请求中，应提供上传对象时提供的同一加密信息。Amazon S3 需要此信息来解密对象，以便将对象返回给您。
- 获取对象元数据 — 检索对象的元数据。提供创建对象时使用的同一加密信息。
- 复制对象 — 为之前上传的对象创建副本。因为源对象是使用 SSE-C 存储的，因此必须在复制请求中提供其加密信息。默认情况下，仅当您显式请求加密时，Amazon S3 才会为对象的副本加密。此示例指示 Amazon S3 存储使用新的 SSE-C 密钥加密的对象副本。

### Note

本示例显示如何在单个操作中上传对象。当使用分段上传 API 上传大型对象时，应按照此示例中所示的方式提供加密信息。有关使用 AWS SDK for Java 的分段上传的示例，请参阅[使用适用于分段上传的 AWS Java 开发工具包 \(高级别 API\) \(p. 160\)](#) 和[使用适用于分段上传的 AWS Java 开发工具包 \(低级别 API\) \(p. 164\)](#)。

要添加必需的加密信息，请在请求中包含 `SSECustomerKey`。有关 `SSECustomerKey` 类的更多信息，请参阅[使用 SSE-C \(p. 359\)](#)。

有关 SSE-C 的信息，请参阅[通过使用客户提供的加密密钥的服务器端加密 \(SSE-C\) 保护数据 \(p. 359\)](#)。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

### Example

```
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
```

```
import javax.crypto.KeyGenerator;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.GetObjectMetadataRequest;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.services.s3.model.SSECustomerKey;

public class ServerSideEncryptionUsingClientSideEncryptionKey {
    private static SSECustomerKey SSE_KEY;
    private static AmazonS3 S3_CLIENT;
    private static KeyGenerator KEY_GENERATOR;

    public static void main(String[] args) throws IOException, NoSuchAlgorithmException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String keyName = "**** Key name ****";
        String uploadFileName = "**** File path ****";
        String targetKeyName = "**** Target key name ****";

        // Create an encryption key.
        KEY_GENERATOR = KeyGenerator.getInstance("AES");
        KEY_GENERATOR.init(256, new SecureRandom());
        SSE_KEY = new SSECustomerKey(KEY_GENERATOR.generateKey());

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Upload an object.
            uploadObject(bucketName, keyName, new File(uploadFileName));

            // Download the object.
            downloadObject(bucketName, keyName);

            // Verify that the object is properly encrypted by attempting to retrieve it
            // using the encryption key.
            retrieveObjectMetadata(bucketName, keyName);

            // Copy the object into a new object that also uses SSE-C.
            copyObject(bucketName, keyName, targetKeyName);
        }
        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();
        }
    }

    private static void uploadObject(String bucketName, String keyName, File file) {
```

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
file).withSSECustomerKey(SSE_KEY);
S3_CLIENT.putObject(putRequest);
System.out.println("Object uploaded");
}

private static void downloadObject(String bucketName, String keyName) throws
IOException {
    GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName,
keyName).withSSECustomerKey(SSE_KEY);
    S3Object object = S3_CLIENT.getObject(getObjectRequest);

    System.out.println("Object content: ");
    displayTextInputStream(object.getObjectContent());
}

private static void retrieveObjectMetadata(String bucketName, String keyName) {
    GetObjectMetadataRequest getMetadataRequest = new
GetObjectMetadataRequest(bucketName, keyName)

.withSSECustomerKey(SSE_KEY);
    ObjectMetadata objectMetadata = S3_CLIENT.getObjectMetadata(getMetadataRequest);
    System.out.println("Metadata retrieved. Object size: " +
objectMetadata.getContentLength());
}

private static void copyObject(String bucketName, String keyName, String targetKeyName)
throws NoSuchAlgorithmException {
    // Create a new encryption key for target so that the target is saved using SSE-C.
    SSECustomerKey newSSEKey = new SSECustomerKey(KEY_GENERATOR.generateKey());

    CopyObjectRequest copyRequest = new CopyObjectRequest(bucketName, keyName,
bucketName, targetKeyName)
        .withSourceSSECustomerKey(SSE_KEY)
        .withDestinationSSECustomerKey(newSSEKey);

    S3_CLIENT.copyObject(copyRequest);
    System.out.println("Object copied");
}

private static void displayTextInputStream(S3ObjectInputStream input) throws
IOException {
    // Read one line at a time from the input stream and display each line.
    BufferedReader reader = new BufferedReader(new InputStreamReader(input));
    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
    System.out.println();
}
}
```

## 其他使用AWS SDK for Java的包含 SSE-C 的 Amazon S3 操作

上一部分中的示例显示了如何在 PUT、GET、Head 和 Copy 操作中请求使用客户提供的密钥的服务器端加密 (SSE-C)。这一部分介绍支持 SSE-C 的其他 API。

要上传大型对象，您可以使用分段上传 API (请参阅[使用分段上传 API 上传对象 \(p. 155\)](#))。可以使用高级或低级 API 上传大型对象。这些 API 支持在请求中使用与加密相关的标头。

- 当使用高级别 TransferManager API 时，应在 PutObjectRequest 中提供特定于加密的标头 (请参阅[使用适用于分段上传的 AWS Java 开发工具包 \(高级别 API\) \(p. 160\)](#))。
- 当使用低级别 API 时，应提供 InitiateMultipartUploadRequest 中的加密相关信息，后跟每个 UploadPartRequest 中的相同加密信息。不需要在 CompleteMultipartUploadRequest 中提供

任何特定于加密的标头。有关示例，请参阅 [使用适用于分段上传的 AWS Java 开发工具包 \(低级别 API\) \(p. 164\)](#)。

以下示例使用 TransferManager 创建对象并显示如何提供 SSE-C 相关信息。本示例执行以下操作：

- 使用 TransferManager.upload() 方法创建对象。在 PutObjectRequest 实例中，应提供要请求的加密密钥信息。Amazon S3 将使用客户提供的加密密钥对对象进行加密。
- 通过调用 TransferManager.copy() 方法创建对象的副本。该示例指示 Amazon S3 使用新的 SSECustomerKey 对对象副本进行加密。由于源对象是使用 SSE-C 加密的，因此 CopyObjectRequest 还提供了源对象的加密密钥，以便 Amazon S3 可以在复制对象之前解密对象。

### Example

```
import java.io.File;
import java.security.SecureRandom;

import javax.crypto.KeyGenerator;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.SSECustomerKey;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

public class ServerSideEncryptionCopyObjectUsingHLwithSSEC {

    public static void main(String[] args) throws Exception {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String fileToUpload = "**** File path ****";
        String keyName = "**** New object key name ****";
        String targetKeyName = "**** Key name for object copy ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .withCredentials(new ProfileCredentialsProvider())
                .build();
            TransferManager tm = TransferManagerBuilder.standard()
                .withS3Client(s3Client)
                .build();

            // Create an object from a file.
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
new File(fileToUpload));

            // Create an encryption key.
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, new SecureRandom());
            SSECustomerKey sseCustomerEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());

            // Upload the object. TransferManager uploads asynchronously, so this call
            // returns immediately.
            putObjectRequest.setSSECustomerKey(sseCustomerEncryptionKey);
        }
    }
}
```

```
Upload upload = tm.upload(putObjectRequest);

// Optionally, wait for the upload to finish before continuing.
upload.waitForCompletion();
System.out.println("Object created.");

// Copy the object and store the copy using SSE-C with a new key.
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucketName,
keyName, bucketName, targetKeyName);
SSECustomerKey sseTargetObjectEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());
copyObjectRequest.setSourceSSECustomerKey(sseCustomerEncryptionKey);
copyObjectRequest.setDestinationSSECustomerKey(sseTargetObjectEncryptionKey);

// Copy the object. TransferManager copies asynchronously, so this call returns
immediately.
Copy copy = tm.copy(copyObjectRequest);

// Optionally, wait for the upload to finish before continuing.
copy.waitForCompletion();
System.out.println("Copy complete.");
}

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();
}
}
}
```

## 使用适用于 .NET 的 AWS 开发工具包指定具有客户提供的加密密钥的服务器端加密

以下 C# 示例演示了使用客户提供的密钥的服务器端加密 (SSE-C) 的工作方式。该示例执行将以下操作。每个操作均演示了如何在请求中指定 SSE-C 相关标头。

- 放置对象 — 上传对象并请求使用客户提供的加密密钥的服务器端加密。
- 获取对象 — 下载您在上一步中上传的对象。该请求提供了在对象上传时提供的同一加密信息。Amazon S3 需要此信息来解密对象并返回给您。
- 获取对象元数据 — 提供在创建对象以检索对象的元数据时使用的同一加密信息。
- 复制对象 — 创建上传对象的副本。由于源对象是使用 SSE-C 存储的，复制请求必须提供加密信息。默认情况下，Amazon S3 不会加密对象的副本。该代码指示 Amazon S3 通过提供目标的加密相关信息来使用 SSE-C 加密复制的对象。它还会存储目标。

### Note

有关使用分段上传 API 上传大型对象的示例，请参阅[使用适用于 .NET 的 AWS 开发工具包进行分段上传 \(高级别 API\) \(p. 169\)](#)和[使用适用于 .NET 的 AWS 开发工具包 \(低级别 API\) 执行分段上传 \(p. 175\)](#)。

有关 SSE-C 的信息，请参阅[通过使用客户提供的加密密钥的服务器端加密 \(SSE-C\) 保护数据 \(p. 359\)](#)。有关创建和测试有效示例的信息，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

## Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSEClientEncryptionKeyObjectOperationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for new object created ***";
        private const string copyTargetKeyName = "*** key name for object copy ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ObjectOpsUsingClientEncryptionKeyAsync().Wait();
        }

        private static async Task ObjectOpsUsingClientEncryptionKeyAsync()
        {
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // 1. Upload the object.
                PutObjectRequest putObjectRequest = await UploadObjectAsync(base64Key);
                // 2. Download the object and verify that its contents matches what you
uploaded.
                await DownloadObjectAsync(base64Key, putObjectRequest);
                // 3. Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(base64Key);
                // 4. Copy both the source and target objects using server-side encryption
with
                //      a customer-provided encryption key.
                await CopyObjectAsync(aesEncryption, base64Key);
            }
            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<PutObjectRequest> UploadObjectAsync(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;
}
private static async Task DownloadObjectAsync(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");

        // Assert.AreEqual(putObjectRequest.ContentBody, content);
        // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getResponse.ServerSideEncryptionCustomerMethod);
    }
}
private static async Task GetObjectMetadataAsync(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);
// Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
}
private static async Task CopyObjectAsync(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);
}
}
```

## 其他 Amazon S3 操作和 SSE-C

上一部分中的示例显示了如何在 PUT、GET、Head 和 Copy 操作中请求使用客户提供的密钥的服务器端加密 (SSE-C)。本节介绍支持 SSE-C 的其他 Amazon S3 API。

要上传大型对象，您可以使用分段上传 API (请参阅[使用分段上传 API 上传对象 \(p. 155\)](#))。适用于 .NET 的 AWS 开发工具包提供了高级或低级 API 来上传大型对象。这些 API 支持在请求中使用与加密相关的标头。

- 当使用高级 Transfer-Utility API 时，可以在 TransferUtilityUploadRequest 中提供特定于加密的标头，如下所示。有关代码示例，请参阅[使用适用于 .NET 的 AWS 开发工具包进行分段上传 \(高级别 API\) \(p. 169\)](#)。

```
TransferUtilityUploadRequest request = new TransferUtilityUploadRequest()
{
    FilePath = filePath,
    BucketName = existingBucketName,
    Key = keyName,
    // Provide encryption information.
    ServerSideEncryptionCustomerMethod = ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};
```

- 当使用低级 API 时，可以在启动分段上传请求中提供加密相关的信息，并在后续的分段上传请求中提供相同加密信息。不需要在完成的分段上传请求中提供任何特定于加密的标头。有关示例，请参阅[使用适用于 .NET 的 AWS 开发工具包 \(低级别 API\) 执行分段上传 \(p. 175\)](#)。

下面是一个低级分段上传示例，该示例复制一个现有大型对象。在本示例中，要复制的对象使用 SSE-C 存储在 Amazon S3，并且您希望使用 SSE-C 保存目标对象。因此，您执行以下操作：

- 通过提供加密密钥和相关信息启动分段上传请求。
- 在 CopyPartRequest 中提供源和目标对象加密密钥及相关信息。

- 通过检索对象元数据获取要复制的源对象的大小。
- 以 5 MB 大小的分段上传对象。

### Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSECLowLevelMPUcopyObjectTest
    {
        private const string existingBucketName = "**** bucket name ****";
        private const string sourceKeyName      = "**** source object key name ****";
        private const string targetKeyName     = "**** key name for the target object
****";
        private const string filePath          = @"**** file path ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CopyObjClientEncryptionKeyAsync().Wait();
        }

        private static async Task CopyObjClientEncryptionKeyAsync()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            await CreateSampleObjUsingClientEncryptionKeyAsync(base64Key, s3Client);

            await CopyObjectAsync(s3Client, base64Key);
        }
        private static async Task CopyObjectAsync(IAmazonS3 s3Client, string base64Key)
        {
            List<CopyPartResponse> uploadResponses = new List<CopyPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = targetKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
await s3Client.InitiateMultipartUploadAsync(initiateRequest);
```

```
// 2. Upload Parts.
long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
long firstByte = 0;
long lastByte = partSize;

try
{
    // First find source object size. Because object is stored encrypted with
    // customer provided key you need to provide encryption information in
    // your request.
    GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest()
{
    BucketName = existingBucketName,
    Key = sourceKeyName,
    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key // " * **source
object encryption key ***"
};

    GetObjectMetadataResponse getObjectMetadataResponse = await
s3Client.GetObjectMetadataAsync(getObjectMetadataRequest);

    long filePosition = 0;
    for (int i = 1; filePosition < getObjectMetadataResponse.ContentLength; i
++)
{
    CopyPartRequest copyPartRequest = new CopyPartRequest
{
    UploadId = initResponse.UploadId,
    // Source.
    SourceBucket = existingBucketName,
    SourceKey = sourceKeyName,
    // Source object is stored using SSE-C. Provide encryption
information.
    CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    CopySourceServerSideEncryptionCustomerProvidedKey =
base64Key, //***source object encryption key ***",
    FirstByte = firstByte,
    // If the last part is smaller then our normal part size then use
the remaining size.
    LastByte = lastByte > getObjectMetadataResponse.ContentLength ?
        getObjectMetadataResponse.ContentLength - 1 : lastByte,

    // Target.
    DestinationBucket = existingBucketName,
    DestinationKey = targetKeyName,
    PartNumber = i,
    // Encryption information for the target object.
    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key
};
    uploadResponses.Add(await s3Client.CopyPartAsync(copyPartRequest));
    filePosition += partSize;
    firstByte += partSize;
    lastByte += partSize;
}

// Step 3: complete.
CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
{
    BucketName = existingBucketName,
```

```
        Key = targetKeyName,
        UploadId = initResponse.UploadId,
    };
    completeRequest.AddPartETags(uploadResponses);

    CompleteMultipartUploadResponse completeUploadResponse =
        await s3Client.CompleteMultipartUploadAsync(completeRequest);
}
catch (Exception exception)
{
    Console.WriteLine("Exception occurred: {0}", exception.Message);
    AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = targetKeyName,
    UploadId = initResponse.UploadId
};
    s3Client.AbortMultipartUpload(abortMPURequest);
}
}
private static async Task CreateSampleObjUsingClientEncryptionKeyAsync(string
base64Key, IAmazonS3 s3Client)
{
    // List to store upload part responses.
    List<UploadPartResponse> uploadResponses = new List<UploadPartResponse>();

    // 1. Initialize.
    InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = sourceKeyName,
    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key
};

    InitiateMultipartUploadResponse initResponse =
        await s3Client.InitiateMultipartUploadAsync(initiateRequest);

    // 2. Upload Parts.
    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 s3Client.UploadPartAsync(uploadRequest));
        }
    }
}
```

```
        filePosition += partSize;
    }

    // Step 3: complete.
    CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = sourceKeyName,
    UploadId = initResponse.UploadId,
    //PartETags = new List<PartETag>(uploadResponses)

};

completeRequest.AddPartETags(uploadResponses);

CompleteMultipartUploadResponse completeUploadResponse =
    await s3Client.CompleteMultipartUploadAsync(completeRequest);

}

catch (Exception exception)
{
    Console.WriteLine("Exception occurred: {0}", exception.Message);
    AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = sourceKeyName,
    UploadId = initResponse.UploadId
};
    await s3Client.AbortMultipartUploadAsync(abortMPURequest);
}
}
}
```

## 使用 REST API 指定具有客户提供的加密密钥的服务器端加密

以下 Amazon S3 REST API 支持与具有客户提供的加密密钥的服务器端加密相关的标头。有关这些标头的更多信息，请参阅 [使用 SSE-C \(p. 359\)](#)。

- [GET Object](#)
- [HEAD Object](#)
- [PUT Object](#)
- [PUT Object – Copy](#)
- [POST Object](#)
- [开始分段上传](#)
- [上传分段](#)
- [上传分段 – 复制](#)

## 使用客户端加密保护数据

客户端加密 是在将数据发送到 Amazon S3 之前加密数据的行为。要启用客户端加密，您可以选择以下方法：

- 使用 AWS KMS 托管客户主密钥
- 使用客户端主密钥

以下 AWS 开发工具包支持客户端加密：

- 适用于 .NET 的 AWS 开发工具包
- 适用于 Go 的 AWS 开发工具包
- AWS SDK for Java
- 适用于 PHP 的 AWS 开发工具包
- 适用于 Ruby 的 AWS 开发工具包

## 选项 1：使用 AWS KMS 托管客户主密钥 (CMK)

当使用 AWS KMS 托管客户主密钥启用客户端数据加密时，应提供 AWS KMS 客户主密钥 ID (CMK ID)。

- 上传对象时 — 通过使用 CMK ID，客户端首先将请求发送到 AWS Key Management Service (AWS KMS) 以获取可用于加密对象数据的密钥。AWS KMS 将返回一个随机生成的数据加密密钥的两个版本：
  - 客户端用于加密对象数据的纯文本版本
  - 客户端将作为对象元数据上传到 Amazon S3 的同一数据加密密钥的密码 blob

### Note

客户端将为其上传的每个对象获取一个唯一的数据加密密钥。

- 下载对象时 — 客户端首先从 Amazon S3 下载加密的对象以及作为对象元数据存储的数据加密密钥的密码 blob 版本。然后，客户端将密码 blob 发送到 AWS KMS 以获取密钥的纯文本版本，以便让客户端解密对象数据。

有关 AWS KMS 的更多信息，请参阅[什么是 AWS Key Management Service？](#) (在 AWS Key Management Service Developer Guide 中)。

### Example

以下示例通过将 AWS KMS 与 AWS SDK for Java 结合使用来将对象上传到 Amazon S3。该示例在将数据上传到 Amazon S3 之前，使用 KMS 托管客户主密钥 (CMK) 在客户端上加密数据。如果您已经有一个 CMK，则可通过在示例代码中指定 `kms_cmk_id` 变量的值来使用该 CMK。如果没有 CMK，或需要另一个 CMK，则可以通过 Java API 生成一个。该示例演示如何生成 CMK。

有关密钥材料的更多信息，请参阅[在 AWS Key Management Service \(AWS KMS\) 中导入密钥材料](#)。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.CreateKeyResult;
import com.amazonaws.services.s3.AmazonS3Encryption;
import com.amazonaws.services.s3.AmazonS3EncryptionClientBuilder;
import com.amazonaws.services.s3.model.CryptoConfiguration;
import com.amazonaws.services.s3.model.KMSEncryptionMaterialsProvider;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

public class UploadObjectKMSKey {

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

```
String bucketName = "*** Bucket name ***";
String keyName = "*** Object key name ***";
String clientRegion = "*** Client region ***";
String kms_cmk_id = "***AWS KMS customer master key ID***";
int readChunkSize = 4096;

try {
    // Optional: If you don't have a KMS key (or need another one),
    // create one. This example creates a key with AWS-created
    // key material.
    AWSKMS kmsClient = AWSKMSClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(clientRegion)
        .build();
    CreateKeyResult keyResult = kmsClient.createKey();
    kms_cmk_id = keyResult.getKeyMetadata().getKeyId();

    // Create the encryption client.
    KMSEncryptionMaterialsProvider materialProvider = new
    KMSEncryptionMaterialsProvider(kms_cmk_id);
    CryptoConfiguration cryptoConfig = new CryptoConfiguration()
        .withAwsKmsRegion(RegionUtils.getRegion(clientRegion));
    AmazonS3Encryption encryptionClient =
    AmazonS3EncryptionClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withEncryptionMaterials(materialProvider)
        .withCryptoConfiguration(cryptoConfig)
        .withRegion(clientRegion).build();

    // Upload an object using the encryption client.
    String origContent = "S3 Encrypted Object Using KMS-Managed Customer Master
Key.";
    int origContentLength = origContent.length();
    encryptionClient.putObject(bucketName, keyName, origContent);

    // Download the object. The downloaded object is still encrypted.
    S3Object downloadedObject = encryptionClient.getObject(bucketName, keyName);
    S3ObjectInputStream input = downloadedObject.getObjectContent();

    // Decrypt and read the object and close the input stream.
    byte[] readBuffer = new byte[readChunkSize];
    ByteArrayOutputStream baos = new ByteArrayOutputStream(readChunkSize);
    int bytesRead = 0;
    int decryptedContentLength = 0;

    while ((bytesRead = input.read(readBuffer)) != -1) {
        baos.write(readBuffer, 0, bytesRead);
        decryptedContentLength += bytesRead;
    }
    input.close();

    // Verify that the original and decrypted contents are the same size.
    System.out.println("Original content length: " + origContentLength);
    System.out.println("Decrypted content length: " + decryptedContentLength);
}
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();
}
}
```

}

## 选项 2：使用客户端主密钥

本节演示如何使用客户端主密钥执行客户端数据加密。

### Important

客户端主密钥和未加密的数据从来不会发送到 AWS。务必安全地管理加密密钥。如果您丢失了加密密钥，将无法解密数据。

以下是具体工作原理：

- 上传对象时 — 将客户端主密钥提供给 Amazon S3 加密客户端。该客户端仅使用该主密钥来加密客户端随机生成的数据加密密钥。该过程的工作方式如下所示：
  1. Amazon S3 加密客户端在本地生成一个一次性对称密钥（也称为“数据加密密钥”或“数据密钥”）。它使用数据密钥加密单个 Amazon S3 对象的数据。该客户端将为每个对象生成一个单独的数据密钥。
  2. 该客户端使用您提供的主密钥来加密数据加密密钥。客户端会将加密的数据密钥及其材料说明作为对象元数据的一部分上传。该客户端利用材料描述来确定要用于解密的客户端主密钥。
  3. 该客户端将加密数据上传到 Amazon S3 并在 Amazon S3 中将加密数据密钥保存为对象元数据 (x-amz-meta-x-amz-key)。
- 下载对象时 — 该客户端从 Amazon S3 下载加密的对象。通过使用对象元数据中的材料说明，该客户端将确定要用于解密数据密钥的主密钥。该客户端将使用该主密钥解密数据密钥，然后使用该数据密钥对对象进行解密。

您提供的客户端主密钥可以是对称密钥，也可以是公有/私有密钥对。以下示例演示如何使用这两种密钥。

有关更多信息，请参阅[使用 AWS SDK for Java 和 Amazon S3 进行的客户端数据加密](#)。

### Note

首次使用加密 API 时，如果您收到密码加密错误消息，则您的 JDK 版本可能带有一个 Java Cryptography Extension (JCE) 法律辖区策略文件，该文件将加密和解密转换的最大密钥长度限制为 128 位。AWS 开发工具包要求的最大密钥长度为 256 位。要检查您的最大密钥长度，请使用 `javax.crypto.Cipher` 类的 `getMaxAllowedKeyLength()` 方法。要取消密钥长度限制，请安装位于[Java SE 下载页面](#)上的 Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File。

### Example

以下示例演示如何执行以下任务：

- 生成 256 位 AES 密钥
- 在文件系统中保存和加载 AES 密钥
- 将数据发送到 Amazon S3 之前在客户端上使用 AES 密钥加密数据
- 使用 AES 密钥解密从 Amazon S3 接收的数据
- 验证解密数据是否与原始数据相同

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.X509EncodedKeySpec;

import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3EncryptionClientBuilder;
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.StaticEncryptionMaterialsProvider;

public class S3ClientSideEncryptionSymMasterKey {

    public static void main(String[] args) throws Exception {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";
        String objectKeyName = "**** Object key name ****";
        String masterKeyDir = System.getProperty("java.io.tmpdir");
        String masterKeyName = "secret.key";

        // Generate a symmetric 256-bit AES key.
        KeyGenerator symKeyGenerator = KeyGenerator.getInstance("AES");
        symKeyGenerator.init(256);
        SecretKey symKey = symKeyGenerator.generateKey();

        // To see how it works, save and load the key to and from the file system.
        saveSymmetricKey(masterKeyDir, masterKeyName, symKey);
        symKey = loadSymmetricAESKey(masterKeyDir, masterKeyName, "AES");

        try {
            // Create the Amazon S3 encryption client.
            EncryptionMaterials encryptionMaterials = new EncryptionMaterials(symKey);
            AmazonS3 s3EncryptionClient = AmazonS3EncryptionClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withEncryptionMaterials(new
            StaticEncryptionMaterialsProvider(encryptionMaterials))
                .withRegion(clientRegion)
                .build();

            // Upload a new object. The encryption client automatically encrypts it.
            byte[] plaintext = "S3 Object Encrypted Using Client-Side Symmetric Master
Key.".getBytes();
            s3EncryptionClient.putObject(new PutObjectRequest(bucketName,
                    objectKeyName,
                    new
            ByteArrayInputStream(plaintext),
                    new ObjectMetadata()));

            // Download and decrypt the object.
            S3Object downloadedObject = s3EncryptionClient.getObject(bucketName,
                    objectKeyName);
            byte[] decrypted =
            com.amazonaws.util.IOUtils.toByteArray(downloadedObject.getObjectContent());

            // Verify that the data that you downloaded is the same as the original data.
            System.out.println("Plaintext: " + new String(plaintext));
            System.out.println("Decrypted text: " + new String(decrypted));
        }
    }
}
```

```

        }
        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();
        }
    }

    private static void saveSymmetricKey(String masterKeyDir, String masterKeyName,
    SecretKey secretKey) throws IOException {
        X509EncodedKeySpec x509EncodedKeySpec = new
        X509EncodedKeySpec(secretKey.getEncoded());
        FileOutputStream keyOutputStream = new FileOutputStream(masterKeyDir +
        File.separator + masterKeyName);
        keyOutputStream.write(x509EncodedKeySpec.getEncoded());
        keyOutputStream.close();
    }

    private static SecretKey loadSymmetricAESKey(String masterKeyDir, String masterKeyName,
    String algorithm)
        throws IOException, NoSuchAlgorithmException, InvalidKeySpecException,
    InvalidKeyException {
        // Read the key from the specified file.
        File keyFile = new File(masterKeyDir + File.separator + masterKeyName);
        FileInputStream keyInputStream = new FileInputStream(keyFile);
        byte[] encodedPrivateKey = new byte[(int) keyFile.length()];
        keyInputStream.read(encodedPrivateKey);
        keyInputStream.close();

        // Reconstruct and return the master key.
        return new SecretKeySpec(encodedPrivateKey, "AES");
    }
}

```

以下示例演示如何执行以下任务：

- 生成 1024 位 RSA 密钥对
- 在文件系统中保存和加载 RSA 密钥
- 将数据发送到 Amazon S3 之前在客户端上使用 RSA 密钥加密数据
- 使用 RSA 密钥解密从 Amazon S3 接收的数据
- 验证解密数据是否与原始数据相同

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.spec.InvalidKeySpecException;

```

```
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3EncryptionClientBuilder;
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.StaticEncryptionMaterialsProvider;
import com.amazonaws.util.IOUtils;

public class S3ClientSideEncryptionAsymmetricMasterKey {

    public static void main(String[] args) throws Exception {
        String clientRegion = "*** Client region ***";
        String bucketName = "*** Bucket name ***";
        String objectKeyName = "*** Key name ***";
        String rsaKeyDir = System.getProperty("java.io.tmpdir");
        String publicKeyName = "public.key";
        String privateKeyName = "private.key";

        // Generate a 1024-bit RSA key pair.
        KeyPairGenerator keyGenerator = KeyPairGenerator.getInstance("RSA");
        keyGenerator.initialize(1024, new SecureRandom());
        KeyPair origKeyPair = keyGenerator.generateKeyPair();

        // To see how it works, save and load the key pair to and from the file system.
        saveKeyPair(rsaKeyDir, publicKeyName, privateKeyName, origKeyPair);
        KeyPair keyPair = loadKeyPair(rsaKeyDir, publicKeyName, privateKeyName, "RSA");

        try {
            // Create the encryption client.
            EncryptionMaterials encryptionMaterials = new EncryptionMaterials(keyPair);
            AmazonS3 s3EncryptionClient = AmazonS3EncryptionClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withEncryptionMaterials(new
                    StaticEncryptionMaterialsProvider(encryptionMaterials))
                .withRegion(clientRegion)
                .build();

            // Create a new object.
            byte[] plaintext = "S3 Object Encrypted Using Client-Side Asymmetric Master
Key.".getBytes();
            S3Object object = new S3Object();
            object.setKey(objectKeyName);
            object.setObjectContent(new ByteArrayInputStream(plaintext));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentLength(plaintext.length);

            // Upload the object. The encryption client automatically encrypts it.
            PutObjectRequest putRequest = new PutObjectRequest(bucketName,
                object.getKey(),
                object.getObjectContent(),
                metadata);
            s3EncryptionClient.putObject(putRequest);

            // Download and decrypt the object.
            S3Object downloadedObject = s3EncryptionClient.getObject(bucketName,
                object.getKey());
            byte[] decrypted = IOUtils.toByteArray(downloadedObject.getObjectContent());

            // Verify that the data that you downloaded is the same as the original data.
        }
    }
}
```

```
        System.out.println("Plaintext: " + new String(plaintext));
        System.out.println("Decrypted text: " + new String(decrypted));
    }
    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();
    }
}

private static void saveKeyPair(String dir,
                                String publicKeyName,
                                String privateKeyName,
                                KeyPair keyPair) throws IOException {
    PrivateKey privateKey = keyPair.getPrivate();
    PublicKey publicKey = keyPair.getPublic();

    // Write the public key to the specified file.
    X509EncodedKeySpec x509EncodedKeySpec = new
X509EncodedKeySpec(publicKey.getEncoded());
    FileOutputStream publicKeyOutputStream = new FileOutputStream(dir + File.separator
+ publicKeyName);
    publicKeyOutputStream.write(x509EncodedKeySpec.getEncoded());
    publicKeyOutputStream.close();

    // Write the private key to the specified file.
    PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new
PKCS8EncodedKeySpec(privateKey.getEncoded());
    FileOutputStream privateKeyOutputStream = new FileOutputStream(dir + File.separator
+ privateKeyName);
    privateKeyOutputStream.write(pkcs8EncodedKeySpec.getEncoded());
    privateKeyOutputStream.close();
}

private static KeyPair loadKeyPair(String dir,
                                    String publicKeyName,
                                    String privateKeyName,
                                    String algorithm)
throws IOException, NoSuchAlgorithmException, InvalidKeySpecException {
    // Read the public key from the specified file.
    File publicKeyFile = new File(dir + File.separator + publicKeyName);
    FileInputStream publicKeyInputStream = new FileInputStream(publicKeyFile);
    byte[] encodedPublicKey = new byte[(int) publicKeyFile.length()];
    publicKeyInputStream.read(encodedPublicKey);
    publicKeyInputStream.close();

    // Read the private key from the specified file.
    File privateKeyFile = new File(dir + File.separator + privateKeyName);
    FileInputStream privateKeyInputStream = new FileInputStream(privateKeyFile);
    byte[] encodedPrivateKey = new byte[(int) privateKeyFile.length()];
    privateKeyInputStream.read(encodedPrivateKey);
    privateKeyInputStream.close();

    // Convert the keys into a key pair.
    KeyFactory keyFactory = KeyFactory.getInstance(algorithm);
    X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(encodedPublicKey);
    PublicKey publicKey = keyFactory.generatePublic(publicKeySpec);

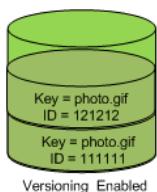
    PKCS8EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(encodedPrivateKey);
    PrivateKey privateKey = keyFactory.generatePrivate(privateKeySpec);
```

```
        return new KeyPair(publicKey, privateKey);
    }
```

## 使用版本控制

版本控制是在相同的存储桶中保留对象的多个变量的方法。对于 Amazon S3 桶中存储的每个对象，您可以使用版本控制功能来保存、检索和还原它们的各个版本。使用版本控制能够轻松从用户意外操作和应用程序故障中恢复数据。

例如，在一个存储桶中，您可以拥有两个具有相同键的对象，但版本 ID 却不同，例如 `photo.gif` (版本 111111) 和 `photo.gif` (版本 121212)。



通过启用了版本控制的存储桶可以恢复因意外删除或覆盖操作而失去的对象。例如：

- 如果删除对象（而不是永久移除它），则 Amazon S3 会插入删除标记，该标记将成为当前对象版本。您始终可以恢复以前的版本。有关更多信息，请参阅 [删除数据元版本 \(p. 392\)](#)。
- 如果覆盖对象，则会导致存储桶中出现新的对象版本。您始终可以恢复以前的版本。

### Important

如果您在不受版本控制的存储桶中具有对象到期生命周期策略，并且希望在启用版本控制时保持相同的永久删除行为，则必须添加非当前版本到期策略。非当前版本到期生命周期策略将管理在受版本控制的存储桶中删除非当前对象版本的行为。（启用版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。）有关更多信息，请参阅 [如何为 S3 存储桶创建生命周期策略？](#)（在 Amazon Simple Storage Service 控制台用户指南中）。

存储桶可处于以下三种状态之一：非版本化（默认）、启用版本控制或暂停版本控制。

### Important

一旦您对存储桶启用了版本控制，它将无法返回到无版本控制状态。但是，您可以在该存储桶上暂停版本控制。

版本控制状态将应用到该存储桶中的所有（不是某些）对象。您第一次对存储桶启用版本控制后，该存储桶中的对象将在之后一直受版本控制，并具有唯一的版本 ID。请注意以下几点：

- 在您设置版本控制状态之前存储在存储桶中的对象具有版本 ID `null`。启用版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理这些对象的方式。有关更多信息，请参阅 [在启用了版本控制的存储桶中管理对象 \(p. 384\)](#)。
- 存储桶拥有者（或任何具有适当权限的用户）可以暂停版本控制以停止累积对象版本。暂停版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理对象的方式。有关更多信息，请参阅 [管理已暂停版本控制的存储桶中的对象 \(p. 398\)](#)。

## 如何对存储桶配置版本控制

可以使用以下任何方法配置存储桶版本控制：

- 使用 Amazon S3 控制台配置版本控制。
- 使用 AWS 开发工具包以编程方式配置版本控制。

控制台和开发工具包都调用 Amazon S3 提供的 REST API 来管理版本控制。

Note

如果需要，也可以直接从代码中调用 Amazon S3 REST API。但是，这可能会比较繁琐，因为需要您编写代码对请求进行身份验证。

您创建的每个存储桶都具有关联的 versioning 子资源（请参阅[存储桶配置选项 \(p. 51\)](#)）。默认情况下，您的存储桶无版本控制，因而 versioning 子资源会存储空的版本控制配置。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

要启用版本控制，请向 Amazon S3 发送一个请求，在请求中指定包含状态的版本控制配置。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

要暂停版本控制，请将状态值设置为 Suspended。

存储桶拥有者（创建存储桶的 AWS 账户，即根账户）和授权用户可以配置存储桶的版本控制状态。有关许可的更多信息，请参阅[管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)。

有关配置版本控制的示例，请参阅[启用存储桶版本控制的示例 \(p. 382\)](#)。

## MFA 删 除

您可以选择将存储桶配置为启用 MFA（多重验证）删除（这需要对以下任一操作额外进行身份验证），从而另外增加一层安全性：

- 更改存储桶的版本控制状态
- 永久删除对象版本

MFA 删除需要结合使用两种形式的身份验证：

- 您的安全凭证
- 在已批准的身份验证设备上显示的有效序列号、空格和 6 位代码的组合

MFA 删除因而因此可在某些情况下（例如在安全凭证受损时）提供更高的安全性。

要启用或禁用 MFA 删除，请使用对存储桶配置版本控制时所用的相同 API。Amazon S3 将 MFA 删除配置存储在用于存储存储桶版本控制状态的相同 versioning 子资源中。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

要使用 MFA 删除，您可以使用硬件或虚拟 MFA 设备来生成身份验证代码。以下示例显示了在硬件设备上显示的生成的身份验证代码。



#### Note

MFA 删除和受 MFA 保护的 API 访问的功能旨在提供对不同方案的保护。您可以在存储桶上配置 MFA 删除，以确保存储桶中的数据无法被意外删除。受 MFA 保护的 API 访问用于在访问敏感的 Amazon S3 资源时，强制执行其他身份验证因素（MFA 代码）。您可以要求使用 MFA 创建的临时凭证来完成针对这些 Amazon S3 资源的任何操作。有关示例，请参阅[添加存储桶策略以请求 MFA \(p. 308\)](#)。

有关如何购买和激活身份验证设备的更多信息，请参阅 <https://aws.amazon.com/iam/details/mfa/>。

#### Note

该存储桶的拥有者（创建存储桶的 AWS 账户，根账户）和所有授权 IAM 用户都可以启用版本控制，但只有存储桶拥有者（根账户）才能启用 MFA 删除。

## 相关主题

有关更多信息，请参阅以下主题：

- [启用存储桶版本控制的示例 \(p. 382\)](#)
- [在启用了版本控制的存储桶中管理对象 \(p. 384\)](#)
- [管理已暂停版本控制的存储桶中的对象 \(p. 398\)](#)
- [启用版本控制后，Amazon S3 对存储桶请求的 HTTP 503 响应显著增加 \(p. 502\)](#)

## 启用存储桶版本控制的示例

### 主题

- [使用 Amazon S3 控制台 \(p. 382\)](#)
- [使用适用于 Java 的 AWS 开发工具包 \(p. 382\)](#)
- [使用适用于 .NET 的 AWS 开发工具包 \(p. 383\)](#)
- [使用其他 AWS 开发工具包 \(p. 384\)](#)

此部分提供对存储桶启用版本控制的示例。这些示例首先对存储桶启用版本控制，然后检索版本控制状态。有关简介，请参阅[使用版本控制 \(p. 380\)](#)。

## 使用 Amazon S3 控制台

有关使用 Amazon S3 控制台在存储桶上启用版本控制的更多信息，请参阅[如何为 S3 存储桶启用或暂停版本控制？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

## 使用适用于 Java 的 AWS 开发工具包

### Example

有关如何创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;

public class BucketVersioningConfigurationExample {
    public static String bucketName = "*** bucket name ***";
    public static AmazonS3Client s3Client;

    public static void main(String[] args) throws IOException {
        s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        s3Client.setRegion(Region.getRegion(Regions.US_EAST_1));
        try {

            // 1. Enable versioning on the bucket.
            BucketVersioningConfiguration configuration =
                new BucketVersioningConfiguration().withStatus("Enabled");

            SetBucketVersioningConfigurationRequest setBucketVersioningConfigurationRequest =
                new SetBucketVersioningConfigurationRequest(bucketName, configuration);

            s3Client.setBucketVersioningConfiguration(setBucketVersioningConfigurationRequest);

            // 2. Get bucket versioning configuration information.
            BucketVersioningConfiguration conf =
                s3Client.getBucketVersioningConfiguration(bucketName);
            System.out.println("bucket versioning configuration status: " + conf.getStatus());

        } catch (AmazonS3Exception amazonS3Exception) {
            System.out.format("An Amazon S3 error occurred. Exception: %s",
                amazonS3Exception.toString());
        } catch (Exception ex) {
            System.out.format("Exception: %s", ex.toString());
        }
    }
}
```

## 使用适用于 .NET 的 AWS 开发工具包

有关如何创建和测试有效示例的信息，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

### Example

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

namespace s3.amazon.com.docsamples
{
    class BucketVersioningConfiguration
    {
        static string bucketName = "*** bucket name ***";

        public static void Main(string[] args)
        {
            using (var client = new AmazonS3Client(Amazon.RegionEndpoint.USEast1))
            {
                try
                {
                    EnableVersioningOnBucket(client);
                    string bucketVersioningStatus =
                    RetrieveBucketVersioningConfiguration(client);
                }
                catch (AmazonS3Exception amazonS3Exception)

```

```
        {
            if (amazonS3Exception.ErrorCode != null &&
                (amazonS3Exception.ErrorCode.Equals("InvalidAccessKeyId") ||
                 amazonS3Exception.ErrorCode.Equals("InvalidSecurity")))
            {
                Console.WriteLine("Check the provided AWS Credentials.");
                Console.WriteLine(
                    "To sign up for service, go to http://aws.amazon.com/s3");
            }
            else
            {
                Console.WriteLine(
                    "Error occurred. Message:{0} when listing objects",
                    amazonS3Exception.Message);
            }
        }
    }

    Console.WriteLine("Press any key to continue...");
    Console.ReadKey();
}

static void EnableVersioningOnBucket(IAmazonS3 client)
{
    PutBucketVersioningRequest request = new PutBucketVersioningRequest
    {
        BucketName = bucketName,
        VersioningConfig = new S3BucketVersioningConfig
        {
            Status = VersionStatus.Enabled
        }
    };

    PutBucketVersioningResponse response = client.PutBucketVersioning(request);
}

static string RetrieveBucketVersioningConfiguration(IAmazonS3 client)
{
    GetBucketVersioningRequest request = new GetBucketVersioningRequest
    {
        BucketName = bucketName
    };

    GetBucketVersioningResponse response = client.GetBucketVersioning(request);
    return response.VersioningConfig.Status;
}
}
```

## 使用其他 AWS 开发工具包

有关使用其他 AWS 开发工具包的信息，请参阅[示例代码和库](#)。

# 在启用了版本控制的存储桶中管理对象

## 主题

- [将对象添加到已启用版本控制的存储桶 \(p. 385\)](#)
- [列出启用版本控制的存储桶中的对象 \(p. 386\)](#)
- [检索数据元版本 \(p. 390\)](#)

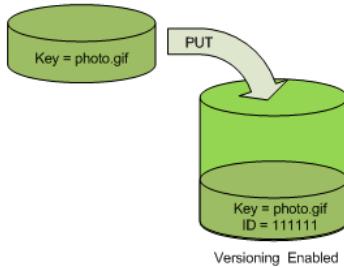
- [删除数据元版本 \(p. 392\)](#)
- [转换对象版本 \(p. 396\)](#)
- [还原早期版本 \(p. 396\)](#)
- [受版本控制的对象的权限 \(p. 397\)](#)

在您设置版本控制状态之前存储在存储桶中的对象的版本 ID 为 `null`。启用版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理这些对象的方式。此部分中的主题说明在启用了版本控制的存储桶中进行的各种对象操作。

## 将对象添加到已启用版本控制的存储桶

在存储桶上启用版本控制之后，Amazon S3 会为存储桶中存储的每个对象自动添加唯一的版本 ID (使用 `PUT`、`POST` 或 `COPY`)。

如下图所示，在启用了版本控制的存储桶中添加对象时，Amazon S3 会为该对象添加唯一的版本 ID。



### 主题

- [使用控制台 \(p. 385\)](#)
- [使用 AWS 开发工具包 \(p. 385\)](#)
- [使用 REST API \(p. 385\)](#)

## 使用控制台

有关说明，请参阅[如何将对象上传到 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南中)。

## 使用 AWS 开发工具包

有关使用适用于 Java、.NET 和 PHP 的 AWS 开发工具包上传对象的示例，请参阅[上传对象 \(p. 150\)](#)。在无版本控制和启用版本控制的存储桶中上传对象的示例是相同的，只是对于启用版本控制的存储桶，Amazon S3 会分配版本号。否则，版本号为空。

有关使用其他 AWS 开发工具包的信息，请参阅[示例代码和库](#)。

## 使用 REST API

### 将对象添加到已启用版本控制的存储桶

1	使用 <code>PUT Bucket versioning</code> 请求在存储桶上启用版本控制。有关更多信息，请参阅 <a href="#">PUT Bucket versioning</a> 。
2	发送 <code>PUT</code> 、 <code>POST</code> 或 <code>COPY</code> 请求，以在存储桶中存储对象。

在启用版本控制的存储桶中添加对象时，Amazon S3 将在 `x-amz-versionid` 响应标头中返回该对象的版本 ID，例如：

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

#### Note

对于存储和传输的每个对象版本，都适用正常 Amazon S3 费率。对象的每个版本都是完整的对象；它并非只是与上一版本有所不同。因此，如果您存储了三个版本的对象，则会收取您三个对象的费用。

#### Note

Amazon S3 分配的版本 ID 值是 URL 安全的（可以包含在 URI 中）。

## 列出启用版本控制的存储桶中的对象

### 主题

- [使用控制台 \(p. 386\)](#)
- [使用 AWS 开发工具包 \(p. 386\)](#)
- [使用 REST API \(p. 389\)](#)

此部分提供从启用版本控制的存储桶列出对象版本的示例。Amazon S3 将对象版本信息存储在与存储桶关联的 versions 子资源中（请参阅[存储桶配置选项 \(p. 51\)](#)）。

### 使用控制台

有关使用 Amazon S3 控制台列出对象版本的信息，请参阅[如何查看 S3 对象的版本？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

### 使用 AWS 开发工具包

此部分中的示例说明如何从启用了版本控制的存储桶中检索对象列表。每个请求最多返回 1000 个版本，除非您指定一个较小的数字。如果存储桶包含的版本数超过此限制，请发送一系列请求来检索所有版本的列表。以“页面”形式返回结果的过程称为分页。为了说明分页的工作原理，该示例将每个响应限制为两个对象版本。在检索第一页结果后，每个示例将检查以确定版本列表是否已被截断。如果是，示例将继续检索页面，直至检索到所有版本。

#### Note

以下示例还使用未启用版本控制的存储桶或用于没有单独版本的对象。在这些情况下，Amazon S3 将返回版本 ID 为 null 的对象列表。

有关使用其他 AWS 开发工具包的信息，请参阅[示例代码和库](#)。

### 使用 AWS SDK for Java

有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

#### Example

```
import java.io.IOException;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;
```

```
public class ListKeysVersioningEnabledBucket {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String bucketName = "**** Bucket name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Retrieve the list of versions. If the bucket contains more versions
            // than the specified maximum number of results, Amazon S3 returns
            // one page of results per request.
            ListVersionsRequest request = new ListVersionsRequest()
                .withBucketName(bucketName)
                .withMaxResults(2);
            VersionListing versionListing = s3Client.listVersions(request);
            int numVersions = 0, numPages = 0;
            while(true) {
                numPages++;
                for (S3VersionSummary objectSummary :
                    versionListing.getVersionSummaries()) {
                    System.out.printf("Retrieved object %s, version %s\n",
                        objectSummary.getKey(),
                        objectSummary.getVersionId());
                    numVersions++;
                }
                // Check whether there are more pages of versions to retrieve. If
                // there are, retrieve them. Otherwise, exit the loop.
                if(versionListing.isTruncated()) {
                    versionListing = s3Client.listNextBatchOfVersions(versionListing);
                }
                else {
                    break;
                }
            }
            System.out.println(numVersions + " object versions retrieved in " + numPages +
                " pages");
        }
        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();
        }
    }
}
```

## 使用适用于 .NET 的 AWS 开发工具包

有关如何创建和测试有效示例的信息，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

### Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

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

namespace Amazon.DocSamples.S3
{
    class ListObjectsVersioningEnabledBucketTest
    {
        static 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(string[] args)
        {
            s3Client = new AmazonS3Client(bucketRegion);
            GetObjectListWithAllVersionsAsync().Wait();
        }

        static async Task GetObjectListWithAllVersionsAsync()
        {
            try
            {
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    // You can optionally specify key name prefix in the request
                    // if you want list of object versions of a specific object.

                    // For this example we limit response to return list of 2 versions.
                    MaxKeys = 2
                };
                do
                {
                    ListVersionsResponse response = await
s3Client.ListVersionsAsync(request);
                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, 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 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);
            }
        }
    }
}
```

```
    }
}
}
```

## 使用 REST API

要列出存储桶中所有对象的所有版本，请在 GET Bucket 请求中使用 versions 子资源。Amazon S3 最多只能检索 1000 个对象，且每个对象版本都将计为一个完整的对象。因此，如果存储桶包含两个键（例如 photo.gif 和 picture.jpg），并且第一个键有 990 个版本，第二个键有 400 个版本，则单个请求将检索 photo.gif 的所有 990 个版本，另加 picture.jpg 的最近 10 个版本。

Amazon S3 按照存储顺序返回对象版本，最先返回最近存储的版本。

### 在存储桶中列出所有对象版本的步骤

- 在 GET Bucket 请求中，包含 versions 子资源。

```
GET /?versions HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ORQf4/cRonhpaBX5sCYVf1bNRuU=
```

## 检索存储桶中对象的子集

此部分讨论以下两个示例场景：

- 您希望检索存储桶中所有对象版本的子集，例如检索特定对象的所有版本。
- 响应中对象版本的数目超过 max-key 的值（默认情况下为 1000），因此您必须再提交一个请求以检索其余的对象版本。

要检索对象版本的子集，请为 GET Bucket 使用请求参数。有关更多信息，请参阅 [GET Bucket](#)。

### 示例 1：仅检索特定对象的所有版本

您可以通过以下过程，使用 versions 子资源和 prefix 请求参数检索某对象的所有版本。有关 prefix 的更多信息，请参阅 [GET Bucket](#)。

#### 检索键的所有版本

- |   |  |
|---|--|
| 1 | 将 prefix 参数设置为您要检索的对象的键。   |
| 2 | 使用 versions 子资源和 prefix 发送 GET Bucket 请求。<br><br>GET /?versions&prefix=objectName HTTP/1.1 |

#### Example 使用前缀检索对象

以下示例将检索其键是 myObject 或由它开头的对象。

```
GET /?versions&prefix=myObject HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ORQf4/cRonhpaBX5sCYVf1bNRuU=
```

您可以使用其他请求参数来检索对象的所有版本的子集。有关更多信息，请参阅 [GET Bucket](#)。

## 示例 2：在响应截断时检索其他对象的列表

如果可以在 GET 请求中返回的对象数量超过 max-keys 的值，则响应将包含 `<isTruncated>true</isTruncated>`，并包含满足该请求但不会返回的第一个键（在 `NextKeyMarker` 中）和第一个版本 ID（在 `NextVersionIdMarker` 中）。您可以在后续请求中将这些返回的值用作开始位置，以检索满足 GET 请求的其他对象。

使用以下过程从存储桶中检索满足原始 GET Bucket versions 请求的其他对象。有关 key-marker、version-id-marker、`NextKeyMarker` 和 `NextVersionIdMarker` 的更多信息，请参阅 [GET Bucket](#)。

### 检索满足原始 GET 请求的其他响应

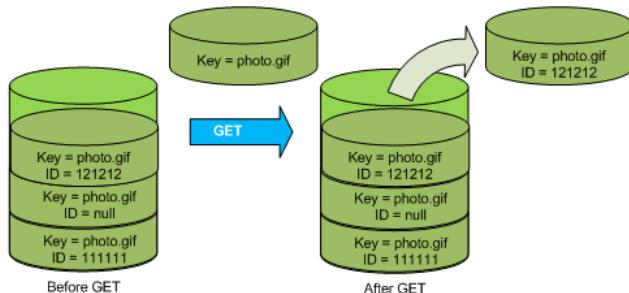
1	将 key-marker 的值设置为在上一个响应中的 <code>NextKeyMarker</code> 中返回的键。
2	将 version-id-marker 的值设置为在上一个响应中的 <code>NextVersionIdMarker</code> 中返回的版本 ID。
3	使用 key-marker 和 version-id-marker 发送 GET Bucket versions 请求。

### Example 检索从指定的密钥和版本 ID 开始的数据元

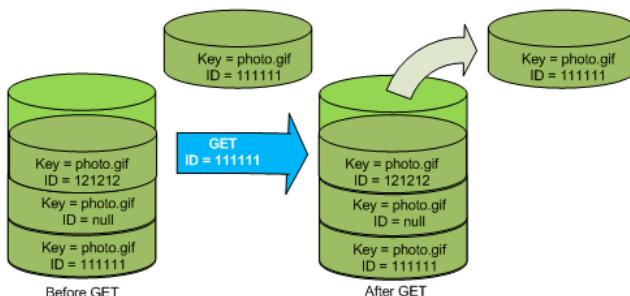
```
GET /?versions&key-marker=myObject&version-id-marker=298459348571 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

## 检索数据元版本

简单 GET 请求将检索对象的当前版本。下图显示 GET 如何返回 `photo.gif` 对象的当前版本。



要检索特定版本，您需要指定其版本 ID。下图显示 GET `versionId` 请求检索对象的指定版本（不一定是当前版本）。



## 使用控制台

有关说明，请参阅[如何查看 S3 对象的版本？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

## 使用 AWS 开发工具包

有关使用适用于 Java、.NET 和 PHP 的 AWS 开发工具包上传对象的示例，请参阅[获取对象 \(p. 142\)](#)。在无版本控制和启用版本控制的存储桶中上传对象的示例是相同的，只是对于启用版本控制的存储桶，Amazon S3 会分配版本号。否则，版本号为空。

有关使用其他 AWS 开发工具包的信息，请参阅[示例代码和库](#)。

## 使用 REST

### 检索特定对象版本的步骤

1. 将 `versionId` 设置为您要检索的对象的版本 ID。
2. 发送 GET Object `versionId` 请求。

#### Example 检索受版本控制的对象

以下请求将检索 `my-image.jpg` 的版本 `L4kqtJlcpXroDTDmpUMLUo`。

```
GET /my-image.jpg?versionId=L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

## 相关主题

[检索对象版本的元数据 \(p. 391\)](#)

## 检索对象版本的元数据

如果您仅想检索对象的元数据 (而不是其内容)，您可以使用 HEAD 操作。默认情况下，您将获得最新版本的元数据。要检索特定对象版本的元数据，需要指定其版本 ID。

### 检索对象版本的元数据的步骤

1. 将 `versionId` 设置为您要检索其元数据的对象的版本 ID。
2. 发送 HEAD Object `versionId` 请求。

#### Example 检索受版本控制的对象的元数据

以下请求将检索 `my-image.jpg` 的版本 `3HL4kqCxf3vjVBH40Nrjfkd` 的元数据。

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

以下显示了示例响应。

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
```

```
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

## 删除数据元版本

您在需要时随时可以删除对象版本。此外，还可为具有明确定义的生命周期的对象定义生命周期配置规则，请求 Amazon S3 使当前对象版本过期，或者永久删除非当前对象版本。当存储桶启用了版本控制或者版本控制功能已暂停时，生命周期配置操作的工作方式如下：

- `Expiration` 操作应用于当前对象版本，Amazon S3 通过添加删除标记将当前版本作为非当前版本保留（而不是删除当前对象版本），然后删除标记将成为当前版本。
- `NoncurrentVersionExpiration` 操作适用于非当前对象版本，Amazon S3 会永久删除这些对象版本。无法恢复永久删除的对象。

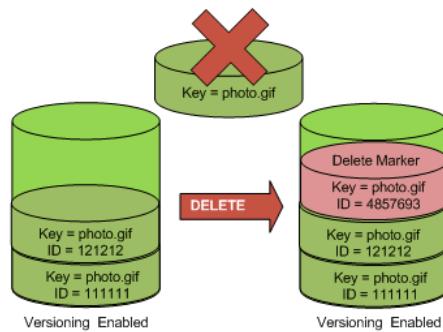
有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。

`DELETE` 请求具有以下使用案例：

- 启用版本控制后，简单 `DELETE` 无法永久删除对象。

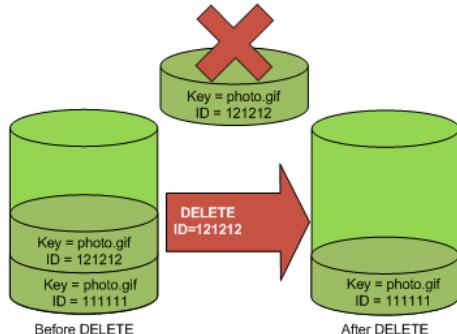
Amazon S3 将在存储桶中插入删除标记，该删除标记将成为对象的当前版本并具有新的 ID。当您尝试对当前版本为删除标记的对象执行 `GET` 操作时，Amazon S3 将该对象作为已删除对象对待（即使它尚未被擦除），并返回 404 错误。

下图显示简单 `DELETE` 实际上不会删除指定的对象。Amazon S3 插入一个删除标记。



- 要永久删除受版本控制的对象，您必须使用 `DELETE Object versionId`。

下图显示删除指定的对象版本将永久删除该对象。



## 使用控制台

有关说明，请参阅[如何查看 S3 对象的版本？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

## 使用 AWS 开发工具包

有关使用适用于 Java、.NET 和 PHP 的 AWS 开发工具包上传对象的示例，请参阅[删除对象 \(p. 203\)](#)。在无版本控制和启用版本控制的存储桶中上传对象的示例是相同的，只是对于启用版本控制的存储桶，Amazon S3 会分配版本号。否则，版本号为空。

有关使用其他 AWS 开发工具包的信息，请参阅[示例代码和库](#)。

## 使用 REST

### 删除对象的特定版本的步骤

- 在 DELETE 中，指定版本 ID。

#### Example 删除特定版本

以下示例显示了如何删除 photo.gif 的 版本 UIORUnfnd89493jJFJ。

```
DELETE /photo.gif?versionId=UIORUnfnd89493jJFJ HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```

## 相关主题

[使用 MFA 删除 \(p. 393\)](#)

[使用删除标记 \(p. 394\)](#)

[删除“删除标记” \(p. 395\)](#)

[使用版本控制 \(p. 380\)](#)

## 使用 MFA 删除

如果存储桶的版本控制配置已启用 MFA 删除，则存储桶拥有者必须在请求中包含 `x-amz-mfa` 请求标头，才能永久删除对象版本或更改该存储桶的版本控制状态。包含 `x-amz-mfa` 的请求必须使用 HTTPS。标头的值由身份验证设备的序列号、空格，以及在该设备上显示的身份验证代码组成。如果您没有包含此请求标头，则请求失败。

有关身份验证设备的更多信息，请参阅 [https://aws.amazon.com/iam/details/mfa/。](https://aws.amazon.com/iam/details/mfa/)

#### Example 从启用了 MFA 删除的存储桶中删除对象

以下示例显示了如何删除在存储桶中配置为启用 MFA 删除的 my-image.jpg (具有指定版本)。请注意 `[SerialNumber]` 和 `[AuthenticationCode]` 之间的空格。有关更多信息，请参阅 [DELETE 对象](#)。

```
DELETE /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTPS/1.1
```

```
Host: bucketName.s3.amazonaws.com
x-amz-mfa: 20899872 301749
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

有关启用 MFA 删除的更多信息，请参阅[MFA 删除 \(p. 381\)](#)。

## 使用删除标记

删除标记是用于受版本控制的对象（已在简单 DELETE 请求中命名）的占位符（标记）。因为对象位于已启用版本控制的存储桶中，所以不能删除该对象。但是，删除标记可以使 Amazon S3 将对象视为已删除。

与任何其他对象一样，删除标记同样有键名称（键）和版本 ID。但是，删除标记在以下方面与其他对象不同：

- 没有关联的数据。
- 没有关联的访问控制列表（ACL）值。
- 由于删除标记不包含数据，因此 GET 请求检索不到任何内容；该操作会引发 404 错误。
- 可以对删除标记执行的操作只有 DELETE，并且只有存储桶拥有者可以发出此请求。

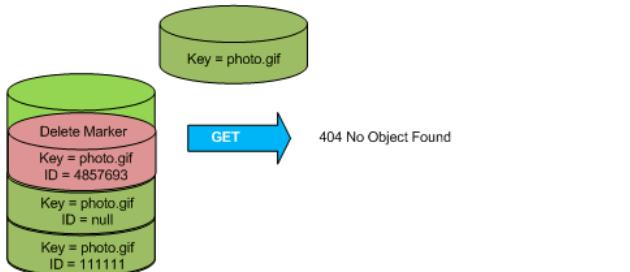
删除标记将累积对 Amazon S3 中的存储的名义费用。删除标记的存储大小等于删除标记键名称的大小。键名称是 Unicode 字符序列。对于名称中的每个字符，UTF-8 编码将 1 到 4 字节的存储添加到存储桶。有关键名称的更多信息，请参阅[对象键 \(p. 87\)](#)。有关将删除标记删除的信息，请参阅[删除“删除标记” \(p. 395\)](#)。

仅 Amazon S3 可以创建删除标记，每当您对启用了版本控制或已暂停版本控制的存储桶中的对象发送 DELETE Object 请求时，它就会创建删除标记。在 DELETE 请求中指定的对象实际上不会删除。而是使删除标记成为对象的当前版本。（该对象的键名称（键）将成为删除标记的键。）如果您尝试获取对象，而其当前版本为删除标记，则 Amazon S3 的响应为：

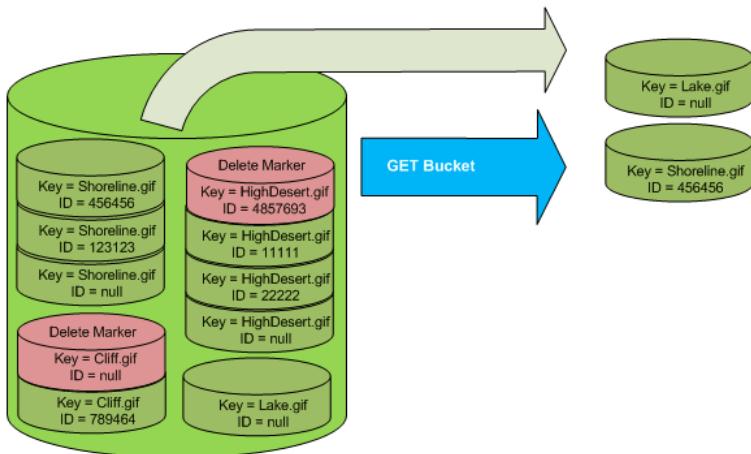
- 404 (Object not found) 错误
- 响应标头 x-amz-delete-marker: true

响应标头告知您所访问的对象是删除标记。此响应标头从不会返回 false；如果值为 false，则 Amazon S3 不会在响应中包含此响应标头。

下图显示了对对象（其当前版本为删除标记）的简单 GET 如何返回 404 No Object Found（未找到对象）错误。



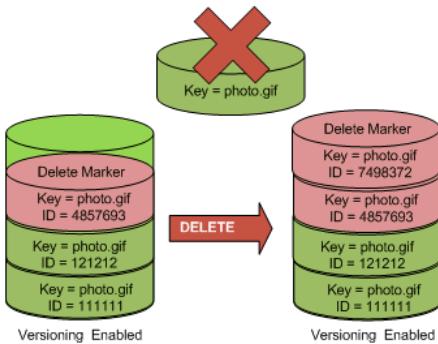
列出删除标记（和对象的其他版本）的唯一方法在 GET Bucket versions 请求中使用 versions 子资源。简单 GET 不会检索删除标记对象。下图显示 GET Bucket 请求不会返回其当前版本为删除标记的对象。



## 删除“删除标记”

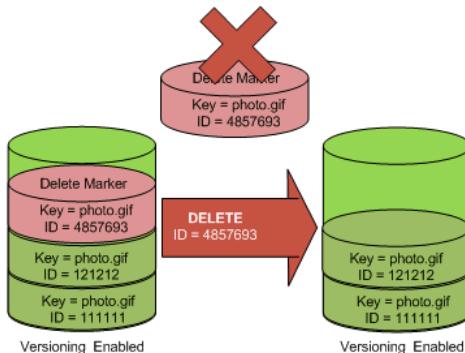
要删除“删除标记”，必须在 `DELETE Object versionId` 请求中指定它的版本 ID。如果您使用 `DELETE` 请求将删除标记删除（而未指定删除标记的版本 ID），Amazon S3 将不会删除该删除标记，而是再插入一个删除标记。

下图显示了删除标记上的简单 `DELETE` 未删除任何内容，但向存储桶添加了新删除标记的方式。



在已启用版本控制的存储桶中，这一新删除标记将具有唯一的版本 ID。因此，在一个存储桶中，相同的对象可能有多个删除标记。

要永久删除“删除标记”，必须在 `DELETE Object versionId` 请求中包含其版本 ID。下图显示了 `DELETE Object versionId` 请求如何永久删除“删除标记”。只有存储桶拥有者可以永久删除“删除标记”。



去除“删除标记”的效果是：简单 GET 请求现在将检索对象的当前版本 (121212)。

### 永久删除“删除标记”的步骤

1. 将 `versionId` 设置为要删除的删除标记的版本 ID。
2. 发送 `DELETE Object versionId` 请求。

#### Example 删除“删除标记”

以下示例删除用于 `photo.gif` 版本 4857693 的删除标记。

```
DELETE /photo.gif?versionId=4857693 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

当您将删除标记删除时，Amazon S3 在响应中包含：

```
204 NoContent
x-amz-version-id: versionID
x-amz-delete-marker: true
```

## 转换对象版本

可为具有明确定义的生命周期的对象定义生命周期配置规则，以便在对象生命周期的特定时间将对象版本转换为 `GLACIER` 存储类。有关更多信息，请参阅 [对象生命周期管理 \(p. 104\)](#)。

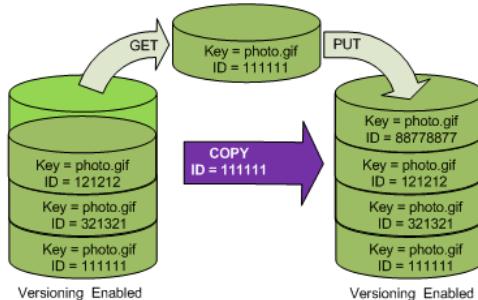
## 还原早期版本

版本控制的其中一个价值主张是能够检索对象的早期版本。有两种方法可执行该操作：

- 将对象的早期版本复制到同一存储桶中
  - 复制的对象将成为该对象的当前版本，且所有对象版本都保留。
- 永久删除对象的当前版本

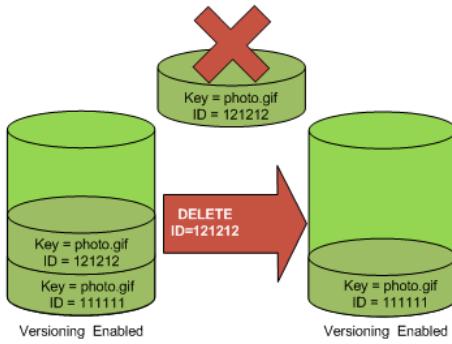
当您删除当前对象版本时，实际上会将先前版本转换为该对象的当前版本。

由于所有对象版本都保留，所以您可以通过将对象的特定版本复制到同一存储桶，使任何较早版本成为当前版本。在下图中，源对象 (`ID = 111111`) 已复制到同一存储桶中。Amazon S3 将提供一个新 ID (`88778877`)，且它将成为该对象的当前版本。因此，该存储桶同时具有原始对象版本 (`111111`) 及其副本 (`88778877`)。



后续 `GET` 将检索版本 88778877。

下图显示了如何删除对象的当前版本 (121212)，该操作保留早期版本 (111111) 作为当前版本。



后续 GET 将检索版本 111111。

## 受版本控制的对象的权限

权限将在版本级别上设置。每个版本都有自己的对象拥有者；创建对象版本的 AWS 账户就是拥有者。因此，您可以为同一对象的不同版本设置不同的权限。要这样做，您必须指定要在 `PUT Object versionId acl` 请求中设置其权限的对象的版本 ID。有关详细描述和关于使用 ACL 的说明，请参阅[管理对 Amazon S3 资源的访问权限 \(p. 242\)](#)。

### Example 为对象版本设置权限

以下请求将被授权者 (`BucketOwner@amazon.com`) 对键 (`FULL_CONTROL`) 和版本 ID (`3HL4kqtJvjVBH40Nrjfk`) 的权限设置为 `my-image.jpg`。

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJvjVBH40Nrjfk HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeef76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bfe65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>BucketOwner@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

同样地，要获取特定对象版本的权限，必须在 `GET Object versionId acl` 请求中指定它的版本 ID。您需要包含版本 ID，因为在默认情况下，`GET Object acl` 将返回对象的当前版本的权限。

### Example 检索用于指定对象版本的权限

在以下示例中，Amazon S3 将返回用于键 (`my-image.jpg`) 和版本 ID (`DVBH40Nr8X8gUMLUo`) 的权限。

```
GET /my-image.jpg?versionId=DVBH40Nr8X8gUMLUo&acl HTTP/1.1
```

```
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU
```

有关更多信息，请参阅 [GET Object acl](#)。

## 管理已暂停版本控制的存储桶中的对象

### 主题

- [将对象添加到已暂停版本控制的存储桶 \(p. 398\)](#)
- [从已暂停版本控制的存储桶检索对象 \(p. 399\)](#)
- [从已暂停版本控制的存储桶中删除对象 \(p. 399\)](#)

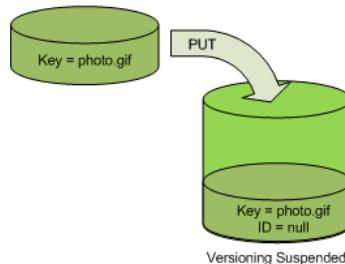
您可以暂停版本控制，以停止在存储桶中累积同一对象的新版本。您可能会因为您只想在存储桶中拥有单个对象版本，或不想累积对多个版本收取的费用而执行此操作。

暂停版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理对象的方式。此部分中的主题说明在已暂停版本控制的存储桶中进行的各种对象操作。

### 将对象添加到已暂停版本控制的存储桶

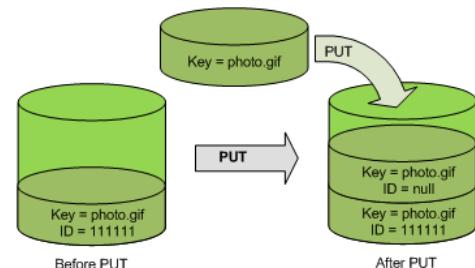
在存储桶上暂停版本控制后，Amazon S3 会自动将 null 版本 ID 添加到之后存储在该存储桶中的每个后续对象（使用 PUT、POST 或 COPY）。

下图显示当向已暂停版本控制的存储桶中添加对象时，Amazon S3 如何向该对象添加 null 版本 ID。

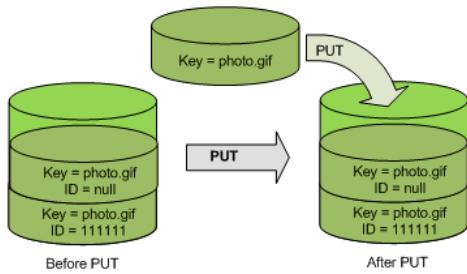


如果存储桶中已存在空版本，且您使用同一键添加了其他对象，则添加的对象将覆盖原始的空版本。

如果存储桶中存在受版本控制的对象，则使用 PUT 存储的版本将成为该对象的当前版本。下图显示了如何将对象添加到包含受版本控制的对象（不会覆盖已存在于该存储桶中的对象）的存储桶。在这种情况下，版本 111111 已存在于该存储桶中。Amazon S3 会将空的版本 ID 附加到所添加的对象，并将其存储在存储桶中。版本 111111 不会被覆盖。



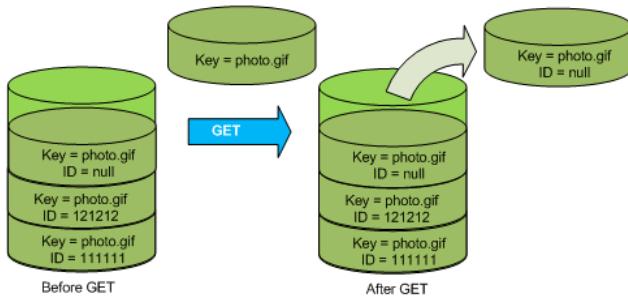
如果存储桶中已存在空版本，则该空版本将被覆盖，如下图所示。



注意，虽然空版本的键和版本 ID (null) 在 PUT 之前和之后都相同，但是原来存储在存储桶中的空版本的内容将替换为该存储桶中对象 PUT 的内容。

## 从已暂停版本控制的存储桶检索对象

无论是否在存储桶上启用版本控制，GET Object 请求都将返回对象的当前版本。下图显示了简单 GET 将如何返回对象的当前版本。

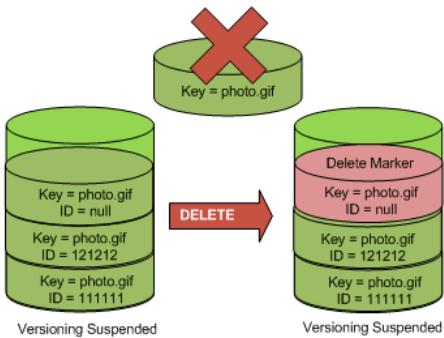


## 从已暂停版本控制的存储桶中删除对象

如果暂停了版本控制，DELETE 请求：

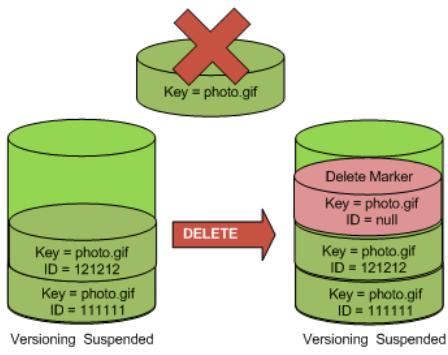
- 可以仅删除其版本 ID 为 null 的对象  
如果存储桶中没有对象的空版本，则不删除任何内容。
- 将删除标记插入到存储桶。

下图显示简单 DELETE 如何删除空版本，以及 Amazon S3 如何使用 null 版本 ID 在其位置插入删除标记。

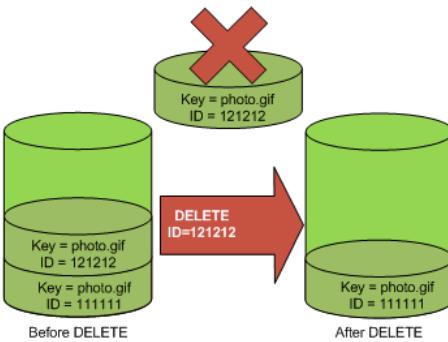


请记住，删除标记不包含任何内容，所以您会在删除标记替换空版本时丢失该空版本的内容。

下图显示不具有空版本的存储桶。在这种情况下，DELETE 不会删除任何内容；Amazon S3 仅插入删除标记。



即使是在已暂停版本控制的存储桶中，存储桶拥有者也可以永久删除指定版本。下图显示删除指定的对象版本将永久删除该对象。只有存储桶拥有者可以删除指定的对象版本。



# 在 Amazon S3 上托管静态网站

您可以在 Amazon Simple Storage Service (Amazon S3) 上托管静态网站。在静态网站上，单独的网页包含静态内容。它们也可能包含客户端脚本。通过对比得知，动态网站依赖服务器端处理，包括诸如 PHP、JSP 或 ASP.NET 的服务器端脚本。Amazon S3 不支持服务器端脚本编写。Amazon Web Services (AWS) 还提供用于托管动态网站的资源。要了解有关 AWS 上的网站托管的更多信息，请转到[网站和网站托管](#)。

## 主题

- [网站终端节点 \(p. 402\)](#)
- [为网站托管配置存储桶 \(p. 403\)](#)
- [示例演练 – 在 Amazon S3 上托管网站 \(p. 414\)](#)

要托管静态网站，您需要为网站托管配置 Amazon S3 存储桶，然后将网站内容上传到存储桶。此存储桶必须拥有公共读取访问权限。全世界的任何人都将可以读取此存储桶是刻意的设置。网站将在存储桶中 AWS 区域特定的网站终端节点中提供，使用以下格式之一：

```
<bucket-name>.s3-website-<AWS-region>.amazonaws.com
```

```
<bucket-name>.s3-website.<AWS-region>.amazonaws.com
```

有关 Amazon S3 的 AWS 区域特定的网站终端节点列表，请参阅[网站终端节点 \(p. 402\)](#)。例如，假设您在美国西部（俄勒冈）区域 创建了名为 examplebucket 的存储桶，并将其配置为网站。以下示例 URL 将提供对您的网站内容的访问：

- 此 URL 将返回您为该网站配置的默认索引文档。

```
http://examplebucket.s3-website-us-west-2.amazonaws.com/
```

- 此 URL 将请求 photo.jpg 对象，该对象存储在存储桶的根级。

```
http://examplebucket.s3-website-us-west-2.amazonaws.com/photo.jpg
```

- 此 URL 将请求存储桶中的 docs/doc1.html 对象。

```
http://examplebucket.s3-website-us-west-2.amazonaws.com/docs/doc1.html
```

## 使用您自己的域

您可以使用自己的域（例如 example.com）提供内容，而不是通过使用 Amazon S3 网站终端节点来访问网站。Amazon S3 与 Amazon Route 53 相结合，支持在根域中托管网站。例如，如果您有根域 example.com，并在 Amazon S3 上托管网站，则您的网站访问者可以通过键入 `http://www.example.com` 或 `http://example.com` 从他们的浏览器访问该站点。有关示例演练的信息，请参阅[示例：使用自定义域设置静态网站 \(p. 415\)](#)。

要为网站托管配置存储桶，需将网站配置添加到该存储桶。有关更多信息，请参阅[为网站托管配置存储桶 \(p. 403\)](#)。

## 网站终端节点

当您为网站托管配置存储桶时，该网站将在特定区域的网站终端节点上可用。网站终端节点不同于您在其上发送 REST API 请求的终端节点。有关终端节点之间的差异的更多信息，请参阅[Amazon 网站和 REST API 终端节点之间的主要差异 \(p. 402\)](#)。

Amazon S3 网站终端节点的两种一般格式如下：

`bucket-name.s3-website-region.amazonaws.com`

`bucket-name.s3-website.region.amazonaws.com`

用于终端节点的格式取决于存储桶所在的区域。例如，如果存储桶名为 `example-bucket`，位于美国西部（俄勒冈）区域，则此网站在以下 Amazon S3 网站终端节点可用：

`http://example-bucket.s3-website-us-west-2.amazonaws.com/`

或者，如果存储桶名为 `example-bucket`，位于欧洲（法兰克福）区域，则此网站在以下 Amazon S3 网站终端节点可用：

`http://example-bucket.s3-website.eu-central-1.amazonaws.com/`

有关按区域列出的 Amazon S3 网站终端节点的列表，请参阅 AWS 一般参考中的 [Amazon Simple Storage Service 网站终端节点](#)。

为了使您的客户可以访问网站终端节点上的内容，您必须使您的所有内容公开可读。这样，您可以使用对象上的存储桶策略或 ACL 来授予必要的权限。

### Note

申请方付款存储桶 不允许通过网站终端节点访问。对此种存储桶的任何请求都将收到 403 Access Denied 响应。有关更多信息，请参阅 [申请方付款存储桶 \(p. 73\)](#)。

如果您有已注册的域，则可以添加指向 Amazon S3 网站终端节点的 DNS 别名记录条目。例如，如果您拥有已注册的域 `www.example-bucket.com`，则可以创建存储桶 `www.example-bucket.com`，并添加指向 `www.example-bucket.com.s3-website-<region>.amazonaws.com` 的 DNS 别名记录。对 `http://www.example-bucket.com` 的所有请求都将路由到 `www.example-bucket.com.s3-website-<region>.amazonaws.com`。有关更多信息，请参阅 [存储桶的虚拟托管 \(p. 42\)](#)。

## Amazon 网站和 REST API 终端节点之间的主要差异

网站终端节点针对通过 Web 浏览器访问进行了优化。下表描述了 Amazon REST API 终端节点和网站终端节点之间的主要差异。

主要差异	REST API 终端节点	网站终端节点
访问控制	同时支持公共内容和私有内容。	仅支持公开可读的内容。
错误消息处理	返回 XML 格式的错误响应。	返回 HTML 文档。
重定向支持	不适用	同时支持对象级和存储桶级重定向。
支持的请求	支持所有存储桶和对象操作	仅支持对象上的 GET 和 HEAD 请求。

主要差异	REST API 终端节点	网站终端节点
对存储桶根级的 GET 和 HEAD 请求的响应	返回存储桶中对象键的列表。	返回在网站配置中指定的索引文档。
安全套接字层 (SSL) 支持	支持 SSL 连接。	不支持 SSL 连接。

有关 Amazon S3 终端节点的列表，请参阅[请求终端节点 \(p. 10\)](#)。

## 为网站托管配置存储桶

您可以在 Amazon Simple Storage Service (Amazon S3) 存储桶中托管静态网站。但是，执行此操作需要进行某个配置。还有一些可选配置，具体取决于您的网站要求。

必需配置：

- [启用网站托管 \(p. 403\)](#)
- [配置索引文档支持 \(p. 403\)](#)
- [网站访问所需的权限 \(p. 405\)](#)

可选配置：

- [\(可选\) 配置 Web 流量日志记录 \(p. 406\)](#)
- [\(可选\) 自定义错误文档支持 \(p. 406\)](#)
- [\(可选\) 配置网页重定向 \(p. 408\)](#)

## 启用网站托管

执行下列步骤可使用 [Amazon S3 控制台](#) 为您的 Amazon S3 存储桶启用网站托管：

为 Amazon S3 存储桶启用网站托管

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 在列表中，选择要用于托管网站的存储桶。
3. 选择 Properties 选项卡。
4. 选择 Static website hosting，然后选择 Use this bucket to host a website。
5. 系统将提示您提供索引文档和需要的任何可选错误文档和重定向规则。

有关索引文档的概念的信息，请参阅[配置索引文档支持 \(p. 403\)](#)。

## 配置索引文档支持

索引文档 是在对网站的根或任何子文件夹发出请求时 Amazon S3 返回的网页。例如，如果用户在浏览器中输入 `http://www.example.com`，则该用户没有请求任何特定页面。在这种情况下，Amazon S3 将提供索引文档，该文档有时也称为默认页面。

当您将您的存储桶配置为网站时，应该提供索引文档的名称。然后必须上传具有此名称的对象并将其配置为公开可读。

根级 URL 的尾部斜杠是可选的。例如，如果您将具有 `index.html` 的网站配置为索引文档，以下任意一个 URL 将返回 `index.html`。

```
http://example-bucket.s3-website-region.amazonaws.com/  
http://example-bucket.s3-website-region.amazonaws.com
```

有关 Amazon S3 网站终端节点的更多信息，请参阅[网站终端节点 \(p. 402\)](#)。

## 索引文档和文件夹

在 Amazon S3 中，存储桶是对象的平面容器；它不会像计算机上的文件系统那样提供任何分层组织。您可以使用表示文件夹结构的对象键名称创建逻辑层级结构。例如，考虑具有三个对象的存储桶和以下键名称。

- `sample1.jpg`
- `photos/2006/Jan/sample2.jpg`
- `photos/2006/Feb/sample3.jpg`

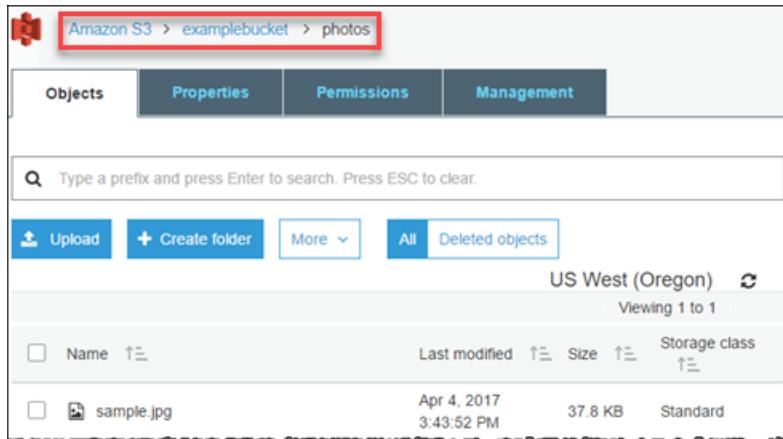
虽然它们没有按任何物理分层组织进行存储，但您可以从键名称推断以下逻辑文件夹结构。

- `sample1.jpg` 对象位于存储桶的根级。
- `sample2.jpg` 对象位于 `photos/2006/Jan` 子文件夹中。
- `sample3.jpg` 对象位于 `photos/2006/Feb` 子文件夹中。

Amazon S3 控制台支持的文件夹概念基于对象键名称。要继续以前的示例，控制台需显示包含 `photos` 文件夹的 `examplebucket`。

The screenshot shows the Amazon S3 console interface for the 'examplebucket' bucket. At the top, there's a navigation bar with the S3 logo and the path 'Amazon S3 > examplebucket'. Below this is a header with tabs: 'Objects' (selected), 'Properties' (highlighted in blue), 'Permissions', and 'Management'. A search bar below the header contains the placeholder text 'Type a prefix and press Enter to search. Press ESC to clear.' Under the search bar are buttons for 'Upload', '+ Create folder', 'More', 'All' (selected), and 'Deleted objects'. The main area is titled 'Objects' and shows a table with one item: a folder named 'photos'. The table has columns for 'Name' and 'Last modified'. The 'Name' column shows the folder icon followed by 'photos'. The 'Last modified' column shows the date '2006-03-01'. There are also checkboxes and other icons in the table.

您可以将对象上传到存储桶或该存储桶中的 `photos` 文件夹。如果将对象 `sample.jpg` 添加到存储桶，则键名称为 `sample.jpg`。如果将对象上传到 `photos` 文件夹，则对象键名称为 `photos/sample.jpg`。



如果您在存储桶中创建了此类文件夹结构，则您必须在每个级别上都具有索引文档。当用户指定类似于文件夹查找的 URL 时，是否存在尾部斜杠将决定网站的行为。例如，以下具有尾部反斜杠的 URL 将返回 `photos/index.html` 索引文档。

```
http://example-bucket.s3-website-region.amazonaws.com/photos/
```

但是，如果您从之前的 URL 排除尾部斜杠，则 Amazon S3 将首先在存储桶中查找对象 `photos`。如果未找到 `photos` 对象，则将搜索索引文档 `photos/index.html`。如果找到该文档，则 Amazon S3 返回 `302 Found` 消息并指向 `photos/` 键。对于对 `photos/` 的后续请求，Amazon S3 返回 `photos/index.html`。如果未找到索引文档，Amazon S3 将返回错误。

## 网站访问所需的权限

当您将存储桶配置为网站时，您必须使要提供的对象公开可读。要执行此操作，您需要编写向每个人授予 `s3:GetObject` 权限的存储桶策略。在网站终端节点上，如果用户请求的对象不存在，则 Amazon S3 将返回 HTTP 响应代码 `404 (Not Found)`。如果该对象存在，但您尚未授予对对象的读取权限，则网站终端节点将返回 HTTP 响应代码 `403 (Access Denied)`。用户可以使用该响应代码推断特定对象是否存在。如果您不需要此行为，则不应启用对存储桶的网站支持。

以下示例存储桶策略向每个人授予了访问指定文件夹中的对象的权限。有关存储桶策略的更多信息，请参阅[使用存储桶策略和用户策略](#) (p. 279)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::example-bucket/*"]
    }
  ]
}
```

### Note

记住以下内容：

- 若要托管网站，您的存储桶必须具有公共读取访问权限。全世界的任何人都将可以读取此存储桶是刻意的设置。

- 存储桶策略仅适用于存储桶拥有者所拥有的对象。如果存储桶包含并非由存储桶拥有者拥有的对象，则应使用对象访问控制列表 (ACL) 授予对这些对象的公有 READ 权限。

您可以通过存储桶策略或对象 ACL 授予对您的对象的公开读取权限。要使用 ACL 使对象公开可读，您可以向 AllUsers 组授予 READ 权限，如以下授权元素所示。可以将此授权元素添加到对象 ACL 中。有关管理 ACL 的信息，请参阅[使用 ACL 管理访问 \(p. 333\)](#)。

```
<Grant>
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="Group">
  <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
</Grantee>
<Permission>READ</Permission>
</Grant>
```

## (可选) 配置 Web 流量日志记录

如果想要追踪访问您的网站的访问者人数，需要对根域存储桶启用日志记录。启用日志记录是可选的。

### 对根域存储桶启用日志记录

- 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
- 在创建了 example.com 和 www.example.com 存储桶的 AWS 区域中，为名为 logs.example.com 的日志记录创建存储桶。
- 在 logs.example.com 存储桶中创建两个文件夹，一个文件夹名为 root，另一个文件夹名为 cdn。如果配置 Amazon CloudFront 以便为网站提速，则会使用 cdn 文件夹。
- 在存储桶名称列表中，选择您的根域存储桶，然后依次选择属性、服务器访问日志记录。
- 选择启用日志记录。
- 对于目标存储桶，选择您为日志文件创建的存储桶 logs.example.com。
- 对于目标前缀，键入 root/。此设置将存储桶中的日志数据文件分组到名为 root 的文件夹中，以便轻松查找。
- 选择 Save (保存)。

## (可选) 自定义错误文档支持

下表列出了发生错误时 Amazon S3 返回的 HTTP 响应代码的子集。

HTTP 错误代码	说明
301 永久移动	当用户将请求直接发送到 Amazon S3 网站终端节点 ( <a href="http://s3-website-&lt;region&gt;.amazonaws.com/">http://s3-website-&lt;region&gt;.amazonaws.com/</a> ) 时，Amazon S3 返回 301 Moved Permanently (301 永久移动) 响应并将这些请求重定向到 <a href="https://aws.amazon.com/s3/">https://aws.amazon.com/s3/</a> 。
302 Found (302 已找到)	当 Amazon S3 接收对不包含尾部斜杠的键 x ( <a href="http://&lt;bucket&gt;.s3-website-&lt;region&gt;.amazonaws.com/x">http://&lt;bucket&gt;.s3-website-&lt;region&gt;.amazonaws.com/x</a> ) 的请求时，它首先查找键名称为 x 的对象。如果未找到对象，则 Amazon S3 确定该请求是针对子文件夹 x 发出的，并通过在末尾添加斜杠重定向请求并返回 302 Found (302 已找到)。
304 Not Modified (304 未修改)	Amazon S3 用户请求标头 If-Modified-Since、If-Unmodified-Since、If-Match 和/或 If-None-Match 以确定所请求的对象是否与客户端保存的缓存副本相同。如果对象相同，网站终端节点将返回 304 Not Modified 响应。

HTTP 错误代码	说明
400 Malformed Request	当用户尝试通过错误的区域终端节点访问存储桶时，网站终端节点的响应包含 400 Malformed Request。
403 禁止访问	当用户请求转换为不可公开读取的对象时，网站终端节点的响应包含 403 Forbidden。对象所有者必须使用存储桶策略或 ACL 使该对象公开可读。
404 未找到	由于以下原因，网站终端节点的响应包含 404 Not Found： <ul style="list-style-type: none"><li>Amazon S3 确定网站 URL 引用了不存在的对象键。</li><li>Amazon 推断该请求针对不存在的索引文档。</li><li>在 URL 中指定的存储桶不存在。</li><li>URL 中指定的存储桶存在，但未配置为网站。</li></ul> <p>您可以创建为 404 Not Found 返回的自定义文档。确保该文档已上传到配置为网站的存储桶，且网站托管配置已设置为使用该文档。 有关 Amazon S3 如何将 URL 解释为对对象或索引文档的请求的信息，请参阅<a href="#">配置索引文档支持 (p. 403)</a>。</p>
500 Service Error	当出现内部服务器错误时，网站终端节点的响应包含 500 Service Error。
503 服务不可用	当 Amazon S3 确定您需要降低请求率时，网站终端节点的响应包含 503 Service Unavailable (503 服务不可用)。

对于其中每个错误，Amazon S3 都返回一条预定义的 HTML 消息。以下是对 403 Forbidden 响应返回的 HTML 消息示例。

## 403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: 873CA367A51F7EC7
- HostId: DdQezl9vkuw5huD5HKsFaTDm9KH4PZzCPRkW3igmILbTu1DiYlvXjgyd7pVxq32

### An Error Occurred While Attempting to Retrieve a Custom Error Document

- Code: AccessDenied
- Message: Access Denied

## 自定义错误文档

(可选) 您可以提供包含用户友好错误消息和其他帮助的自定义错误文档。您将提供此自定义错误文档作为将网站配置添加到您的存储桶的一部分。Amazon S3 将仅为 HTTP 4XX 类的错误代码返回您的自定义错误文档。

## 错误文档和浏览器行为

出现错误时，Amazon S3 返回 HTML 错误文档。如果您为网站配置了自定义错误文档，则 Amazon S3 将返回该错误文档。但是，当出现错误时，某些浏览器将显示自己的错误消息，而忽略 Amazon S3 返回的错误文档。例如，当出现“HTTP 404 未找到”错误时，Google Chrome 可能忽略 Amazon S3 返回的错误文档并显示自己的错误。

## (可选) 配置网页重定向

如果为网站托管配置了 Amazon S3 存储桶，则可以将对某个对象的请求重定向到同一存储桶中的其他对象或重定向到外部 URL。

### 主题

- [Amazon S3 控制台中的页面重定向支持 \(p. 408\)](#)
- [从 REST API 设置页面重定向 \(p. 409\)](#)
- [高级条件重定向 \(p. 410\)](#)

您可以通过将 `x-amz-website-redirect-location` 属性添加到对象元数据来设置重定向。然后，该网站将把对象当作 301 重定向。要重定向对其他对象的请求，您可以将重定向位置设置为目标对象的键。要重定向对外部 URL 的请求，您可以将重定向位置设置为所需的 URL。有关对象元数据的更多信息，请参阅 [系统定义的元数据 \(p. 89\)](#)。

为网站托管配置的存储桶具有网站终端节点和 REST 终端节点。对配置为 301 重定向的页面的请求具有以下可能的结果，具体取决于请求的终端节点：

- 特定于区域的网站终端节点 – Amazon S3 根据 `x-amz-website-redirect-location` 属性的值重定向页面请求。
- REST 终端节点 – Amazon S3 不重定向页面请求。它将返回请求的对象。

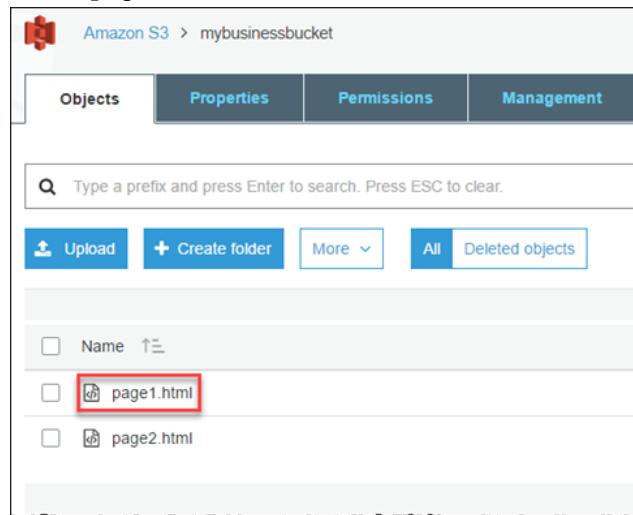
有关终端节点的更多信息，请参阅 [Amazon 网站和 REST API 终端节点之间的主要差异 \(p. 402\)](#)。

您可从 Amazon S3 控制台或使用 Amazon S3 REST API 设置页面重定向。

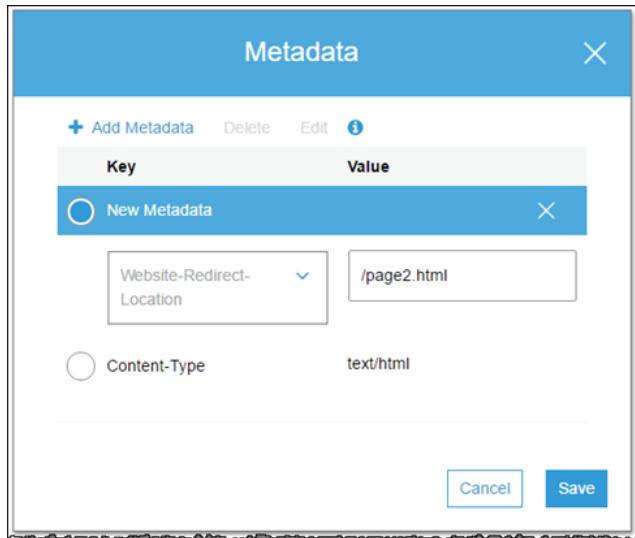
## Amazon S3 控制台中的页面重定向支持

您可以使用 Amazon S3 控制台在对象的元数据中设置网站重定向位置。当您设置页面重定向时，您可以保留或删除源对象内容。例如，假设您的存储桶中有 `page1.html` 对象。要将对此页面的任何请求重定向到另一个对象 `page2.html`，您可以执行以下操作之一：

- 要保留 `page1.html` 对象的内容并仅重定向页面请求，请选择 `page1.html` 对象。



选择 `page1.html` 的属性选项卡，然后选择元数据框。如以下示例所示，将 `Website Redirect Location` 添加到元数据，然后将其值设置为 `/page2.html`。需要该值中的 / 前缀。



您也可以将该值设置为外部 URL，例如 `http://www.example.com`。例如，如果您的根域为 `example.com`，并且您要为 `http://example.com` 和 `http://www.example.com` 的请求服务，则可以创建两个分别名为 `example.com` 和 `www.example.com` 的存储桶。然后，将内容保留在其中一个存储桶中（例如 `example.com`），然后配置另一个存储桶以将所有请求重定向至 `example.com` 存储桶。

- 要删除 `page1.html` 对象的内容并重定向请求，您可以使用相同的键上传新的零字节对象 `page1.html` 来替换现有对象，然后在上传过程中为 `page1.html` 指定 Website Redirect Location。有关上传对象的信息，请参阅 Amazon Simple Storage Service 控制台用户指南中的 [上传 S3 对象](#)。

## 从 REST API 设置页面重定向

以下 Amazon S3 API 操作支持请求中的 `x-amz-website-redirect-location` 标头。Amazon S3 将对对象元数据中的标头值存储为 `x-amz-website-redirect-location`。

- [PUT Object](#)
- [开始分段上传](#)
- [POST Object](#)
- [PUT Object – Copy](#)

设置页面重定向时，您可以保留或删除对象内容。例如，假设您在您的存储桶中具有 `page1.html` 对象。

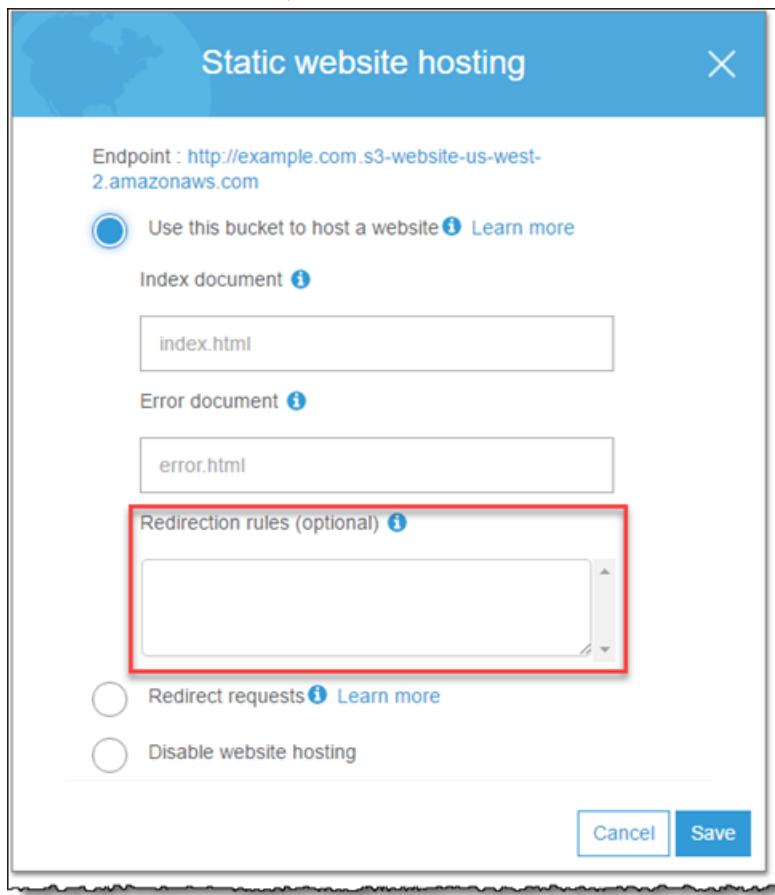
- 要保留 `page1.html` 的内容并仅重定向页面请求，您可以提交 [PUT Object – Copy](#) 请求来创建新的 `page1.html` 对象，该对象使用现有 `page1.html` 对象作为源。在您的请求中，您可以设置 `x-amz-website-redirect-location` 标头。当请求完成时，您将拥有未更改内容的原始页面，但 Amazon S3 会将对该页面的任何请求重定向到指定的重定向位置。
- 要删除 `page1.html` 对象的内容并重定向对该页面的请求，您可以发送 [PUT Object](#) 请求，以上传具有相同对象键的零字节对象 `page1.html`。在 [PUT](#) 请求中，您可以将 `page1.html` 的 `x-amz-website-redirect-location` 设置为新对象。当请求完成时，`page1.html` 将不包含任何内容，且后续请求将被重定向到由 `x-amz-website-redirect-location` 指定的位置。

当使用 [GET Object](#) 操作以及其他对象元数据检索该对象时，Amazon S3 在响应中返回 `x-amz-website-redirect-location` 标头。

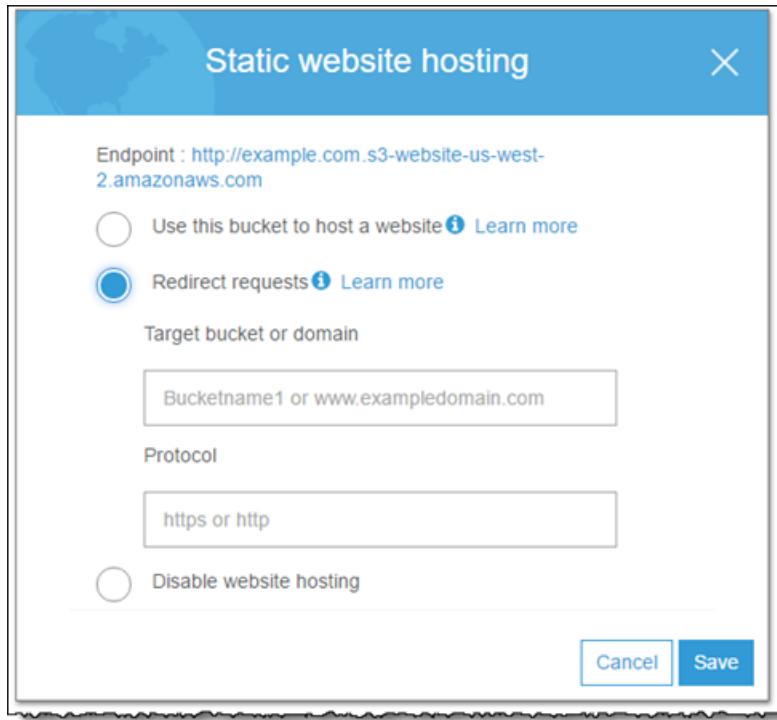
## 高级条件重定向

利用高级重定向规则，您可以根据特定对象键名称、请求中的前缀或者响应代码来按条件路由请求。例如，假设您在存储桶中删除或重命名了某个对象。您可以添加一个将请求重定向到其他对象的路由规则。如果您要使文件夹不可用，则可以添加路由规则将请求重定向至其他网页。处理错误时，您还可以通过将返回错误的请求路由到其他域来添加一个处理错误条件的路由规则。

为网站托管配置存储桶时，您可以选择指定高级重定向规则。



要将对存储桶网站终端节点的所有请求重定向至其他主机，只需提供主机名。



您使用 XML 描述规则。下一节提供了指定重定向规则的一般语法和示例。

## 用于指定路由规则的语法

以下是在网站配置中定义路由规则的一般语法：

```
<RoutingRules> =
<RoutingRules>
  <RoutingRule>...</RoutingRule>
  [<RoutingRule>...</RoutingRule>
  ...]
</RoutingRules>

<RoutingRule> =
<RoutingRule>
  [ <Condition>...</Condition> ]
  <Redirect>...</Redirect>
</RoutingRule>

<Condition> =
<Condition>
  [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
  [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
</Condition>
Note: <Condition> must have at least one child element.

<Redirect> =
<Redirect>
  [ <HostName>...</HostName> ]
  [ <Protocol>...</Protocol> ]
  [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith> ]
  [ <ReplaceKeyWith>...</ReplaceKeyWith> ]
  [ <HttpRedirectCode>...</HttpRedirectCode> ]
</Redirect>
```

*Note: <Redirect> must have at least one child element.  
 Also, you can have either ReplaceKeyPrefix with or ReplaceKeyWith,  
 but not both.*

下表描述了路由规则中的元素。

名称	描述
RoutingRules	用于收集 RoutingRule 元素的容器。
RoutingRule	标识在满足条件时应用的条件和重定向的规则。  条件：RoutingRules 容器必须包含至少一个路由规则。
Condition	对于要应用的指定的重定向，用于描述必须满足的条件的容器。如果路由规则不包含条件，则该规则将应用于所有请求。
KeyPrefixEquals	从中重定向请求的对象键名称的前缀。  KeyPrefixEquals 如果未指定 ErrorCodeReturnedEquals，则需要。如果同时指定 KeyPrefixEquals 和 ErrorCodeReturnedEquals，则两者都必须为真才能满足条件。
HttpErrorCodeReturnedEquals	要应用的重定向必须匹配的 HTTP 错误代码。如果出现错误，并且错误代码满足此值，则应用指定的重定向。  HttpErrorCodeReturnedEquals 如果未指定 KeyPrefixEquals，则需要。如果同时指定 KeyPrefixEquals 和 ErrorCodeReturnedEquals，则两者都必须为真才能满足条件。
Redirect	提供用于重定向请求的说明的容器元素。您可以将请求重定向到其他主机或其他页面，或者也可以指定要使用的其他协议。RoutingRule 必须具有 Redirect 元素。Redirect 元素必须至少包含以下一个同级元素：Protocol、HostName、ReplaceKeyPrefixWith、ReplaceKeyWith 或 HttpRedirectCode。
Protocol	响应中返回的 Location 标头中将要使用的协议 (http 或 https)。  如果提供了一个同级，则 Protocol 不是必需的。
HostName	可在响应中返回的位置标头中使用的主机名。  如果提供了一个同级，则 HostName 不是必需的。
ReplaceKeyPrefixWith	将替换重定向请求中的 KeyPrefixEquals 值的对象键名称的前缀。  如果提供了一个同级，则 ReplaceKeyPrefixWith 不是必需的。仅在不提供 ReplaceKeyWith 时提供它。
ReplaceKeyWith	可在响应中返回的位置标头中使用的对象键。  如果提供了一个同级，则 ReplaceKeyWith 不是必需的。仅在不提供 ReplaceKeyPrefixWith 时提供它。
HttpRedirectCode	可在响应中返回的位置标头中使用的 HTTP 重定向代码。  如果提供了一个同级，则 HttpRedirectCode 不是必需的。

以下示例介绍了常见的重定向任务：

#### Example 1：保留键前缀后进行重定向

假设您的存储桶包含以下对象：

- index.html
- docs/article1.html
- docs/article2.html

您决定将文件夹从 `docs/` 重命名为 `documents/`。做出此更改后，您需要将对前缀 `docs/` 的请求重定向至 `documents/`。例如，对 `docs/article1.html` 的请求将重定向至 `documents/article1.html`。

在这种情况下，您可以以下路由规则添加到网站配置：

```
<RoutingRules>
  <RoutingRule>
    <Condition>
      <KeyPrefixEquals>docs/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
  </RoutingRule>
</RoutingRules>
```

#### Example 2：将对已删除文件夹的请求重定向到页面

假设您删除了 `images/` 文件夹（即，删除了具有键前缀 `images/` 的所有对象）。您可以添加路由规则以将具有键前缀 `images/` 的任何对象的请求重定向至名为 `folderdeleted.html` 的页。

```
<RoutingRules>
  <RoutingRule>
    <Condition>
      <KeyPrefixEquals>images/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
    </Redirect>
  </RoutingRule>
</RoutingRules>
```

#### Example 3：为 HTTP 错误进行重定向

假设在未找到请求的对象时，您需要将请求重定向至 Amazon Elastic Compute Cloud (Amazon EC2) 实例。添加重定向规则，以便当返回 HTTP 状态代码 404（未找到）时，站点访问者可重定向到将处理该请求的 Amazon EC2 实例。以下示例也将在重定向中插入对象键前缀 `report-404/`。例如，如果您请求了页面 `ExamplePage.html` 并且它导致了 HTTP 404 错误，该请求将重定向至指定 Amazon EC2 实例上的 `report-404/ExamplePage.html` 页面。如果没有路由规则且发生了 HTTP 错误 404，将返回配置中指定的错误文档。

```
<RoutingRules>
  <RoutingRule>
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
    </Condition>
    <Redirect>
      <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
```

```
<ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
</Redirect>
</RoutingRule>
</RoutingRules>
```

## 示例演练 – 在 Amazon S3 上托管网站

### 主题

- [示例：设置静态网站 \(p. 414\)](#)
- [示例：使用自定义域设置静态网站 \(p. 415\)](#)
- [示例：使用 Amazon CloudFront 为网站提速 \(p. 421\)](#)
- [清理示例资源 \(p. 423\)](#)

本节提供两个示例。在第一个示例中，您将为网站托管配置存储桶、上传示例索引文档，并使用存储桶的 Amazon S3 网站终端节点测试网站。第二个示例演示如何使用自己的域（如 example.com）而不使用 S3 存储桶网站终端节点，以及如何从配置为网站的 Amazon S3 存储桶提供内容。该示例还演示 Amazon S3 提供根域支持的方式。

### 示例：设置静态网站

您可以配置 Amazon S3 存储桶，使其具有与网站类似的功能。本示例将指导您完成在 Amazon S3 上托管网站的步骤。

### 主题

- [步骤 1：创建存储桶并将其配置为网站 \(p. 414\)](#)
- [步骤 2：添加可使您的存储桶内容公开可用的存储桶策略 \(p. 415\)](#)
- [步骤 3：上传索引文档 \(p. 415\)](#)
- [步骤 4：测试网站 \(p. 415\)](#)

### 步骤 1：创建存储桶并将其配置为网站

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 创建存储桶。

有关分步说明，请参阅[如何创建 S3 存储桶？](#)（在 Amazon Simple Storage Service 控制台用户指南中）。

有关存储桶命名指南，请参阅[存储桶限制 \(p. 53\)](#)。如果您拥有已注册的域名，请参阅[使用别名记录自定义 Amazon S3 URL \(p. 45\)](#)来获取有关存储桶命名的其他信息。

3. 打开存储桶的 Properties 窗格，选择 Static Website Hosting，然后执行以下操作：
  - a. 选择 Enable website hosting。
  - b. 在 Index Document 框中，键入索引文档的名称。该名称通常为 index.html。
  - c. 选择 Save 以保存网站配置。
  - d. 记下 Endpoint。

这是 Amazon S3 为存储桶提供的网站终端节点。您可以在以下步骤中使用此终端节点测试您的网站。

## 步骤 2：添加可使您的存储桶内容公开可用的存储桶策略

1. 在存储桶的 Properties 窗格中，选择 Permissions。
2. 选择 Add Bucket Policy。
3. 若要托管网站，您的存储桶必须具有公共读取访问权限。全世界的任何人都将可以读取此存储桶是刻意的设置。复制以下存储桶策略，然后将其粘贴到 Bucket Policy Editor 中。

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Sid": "PublicReadForGetBucketObjects",  
        "Effect": "Allow",  
        "Principal": "*",  
        "Action": ["s3:GetObject"],  
        "Resource": ["arn:aws:s3:::example-bucket/*"]  
    }]  
}
```

4. 在该策略中，将 *example-bucket* 替换为您的存储桶的名称。
5. 选择 Save (保存)。

## 步骤 3：上传索引文档

1. 创建文档。为此文档指定您之前为索引文档提供的相同名称。
2. 使用该控制台，将索引文档上传到您的存储桶。

有关说明，请参阅 Amazon Simple Storage Service 控制台用户指南 中的 [上传 S3 对象](#)。

## 步骤 4：测试网站

在浏览器中键入以下 URL，将 *example-bucket* 替换为您的存储桶名称，并将 *website-region* 替换为您在其中部署存储桶的 AWS 区域名称。有关 AWS 区域名称的信息，请参阅[网站终端节点 \(p. 402\)](#)。

```
http://example-bucket.s3-website-region.amazonaws.com
```

如果您的浏览器显示了 index.html 页面，则该网站已成功部署。

**Note**

不支持对该网站的 HTTPS 访问。

您现在已在 Amazon S3 上托管了一个网站。该网站在 Amazon S3 网站终端节点上可用。但是，您可能有要用来从已创建的网站提供内容的域（如 example.com）。您可能还需要使用 Amazon S3 的根域支持来响应对 http://www.example.com 和 http://example.com 的请求。此操作需要其他步骤。有关示例，请参阅[示例：使用自定义域设置静态网站 \(p. 415\)](#)。

## 示例：使用自定义域设置静态网站

假设您要在 Amazon S3 上托管您的静态网站。您注册了一个域（例如，example.com），并希望从 Amazon S3 内容响应对 http://www.example.com 和 http://example.com 的请求。您无论是想在 Amazon S3 上托管现有静态网站还是从头开始创建网站，都可以使用此示例了解如何在 Amazon S3 上托管网站。

**主题**

- [开始前的准备工作 \(p. 416\)](#)
- [步骤 1：注册域 \(p. 416\)](#)
- [步骤 2：创建和配置存储桶并上传数据 \(p. 416\)](#)
- [步骤 3：为 example.com 和 www.example.com 添加别名记录 \(p. 419\)](#)
- [步骤 4：测试 \(p. 421\)](#)

## 开始前的准备工作

在按照此示例中的步骤操作时，您将使用以下服务：

Amazon Route 53 – 您使用 Route 53 注册域，并定义要将您的域的 Internet 流量路由到何处。我们将介绍如何创建 Route 53 别名记录，以便将您的域 (example.com) 和子域 (www.example.com) 的流量路由到包含 HTML 文件的 Amazon S3 存储桶。

Amazon S3 – 您将使用 Amazon S3 创建存储桶，上传示例网页，配置权限以便每个人都可以查看内容，然后为网站托管配置存储桶。

### 步骤 1：注册域

如果您还没有已注册的域名（如 example.com），则可向 Route 53 注册一个域名。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的[注册新域](#)。注册域名之后，接下来的任务是创建和配置 Amazon S3 存储桶以实现网站托管并上传网站内容。

### 步骤 2：创建和配置存储桶并上传数据

要支持来自根域（例如 example.com）和子域（例如 www.example.com）的请求，需要创建两个存储桶。一个存储桶包含内容。另一个存储桶用来重定向请求。

#### 步骤 2.1：创建两个存储桶

存储桶名称必须与您托管的网站名称匹配。例如，要在 Amazon S3 上托管 example.com 网站，请创建名为 example.com 的存储桶。要在 www.example.com 下托管网站，您应将该存储桶命名为 www.example.com。在此示例中，您的网站将支持来自 example.com 和 www.example.com 的请求。

在此步骤中，您将使用 AWS 账户凭证登录 Amazon S3 控制台并创建以下两个存储桶。

- [example.com](#)
- [www.example.com](#)

#### Note

与域一样，子域也必须有它们自己的 S3 存储桶，且存储桶必须具有与子域完全一样的名称。在此示例中，我们创建了 www.example.com 子域，因此，我们需要将 S3 存储桶也命名为 www.example.com。

#### 创建存储桶并上传要托管的网站内容

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 创建两个与您的域名和子域相匹配的存储桶。例如，[example.com](#) 和 [www.example.com](#)。  
有关分步说明，请参阅[如何创建 S3 存储桶？](#)（在 Amazon Simple Storage Service 控制台用户指南中）。
3. 将您的网站数据上传到 [example.com](#) 存储桶。

您将在根域存储桶 ([example.com](#)) 之外托管您的内容，并且可以将对 [www.example.com](#) 的请求重定向到根域存储桶。您可以在任意一个存储桶中存储内容。在此示例中，您将在 [example.com](#) 存储桶中托管内容。该内容可以是文本文件、家人的照片、视频 – 任何您想要托管的内容。如果您还未创建网站，则只需为此示例创建一个文件。您可以上传任何文件。例如，您可以使用以下 HTML 创建文件，并将它上传到存储桶。网站主页的文件名通常为 index.html，但也可以命名为其他任何名称。在下一个步骤中，您将提供此文件名作为网站的索引文档名。

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>My Website Home Page</title>
</head>
<body>
    <h1>Welcome to my website</h1>
    <p>Now hosted on Amazon S3!</p>
</body>
</html>
```

有关分步说明，请参阅[如何将对象上传到 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南中)。

- 若要托管网站，您的存储桶必须具有公共读取访问权限。全世界的任何人都将可以读取此存储桶是刻意的设置。要授予公共读取访问权限，将以下存储桶策略附加到 [example.com](#) 存储桶，并将 [example.com](#) 替换为您的存储桶的名称。有关附加存储桶策略的分步说明，请参阅[如何添加 S3 存储桶策略？](#) (在 Amazon Simple Storage Service 控制台用户指南中)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": ["s3:GetObject"],
            "Resource": ["arn:aws:s3:::example.com/*"]
        }
    ]
}
```

现在，您拥有两个存储桶：[example.com](#) 和 [www.example.com](#)，并且您已将您的网站内容上传到了 [example.com](#) 存储桶。在下一步中，您将配置 [www.example.com](#) 以将请求重定向至 [example.com](#) 存储桶。通过重定向请求，您可以仅维护网站内容的一个副本。在浏览器中键入 www 的访客和仅指定根域的访客都将被路由到您的 [example.com](#) 存储桶中的相同网站内容。

## 步骤 2.2：为网站托管配置存储桶

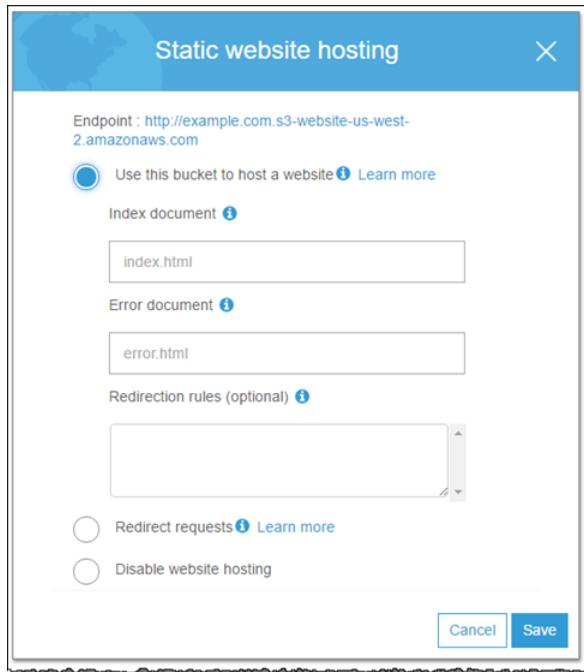
在为网站托管配置存储桶时，您可以使用 Amazon S3 分配的存储桶网站终端节点来访问网站。

在此步骤中，您将为网站托管配置这两个存储桶。首先，您将 [example.com](#) 配置为网站，然后配置 [www.example.com](#)，以将所有请求重定向到 [example.com](#) 存储桶。

### 为网站托管配置您的存储桶

- 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
- 在 Bucket name 列表中，选择要为其启用静态网站托管的存储桶的名称。
- 选择 Properties。
- 选择 Static website hosting。

5. 为网站托管配置 `example.com` 存储桶。在 Index Document 框中，键入您提供给索引页面的名称。



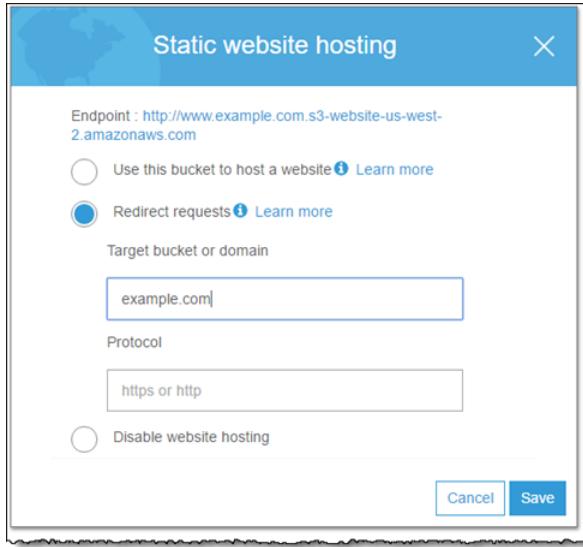
6. 选择 Save (保存)。

### 步骤 2.3：配置网站重定向

您已经配置用于网站托管的存储桶，接下来将配置 `www.example.com` 存储桶以便将对 `www.example.com` 所有请求重定向至 `example.com`。

#### 将请求从 `www.example.com` 重定向到 `example.com` 的步骤

1. 在 Amazon S3 控制台的 Buckets 列表中，选择您的存储桶（在此示例中为 `www.example.com`）。
2. 选择 Properties。
3. 选择 Static website hosting。
4. 选择 Redirect requests。在 Target bucket or domain 框中，键入 `example.com`。
5. 选择 Save (保存)。



## 步骤 2.4：配置网站流量的日志记录

(可选) 您可以配置日志记录以跟踪访问您的网站的访问者数量。为此，应对根域存储桶启用日志记录。有关更多信息，请参阅 [\(可选\) 配置 Web 流量日志记录 \(p. 406\)](#)。

## 步骤 2.5：测试您的终端节点和重定向

要测试网站，请在您的浏览器中键入终端节点的 URL。您的请求将被重定向，浏览器将显示 [example.com](http://example.com) 的索引文档。

接下来，您使用 Amazon Route 53 使客户能够使用所有这些 URL 导航到您的站点。

## 步骤 3：为 example.com 和 www.example.com 添加别名记录

在此步骤中，您创建别名记录并将其添加到您的域的托管区域，而这些别名记录将 [example.com](http://example.com) 和 [www.example.com](http://www.example.com) 映射到相应的 S3 存储桶。别名记录使用 Amazon S3 网站终端节点，而不使用 IP 地址。Amazon Route 53 在该别名记录与 Amazon S3 存储桶所在的 IP 地址之间保持映射。

### 将流量路由到您的网站

1. 通过以下网址打开 Route 53 控制台：[https://console.aws.amazon.com/route53/。](https://console.aws.amazon.com/route53/)
2. 在导航窗格中，选择 Hosted zones。

#### Note

当您注册域时，Amazon Route 53 将自动使用相同的名称创建一个托管区域。托管区域包含有关您希望 Route 53 如何路由域流量的信息。

3. 在托管区域列表中，选择您的域名。
4. 选择 Create Record Set。

#### Note

每个记录都包含有关您希望如何路由一个域 (example.com) 或子域 (www.example.com) 的流量的信息。记录存储在域的托管区域中。

5. 指定以下值：

Name

对于您将创建的第一个记录，接受默认值，该值为您的托管区域和您的域的名称。这会将 Internet 流量路由到与您的域同名的存储桶。

重复此步骤为子域创建第二条记录。对于第二个记录，键入 www. 这会将 Internet 流量路由至 www.*example.com* 存储桶的另一个别名记录。

类型

选择 A – IPv4 address。

别名

选择是。

别名目标

键入您创建 Amazon S3 存储桶所在的区域名称。使用 Amazon Web Services 一般参考 中 [AWS 区域和终端节点](#)一章的表 [Amazon Simple Storage Service 网站终端节点](#)中的 Website Endpoint 列的适用值。

Note

对于别名目标，为两个记录指定相同的值。Route 53 会根据记录的名称确定将流量路由到哪个存储桶。

路由策略

接受默认值 Simple。

Evaluate Target Health

接受默认值 No。

6. 选择 Create。
7. 对于 www.*example.com*，重复步骤 4 到步骤 6 以创建记录。

以下屏幕截图显示 *example.com* 的别名记录作为演示示例。您还需要为 www.*example.com* 创建别名记录。

The screenshot shows the AWS Route 53 service in the AWS Management Console. The left sidebar has 'Hosted zones' selected. The main area displays a table of record sets for the domain 'example.com'. There are two entries:

	Name	Type	Value
<input type="checkbox"/>	example.com.	NS	ns-165.awsdns-20.com. ns-1897.awsdns-45.co.uk. ns-1026.awsdns-00.org. ns-783.awsdns-33.net.
<input type="checkbox"/>	example.com.	SOA	ns-165.awsdns-20.com. awsdns-hostmaster.amazon. ns-165.awsdns-20.com. awsdns-22.com. awsdns-23.com. awsdns-24.com.

#### Note

创建、更改和删除资源记录集的操作需要一定时间才会传播到 Route 53 DNS 服务器。所做的更改通常在几分钟内传播到所有 Route 53 名称服务器。在极少的情况下，传播可能要用长达 30 分钟的时间。

## 步骤 4：测试

要验证该网站可以在您的浏览器中正常工作，请尝试以下 URL：

- <http://example.com> – 显示 `example.com` 存储桶中的索引文档。
- <http://www.example.com> – 将您的请求重定向到 <http://example.com>。

在某些情况下，您可能需要清除 Web 浏览器的缓存才能看到预期行为。

## 示例：使用 Amazon CloudFront 为网站提速

您可以使用 Amazon CloudFront 提高网站性能。CloudFront 可将您的网站文件（如 HTML、图像和视频）提供给全球各地的数据中心（称为边缘站点）使用。当访问者从您的网站请求文件时，CloudFront 会自动将请求重定向到最近边缘站点上的文件副本。这时的下载速度会比访问者从更远的数据中心请求该内容更快。

CloudFront 会将内容在边缘站点上缓存您指定的时间。如果访问者请求已过期的缓存内容，CloudFront 会检查来源服务器，确定该内容是否有更新的版本可用。如果有更新的版本，则 CloudFront 将新版本复制到该边缘站点。当访问者请求内容时，您对原始内容所做的更改便会复制到边缘站点。

要为您的网站提速，请使用 CloudFront 完成以下任务。

### 任务

- 创建 CloudFront 分配 (p. 422)
- 更新域和子域的记录集 (p. 422)
- (可选) 检查日志文件 (p. 423)

## 创建 CloudFront 分配

首先，您应创建 CloudFront 分配。这将使您的网站可供全球各地的数据中心使用。

### 使用 Amazon S3 源创建分配

1. 通过以下网址打开 CloudFront 控制台：<https://console.aws.amazon.com/cloudfront/>。
2. 选择 Create Distribution。
3. 在 Select a delivery method for your content 页上，为 Web 选择 Get Started。
4. 在 Create Distribution 页面上的 Origin Settings 部分中，为 Origin Domain Name 键入您的存储桶的 Amazon S3 静态网站托管终端节点。例如：`example.com.amazonaws.com`。

CloudFront 将为您填写 Origin ID。

5. 对于 Default Cache Behavior Settings，将值保留为默认值。更多有关这些配置选项的信息，请参阅 Amazon CloudFront 开发人员指南 中的 [您创建或更新 Web 分配时指定的值](#)。
6. 对于 Distribution Settings，执行以下操作：
  - a. 将 Price Class (价格级别) 的设置保留为 Use All Edge Locations (Best Performance) (使用所有节点 (最佳性能))。
  - b. 将 Alternate Domain Names (CNAMEs) 设为根域和 www 子域；在本教程中分别为 `example.com` 和 `www.example.com`。必须先设置这些值，然后再为将指定域名连接到 CloudFront 分配的 A 记录创建别名。
  - c. 将 Default Root Object (默认根对象) 设置为 `index.html`。如果用于访问分配的 URL 不包含文件名，这将是 CloudFront 分配返回的默认页面。此值应该与您在 [为网站托管配置存储桶 \(p. 403\)](#) 中设置的索引文档值匹配。
  - d. 将 Logging (日志记录) 设置为 On (打开)。
  - e. 对于 Bucket for Logs，选择您创建的日志记录存储桶。
  - f. 要将由流量生成的日志存储到日志存储桶中名为 `cdn` 的文件夹的 CloudFront 分配，请在 Log Prefix 中键入 `cdn/`。
  - g. 将其他设置保留为默认值。
7. 选择 Create Distribution。

要查看分配的状态，请在控制台中找到该分配，然后检查 Status 列。`InProgress` (进行中) 状态表示分配尚未完成部署。

分配部署完毕后，您可以使用新的 CloudFront 域名来引用您的内容。记录 CloudFront 控制台中显示的 Domain Name 值。下一步中需要使用该值。在本示例中，该值为 `dj4p1rv6mvubz.cloudfront.net`。

要验证您的 CloudFront 分配是否正常运行，请在 Web 浏览器中键入该分配的域名。如果正常运行，则您的网站可见。

## 更新域和子域的记录集

现在您已成功创建 CloudFront 分配，下一步将更新 Route 53 中的 A 记录，以指向新的 CloudFront 分配。

### 更新 A 记录以指向 CloudFront 分配

1. 通过以下网址打开 Route 53 控制台：<https://console.aws.amazon.com/route53/>。

2. 在 Hosted Zones 页上，选择为您的域创建的托管区域。
3. 选择 Go to Record Sets。
4. 选择您为 www 子域创建的 A 记录。
5. 对于 Alias Target，选择 CloudFront 分配。
6. 选择 Save Record Set。
7. 要将根域的 A 记录重定向到 CloudFront 分配，请重复此过程。

记录集的更新需要 2 到 48 小时生效。要查看新的 A 记录是否已生效，请在 Web 浏览器中键入 `http://www.example.com`。如果您的浏览器不再重定向至 `http://example.com`，则说明新的 A 记录已生效。

出现这种行为变化的原因是，由旧 A 记录传输到子域 S3 存储桶的流量，被 Amazon S3 中的设置重定向到了根域。www 当新 A 记录生效时，由新 A 记录传输到 CloudFront 分配的流量不会被重定向至根域。

**Tip**

浏览器可以缓存重定向设置。如果您认为新的 A 记录设置应该已经生效，但是您的浏览器仍然将 `http://www.example.com` 重定向至 `http://example.com`，请尝试清除浏览器的历史记录和缓存，然后关闭再重新打开浏览器应用程序，或使用其他 Web 浏览器。

在新的 A 记录生效后，任何使用 `http://example.com` 或 `http://www.example.com` 引用该网站的访问者都会重定向至最近的 CloudFront 边缘站点，并在这里体验更快的下载。

如果您仅出于练习目的创建网站，则可以删除您所分配的资源，使其不再产生费用。为此，继续执行 [清理示例资源 \(p. 423\)](#)。删除 AWS 资源后，您的网站将不再可用。

## (可选) 检查日志文件

访问日志会告诉您有多少人正在访问网站。它们还包含有价值的业务数据，您可以使用 [Amazon EMR](#) 等其他服务分析这些数据。

在您的存储桶中，较旧的 Amazon S3 日志文件位于 `root` 文件夹中。所有新日志文件（应该是 CloudFront 日志）均位于 `cdn` 文件夹中。Amazon S3 每两小时将网站访问日志写入您的日志存储桶一次。CloudFront 将在相应请求提出后的 24 小时内将日志写入您的日志存储桶。

### 查看网站的日志文件

1. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 为网站选择日志记录存储桶。
3. 要查看存储在 `cdn` 或 `root` 文件夹中的日志文件，请选择 `cdn` 或 `root`。
4. 在浏览器中打开 Amazon S3 日志文件（它们是文本文件）。先下载由 CloudFront 编写的 `.gzip` 文件，然后再打开这些文件。

## 清理示例资源

如果您仅出于练习目的创建静态网站，请务必删除您分配的 AWS 资源，使其不再产生费用。删除 AWS 资源后，您的网站将不再可用。

### 任务

- [删除 Amazon CloudFront 分配 \(p. 424\)](#)
- [删除 Route 53 托管区域 \(p. 424\)](#)
- [删除 S3 存储桶 \(p. 424\)](#)

## 删除 Amazon CloudFront 分配

删除 Amazon CloudFront 分配之前，须先将其禁用。已禁用的分配不再起作用，并且不会产生费用。您可以随时启用已禁用的分配。已禁用的分配在删除后将不再可用。

### 禁用和删除 CloudFront 分配

1. 通过以下网址打开 CloudFront 控制台：<https://console.aws.amazon.com/cloudfront/>。
2. 选择要禁用的分配，然后选择 Disable。
3. 当系统提示确认时，选择 Yes, Disable。
4. 选择禁用的分配，然后选择 Delete。
5. 当系统提示进行确认时，选择 Yes, Delete。

## 删除 Route 53 托管区域

在删除托管区域之前，您必须先删除已创建的记录集。您不需要删除 NS 和 SOA 记录；删除托管区域时，会自动删除这些记录。

### 删除记录集

1. 通过以下网址打开 Route 53 控制台：<https://console.aws.amazon.com/route53/>。
2. 在域名列表中，选择您的域名，然后选择 Go to Record Sets。
3. 在记录集列表中，选择您创建的 A 记录。每个记录集的类型均列在 Type (类型) 列中。
4. 选择 Delete Record Set。
5. 当系统提示进行确认时，选择 Confirm。

### 删除 Route 53 托管区域

1. 紧接上一个步骤，选择 Back to Hosted Zones。
2. 选择您的域名，然后选择 Delete Hosted Zone。
3. 当系统提示进行确认时，选择 Confirm。

## 删除 S3 存储桶

在删除您的 S3 存储桶之前，请确保已禁用该存储桶的日志记录。否则，在您删除存储桶时，AWS 会继续将日志写入其中。

### 禁用存储桶的日志记录

1. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 选择存储桶，然后选择 Properties。
3. 从 Properties 中选择 Logging。
4. 清除 Enabled (启用) 复选框。
5. 选择 Save (保存)。

现在您可以删除存储桶。有关更多信息，请参阅[如何删除 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

# 配置 Amazon S3 事件通知

通过 Amazon S3 通知功能，您可以在存储桶中发生某些事件时接收通知。要启用通知，您必须首先添加通知配置，标识希望 Amazon S3 发布的事件以及希望 Amazon S3 将事件通知发送到的目的地。您可以将此配置存储在与存储桶关联的通知子资源中（请参阅 [存储桶配置选项 \(p. 51\)](#)）。Amazon S3 提供了一个 API，用于管理此子资源。

## 主题

- [概述 \(p. 425\)](#)
- [如何启用事件通知 \(p. 426\)](#)
- [事件通知类型和目标 \(p. 427\)](#)
- [使用对象键名称筛选配置通知 \(p. 428\)](#)
- [授予将事件通知消息发布到目标的权限 \(p. 432\)](#)
- [示例演练 1：为存储桶配置通知（消息目标：SNS 主题和 SQS 队列）\(p. 434\)](#)
- [示例演练 2：为存储桶配置通知（消息目标：AWS Lambda）\(p. 439\)](#)
- [事件消息结构 \(p. 439\)](#)

## 概述

目前，Amazon S3 可以发布以下事件：

- 新对象创建事件 - Amazon S3 支持多个创建对象的 API。您可以在仅使用特定 API（例如 `s3:ObjectCreated:Put`）时请求通知，或在创建对象时使用通配符（例如 `s3:ObjectCreated:*`）请求通知，而不管使用的 API 是什么。
- 对象删除事件 - Amazon S3 支持对受版本控制和不受版本控制的对象的删除。有关对象版本控制的更多信息，请参阅[对象版本控制 \(p. 94\)](#)和[使用版本控制 \(p. 380\)](#)。

您可以使用 `s3:ObjectRemoved:Delete` 事件类型请求在删除对象或永久删除受版本控制的对象时收到通知。或者，也可以使用 `s3:ObjectRemoved:DeleteMarkerCreated` 请求在为受版本控制的对象创建删除标记时收到通知。您还可以使用通配符 `s3:ObjectRemoved:*` 来请求在每次删除对象时收到通知。有关删除受版本控制的对象的信息，请参阅[删除数据元版本 \(p. 392\)](#)。

- 低冗余存储 (RRS) 对象丢失事件 - Amazon S3 在检测到 RRS 存储类别的对象已丢失时发送通知消息。

有关受支持的事件类型的列表，请参阅[受支持的事件类型 \(p. 427\)](#)。

Amazon S3 支持它可以在其中发布事件的以下目标：

- Amazon Simple Notification Service (Amazon SNS) 主题

Amazon SNS 是一项灵活且完全托管的消息推送服务。使用此服务，您可以将消息推送到移动设备或分布式服务。通过 SNS，您只需要发布一次消息，就能将其发送无数遍。SNS 主题是收件人可动态订阅以接收事件通知的接入点。有关 SNS 的更多信息，请参阅 [Amazon SNS 产品详细信息页](#)。

- Amazon Simple Queue Service (Amazon SQS) 队列

Amazon SQS 是一项可扩展且完全托管的消息排队服务。您可以使用 SQS 来传输任何容量的数据，而不需要其他服务始终可用。在您的通知配置中，您可以请求 Amazon S3 将事件发布到 SQS 队列。有关 SQS 的更多信息，请参阅 [Amazon SQS 产品详细信息页](#)。

- AWS Lambda

AWS Lambda 是一项计算服务，它使您可以轻松地构建快速响应新信息的应用程序。AWS Lambda 运行您的代码以响应事件，例如图像上传、应用程序内活动、网站单击或连接设备的输出。您可以使用 AWS Lambda 通过自定义逻辑扩展其他 AWS 服务，或创建您自己的按 AWS 规模、性能和安全性运行的后端。通过 AWS Lambda，您可以轻松创建单独的事件驱动的应用程序，这些应用程序仅在需要时执行并可自动从每天几个请求扩展到每秒数千个请求。

AWS Lambda 可以运行自定义代码以响应 Amazon S3 存储桶事件。您将自定义代码上传到 AWS Lambda 并创建所谓的 Lambda 函数。当 Amazon S3 检测到特定类型的事件（例如，对象创建的事件）时，它可以把该事件发布到 AWS Lambda 并在 Lambda 中调用您的函数。作为响应，AWS Lambda 将执行您的函数。有关更多信息，请参阅 [AWS Lambda 产品详细信息页](#)。

以下部分提供有关如何在存储桶上启用事件通知的更多详细信息。副主题还提供了示例演练以帮助您探索通知功能。

- [示例演练 1：为存储桶配置通知（消息目标：SNS 主题和 SQS 队列）\(p. 434\)](#)
- [示例演练 2：为存储桶配置通知（消息目标：AWS Lambda）\(p. 439\)](#)

## 如何启用事件通知

启用通知是存储桶级别的操作；即，您将通知配置信息存储在与存储桶关联的通知子资源中。您可以使用以下任意方法来管理通知配置：

- 使用 Amazon S3 控制台

控制台 UI 允许您在存储桶上设置通知配置，而无需编写任何代码。有关说明，请参阅[如何为 S3 存储桶启用和配置事件通知？](#)（在 Amazon Simple Storage Service 控制台用户指南中）。

- 使用 AWS 开发工具包以编程方式管理

### Note

如果需要，也可以直接从代码中调用 Amazon S3 REST API。但是，这可能会比较繁琐，因为需要您编写代码对请求进行身份验证。

在内部，控制台和开发工具包都调用 Amazon S3 REST API 来管理与存储桶关联的通知子资源。有关使用 AWS 开发工具包管理通知配置的示例，请参阅前一部分中提供的演练链接。

无论您使用哪种方法，Amazon S3 都将通知配置作为 XML 存储在与存储桶关联的通知子资源中。有关存储桶子资源的信息，请参阅[存储桶配置选项 \(p. 51\)](#)。默认情况下，不为任何类型的事件启用通知。因此，通知子资源最初存储空配置。

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</NotificationConfiguration>
```

要为特定类型的事件启用通知，请将 XML 替换为适当的配置，该配置可标识您希望 Amazon S3 发布的事件类型和您希望将事件发布到的目标。对于每个目标，添加相应的 XML 配置。例如：

- 将事件消息发布到 SQS 队列 — 要将 SQS 队列设置为一个或多个事件类型的通知目标，请添加 QueueConfiguration。

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
```

```
<Event>event-type</Event>
...
</QueueConfiguration>
...
</NotificationConfiguration>
```

- 将事件消息发布到 SNS 主题 - 要将 SNS 主题设置为特定事件类型的通知目标，请添加 TopicConfiguration。

```
<NotificationConfiguration>
  <TopicConfiguration>
    <Id>optional-id-string</Id>
    <Topic>sns-topic-arn</Topic>
    <Event>event-type</Event>
    <Event>event-type</Event>
    ...
  </TopicConfiguration>
  ...
</NotificationConfiguration>
```

- 调用 AWS Lambda 函数并提供一个事件消息作为参数 - 要将 Lambda 函数设置为特定事件类型的通知目标，请添加 CloudFunctionConfiguration。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
    <Id>optional-id-string</Id>
    <Cloudcode>cloud-function-arn</Cloudcode>
    <Event>event-type</Event>
    <Event>event-type</Event>
    ...
  </CloudFunctionConfiguration>
  ...
</NotificationConfiguration>
```

要移除存储桶上配置的所有通知，请在通知子资源中保存一个空 `<NotificationConfiguration/>` 元素。

当 Amazon S3 检测到特定类型的事件时，它会发布包含事件信息的消息。有关更多信息，请参阅 [事件消息结构 \(p. 439\)](#)。

## 事件通知类型和目标

本部分介绍 Amazon S3 支持的事件通知类型和可在其中发布通知的目标的类型。

### 受支持的事件类型

Amazon S3 可以发布以下类型的事件。您在通知配置中指定这些事件类型。

事件类型	描述
s3:ObjectCreated:*	诸如 PUT、POST 和 COPY 之类的 Amazon S3 API 可以创建对象。使用这些事件类型，您可以在使用特定 API 创建对象时启用通知，也可以使用 s3:ObjectCreated:* 事件类型请求通知，而无论使用什么 API 创建对象。
s3:ObjectCreated:Put	
s3:ObjectCreated:Post	您不会从失败的操作收到事件通知。

事件类型	描述
s3:ObjectCreated:Copy	
s3:ObjectCreated:CompleteMultipartUpload	
s3:ObjectRemoved:*	通过使用 ObjectRemoved 事件类型，您可以在从存储桶中删除一个对象或一批对象时启用通知。
s3:ObjectRemoved:Delete	
s3:ObjectRemoved:DeleteMarkerCreated	您可以使用 s3:ObjectRemoved:Delete 事件类型请求在删除对象或永久删除受版本控制的对象时收到通知。或者，也可以使用 s3:ObjectRemoved:DeleteMarkerCreated 请求在为受版本控制的对象创建删除标记时收到通知。有关删除受版本控制的对象的信息，请参阅 <a href="#">删除数据元版本 (p. 392)</a> 。您还可以使用通配符 s3:ObjectRemoved:/* 来请求在每次删除对象时收到通知。
	您不会从生命周期策略中的自动删除或失败的操作收到事件通知。
s3:ReducedRedundancyLostObject	您可以使用此事件类型来请求 Amazon S3 在 Amazon S3 检测到 RRS 存储类别的对象丢失时发送通知消息。

## 受支持的目标

Amazon S3 可以将事件通知消息发送到以下目标。您在通知配置中指定这些目标的 ARN 值。

- 将事件消息发布到 Amazon Simple Notification Service (Amazon SNS) 主题
- 将事件消息发布到 Amazon Simple Queue Service (Amazon SQS) 队列

### Note

如果目标队列启用了 SSE，则 Amazon S3 将需要访问相关的 KMS 密钥来启用消息加密。

- 通过调用 Lambda 函数并提供事件消息作为参数来将事件消息发布到 AWS Lambda

您必须授予 Amazon S3 将消息发布到 Amazon SNS 主题或 Amazon SQS 队列的权限。您还必须授予 Amazon S3 权限以代表您调用 AWS Lambda 函数。有关授予这些权限的信息，请参阅[授予将事件通知消息发布到目标的权限 \(p. 432\)](#)。

## 使用对象键名称筛选配置通知

您可以将通知配置为按对象的键名称的前缀和后缀进行筛选。例如，您可以设置一个配置，以便仅在将带有“.jpg”扩展名的图像文件添加到存储桶时收到通知。或者，您也可以设置一个配置，该配置仅在将带有前缀“images/”的对象添加到存储桶时将通知发送到 Amazon SNS 主题，同时将同一存储桶中带有前缀“logs/”的对象的通知发送到 AWS Lambda 函数。

您可以在 Amazon S3 控制台中以及通过 AWS 开发工具包使用 Amazon S3 API 或直接使用 REST API，设置使用对象键名称筛选的通知配置。有关使用控制台 UI 在存储桶上的设置通知配置的信息，请参阅[如何为 S3 存储桶启用和配置事件通知？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

Amazon S3 在与存储桶关联的通知子资源中以 XML 形式存储通知配置，如[如何启用事件通知 \(p. 426\)](#)中所述。您使用 Filter XML 结构定义要按对象键名称的前缀和后缀进行筛选的通知的规则。有关 Filter XML 结构的详细信息，请参阅 Amazon Simple Storage Service API Reference 中的 [PUT Bucket 通知](#)。

使用 Filter 的通知配置无法定义采用重叠前缀、重叠后缀或前缀和后缀重叠的筛选规则。以下部分包含采用对象键名称筛选的有效通知配置的示例以及因前缀/后缀重叠而失效的通知配置的示例。

## 采用对象键名称筛选的有效通知配置的示例

以下通知配置包含用于标识 Amazon S3 的 Amazon SQS 队列以发布 s3:ObjectCreated:Put 类型的事件的队列配置。每当具有前缀 images/ 和后缀 jpg 的对象被放置 (PUT) 到存储桶时，就会发布这些事件。

```
<NotificationConfiguration>
<QueueConfiguration>
<Id>1</Id>
<Filter>
<S3Key>
<FilterRule>
<Name>prefix</Name>
<Value>images/</Value>
</FilterRule>
<FilterRule>
<Name>suffix</Name>
<Value>.jpg</Value>
</FilterRule>
</S3Key>
</Filter>
<Queue>arn:aws:sqs:us-west-2:444455556666:s3notificationqueue</Queue>
<Event>s3:ObjectCreated:Put</Event>
</QueueConfiguration>
</NotificationConfiguration>
```

以下通知配置有多个非重叠前缀。该配置做出以下定义：images/ 文件夹中针对 PUT 请求的通知将进入队列 A，logs/ 文件夹中针对 PUT 请求的通知将进入队列 B。

```
<NotificationConfiguration>
<QueueConfiguration>
<Id>1</Id>
<Filter>
<S3Key>
<FilterRule>
<Name>prefix</Name>
<Value>images/</Value>
</FilterRule>
</S3Key>
</Filter>
<Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-A</Queue>
<Event>s3:ObjectCreated:Put</Event>
</QueueConfiguration>
<QueueConfiguration>
<Id>2</Id>
<Filter>
<S3Key>
<FilterRule>
<Name>prefix</Name>
<Value>logs/</Value>
</FilterRule>
</S3Key>
</Filter>
<Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-B</Queue>
<Event>s3:ObjectCreated:Put</Event>
</QueueConfiguration>
</NotificationConfiguration>
```

以下通知配置有多个非重叠后缀。该配置做出以下定义：所有新添加到存储桶的.jpg 图像都由 Lambda 云函数 A 处理，所有新添加的.png 图像都由云函数 B 处理。后缀 .png 和 .jpg 是不重叠的，即使它们的最后一个字母相同。如果某个给定的字符串能够以这两个后缀结尾，则将它们视为重叠。一个字符串无法同时以 .png 和 .jpg 结尾，因此示例配置中的后缀不是重叠后缀。

```
<NotificationConfiguration>
<CloudFunctionConfiguration>
<Id>1</Id>
<Filter>
    <S3Key>
        <FilterRule>
            <Name>suffix</Name>
            <Value>.jpg</Value>
        </FilterRule>
    </S3Key>
</Filter>
<Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</Cloudcode>
<Event>s3:ObjectCreated:Put</Event>
</CloudFunctionConfiguration>
<CloudFunctionConfiguration>
<Id>2</Id>
<Filter>
    <S3Key>
        <FilterRule>
            <Name>suffix</Name>
            <Value>.png</Value>
        </FilterRule>
    </S3Key>
</Filter>
<Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</Cloudcode>
<Event>s3:ObjectCreated:Put</Event>
</CloudFunctionConfiguration>
</NotificationConfiguration>
```

使用 Filter 的通知配置不能使用重叠前缀为相同的事件类型定义筛选规则，除非重叠前缀与不重叠的后缀一起使用。以下示例配置显示了如何将使用通用前缀和非重叠后缀创建的对象传递到其他目标。

```
<NotificationConfiguration>
<CloudFunctionConfiguration>
<Id>1</Id>
<Filter>
    <S3Key>
        <FilterRule>
            <Name>prefix</Name>
            <Value>images</Value>
        </FilterRule>
        <FilterRule>
            <Name>suffix</Name>
            <Value>.jpg</Value>
        </FilterRule>
    </S3Key>
</Filter>
<Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</Cloudcode>
<Event>s3:ObjectCreated:Put</Event>
</CloudFunctionConfiguration>
<CloudFunctionConfiguration>
<Id>2</Id>
<Filter>
    <S3Key>
        <FilterRule>
            <Name>prefix</Name>
            <Value>images</Value>
        </FilterRule>
        <FilterRule>
            <Name>suffix</Name>
            <Value>.png</Value>
        </FilterRule>
    </S3Key>
</Filter>
```

```
<Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</Cloudcode>
<Event>s3:ObjectCreated:Put</Event>
</CloudFunctionConfiguration>
</NotificationConfiguration>
```

## 采用无效前缀/后缀重叠的通知配置的示例

在大多数情况下，使用 Filter 的通知配置无法使用重叠前缀、重叠后缀或前缀和后缀的重叠组合为相同的事件类型定义筛选规则。(只要后缀不重叠，您就可以采用重叠前缀。有关示例，请参阅 [使用对象键名称筛选配置通知 \(p. 428\)](#)。)

您可以将重叠对象键名称筛选用于不同的事件类型。例如，您可以创建一个通知配置，该配置将前缀 image/ 用于 ObjectCreated:Put 事件类型并将前缀 image/ 用于 ObjectDeleted:\* 事件类型。

在使用 AWS Amazon S3 控制台或 Amazon S3 API 时，如果您尝试保存具有对相同活动类型无效的重叠名称筛选的通知配置，则会收到错误。此部分显示了因重叠名称筛选而失效的通知配置的示例。

任何现有通知配置规则都被假定为具有与任何其他前缀和后缀分别匹配的默认前缀和后缀。以下通知配置因具有重叠前缀而失效，其中的根前缀与任何其他前缀重叠。(如果我们在此示例中使用后缀而不是前缀，则会发生同样的情况。根后缀与任何其他后缀重叠。)

```
<NotificationConfiguration>
  <TopicConfiguration>
    <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-one</Topic>
    <Event>s3:ObjectCreated:*</Event>
  </TopicConfiguration>
  <TopicConfiguration>
    <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-two</Topic>
    <Event>s3:ObjectCreated:*</Event>
    <Filter>
      <S3Key>
        <FilterRule>
          <Name>prefix</Name>
          <Value>images</Value>
        </FilterRule>
      </S3Key>
    </Filter>
  </TopicConfiguration>
</NotificationConfiguration>
```

由于具有重叠后缀，以下通知配置无效。如果某个给定的字符串能够以这两个后缀结尾，则将它们视为重叠。一个字符串能够以 jpg 和 pg 结尾，因此后缀是重叠的。(对于前缀，此结论也成立，如果给定字符串能够以两个前缀开头，则认为它们是重叠的。)

```
<NotificationConfiguration>
  <TopicConfiguration>
    <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
    <Event>s3:ObjectCreated:*</Event>
    <Filter>
      <S3Key>
        <FilterRule>
          <Name>suffix</Name>
          <Value>jpg</Value>
        </FilterRule>
      </S3Key>
    </Filter>
  </TopicConfiguration>
  <TopicConfiguration>
    <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
    <Event>s3:ObjectCreated:Put</Event>
```

```
<Filter>
  <S3Key>
    <FilterRule>
      <Name>suffix</Name>
      <Value>pg</Value>
    </FilterRule>
  </S3Key>
</Filter>
</TopicConfiguration>
</NotificationConfiguration>
```

由于具有重叠前缀和后缀，以下通知配置无效。

```
<NotificationConfiguration>
  <TopicConfiguration>
    <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
    <Event>s3:ObjectCreated:*</Event>
    <Filter>
      <S3Key>
        <FilterRule>
          <Name>prefix</Name>
          <Value>images</Value>
        </FilterRule>
        <FilterRule>
          <Name>suffix</Name>
          <Value>jpg</Value>
        </FilterRule>
      </S3Key>
    </Filter>
  </TopicConfiguration>
  <TopicConfiguration>
    <Topic>arn:aws:snsus-west-2:444455556666:sns-topic-two</Topic>
    <Event>s3:ObjectCreated:Put</Event>
    <Filter>
      <S3Key>
        <FilterRule>
          <Name>suffix</Name>
          <Value>jpg</Value>
        </FilterRule>
      </S3Key>
    </Filter>
  </TopicConfiguration>
</NotificationConfiguration>
```

## 授予将事件通知消息发布到目标的权限

在 Amazon S3 可以将消息发布到目标之前，您必须授予 Amazon S3 委托人调用相关 API 以将消息发布到 SNS 主题、SQS 队列或 Lambda 函数所需的权限。

### 授予调用 AWS Lambda 函数的权限

Amazon S3 通过调用 Lambda 函数并提供事件消息作为参数来将事件消息发布到 AWS Lambda。

使用 Amazon S3 控制台在 Amazon S3 存储桶上为 Lambda 函数配置事件通知时，Amazon S3 控制台将在 Lambda 函数上设置必要的权限以便 Amazon S3 有权从存储桶调用函数。有关更多信息，请参阅[如何为 S3 存储桶启用和配置事件通知？](#)（在 Amazon Simple Storage Service 控制台用户指南中）。

您还可以从 AWS Lambda 授予 Amazon S3 权限以调用您的 Lambda 函数。有关更多信息，请参阅 AWS Lambda Developer Guide 中的[教程：将 AWS Lambda 与 Amazon S3 一起使用](#)。

## 授予将消息发布到 SNS 主题或 SQS 队列的权限

将 IAM 策略附加到目标 SNS 主题或 SQS 队列以授予 Amazon S3 将消息发布到 SNS 主题或 SQS 队列的权限。

附加到目标 SNS 主题的 IAM 策略示例。

```
{  
    "Version": "2008-10-17",  
    "Id": "example-ID",  
    "Statement": [  
        {  
            "Sid": "example-statement-ID",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "s3.amazonaws.com"  
            },  
            "Action": [  
                "SNS:Publish"  
            ],  
            "Resource": "SNS-ARN",  
            "Condition": {  
                "ArnLike": { "aws:SourceArn": "arn:aws:s3:::bucket-name" }  
            }  
        }  
    ]  
}
```

附加到目标 SQS 队列的 IAM 策略示例。

```
{  
    "Version": "2008-10-17",  
    "Id": "example-ID",  
    "Statement": [  
        {  
            "Sid": "example-statement-ID",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "*"  
            },  
            "Action": [  
                "SQS:SendMessage"  
            ],  
            "Resource": "SQS-ARN",  
            "Condition": {  
                "ArnLike": { "aws:SourceArn": "arn:aws:s3:::bucket-name" }  
            }  
        }  
    ]  
}
```

请注意，对于 Amazon SNS 和 Amazon SQS IAM 策略，您可以在策略中指定 StringLike 条件而不是 ArnLike 条件。

```
"Condition": {  
    "StringLike": { "aws:SourceArn": "arn:aws:s3:::bucket-name" }  
}
```

在 SQS 队列启用了 SSE 的情况下附加到相关的 KMS 密钥的密钥策略的示例。

```
{  
    "Version": "2012-10-17",  
    "Id": "example-ID",  
    "Statement": [  
        {  
            "Sid": "example-statement-ID",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "s3.amazonaws.com"  
            },  
            "Action": [  
                "kms:GenerateDataKey",  
                "kms:Decrypt"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

该策略授予 Amazon S3 服务委托方对添加到队列的消息进行加密所需的特定 KMS 操作的权限。

有关如何将策略附加到 SNS 主题或 SQS 队列的示例，请参阅 [示例演练 1：为存储桶配置通知（消息目标：SNS 主题和 SQS 队列）\(p. 434\)](#)。

有关权限的更多信息，请参阅以下主题：

- Amazon Simple Notification Service 开发人员指南 中的 [Amazon SNS 访问控制的示例案例](#)
- Amazon Simple Queue Service 开发人员指南 中的 [使用 AWS Identity and Access Management \(IAM\) 进行访问控制](#)

## 示例演练 1：为存储桶配置通知（消息目标：SNS 主题和 SQS 队列）

### 主题

- [演练摘要 \(p. 434\)](#)
- [步骤 1：创建 Amazon SNS 主题 \(p. 435\)](#)
- [步骤 2：创建 Amazon SQS 队列 \(p. 436\)](#)
- [步骤 3：将通知配置添加到存储桶 \(p. 437\)](#)
- [步骤 4：测试设置 \(p. 439\)](#)

## 演练摘要

在此演练中，您在存储桶上添加请求 Amazon S3 执行以下操作的通知配置：

- 将 s3:ObjectCreated:\* 类型的事件发布到 Amazon SQS 队列。
- 将 s3:ReducedRedundancyLostObject 类型的事件发布到 Amazon SNS 主题。

有关通知配置的信息，请参阅 [配置 Amazon S3 事件通知 \(p. 425\)](#)。

您可以使用控制台执行这些步骤，无需编写任何代码。此外，还提供了使用适用于 Java 和 .NET 的 AWS 开发工具包的代码示例，因此您可以通过编程方式添加通知配置。

您将在此演练中执行以下操作：

1. 创建一个 Amazon SNS 主题。

您可使用 Amazon SNS 控制台创建 SNS 主题并订阅该主题，以便发布到此主题的所有事件都传送给您。您将指定电子邮件作为通信协议。在创建主题后，Amazon SNS 会发送电子邮件。您必须单击电子邮件中的链接以确认订阅主题。

您可将访问策略附加到此主题以授予 Amazon S3 发布消息的权限。

2. 创建 Amazon SQS 队列。

使用 Amazon SQS 控制台，可创建 SQS 队列。您可以通过编程方式访问 Amazon S3 发送到此队列的所有消息。但对于此演练，您将在控制台中验证通知消息。

您可将访问策略附加到此主题以授予 Amazon S3 发布消息的权限。

3. 将通知配置添加到存储桶。

## 步骤 1：创建 Amazon SNS 主题

按照以下步骤创建并订阅 Amazon Simple Notification Service (Amazon SNS) 主题。

1. 使用 Amazon SNS 控制台创建主题。有关说明，请参阅 Amazon Simple Notification Service 开发人员指南 中的[创建主题](#)。
2. 订阅至主题。对于此练习，请使用电子邮件作为通信协议。有关说明，请参阅 Amazon Simple Notification Service 开发人员指南 中的[订阅主题](#)。

您将收到电子邮件要求您确认订阅该主题。确认订阅。

3. 使用以下策略替换附加到主题的访问策略。您必须通过提供您的 SNS 主题 ARN 和存储桶名称来更新策略：

```
{  
    "Version": "2008-10-17",  
    "Id": "example-ID",  
    "Statement": [  
        {  
            "Sid": "example-statement-ID",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "*"  
            },  
            "Action": [  
                "SNS:Publish"  
            ],  
            "Resource": "SNS-topic-ARN",  
            "Condition": {  
                "ArnLike": { "aws:SourceArn": "arn:aws:s3:::bucket-name" }  
            }  
        }  
    ]  
}
```

4. 记录主题 ARN。

您创建的 SNS 主题是您的 AWS 账户中的另一项资源，它有唯一的 Amazon 资源名称 (ARN)。在下一步骤中，您需要用到此 ARN。ARN 格式如下：

```
arn:aws:sns:aws-region:account-id:topic-name
```

## 步骤 2：创建 Amazon SQS 队列

按照以下步骤创建并订阅 Amazon Simple Queue Service (Amazon SQS) 队列。

1. 使用 Amazon SQS 控制台创建队列。有关说明，请参阅 Amazon Simple Queue Service 开发人员指南中的 [Amazon SQS 入门](#)。
2. 使用以下策略替换附加到队列的访问策略 (在 SQS 控制台中，选择队列，然后在 Permissions 选项卡中，单击 Edit Policy Document (Advanced))。

```
{  
    "Version": "2012-10-17",  
    "Id": "example-ID",  
    "Statement": [  
        {  
            "Sid": "example-statement-ID",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "*"  
            },  
            "Action": [  
                "SQS:SendMessage"  
            ],  
            "Resource": "SQS-queue-ARN",  
            "Condition": {  
                "ArnLike": { "aws:SourceArn": "arn:aws:s3:::bucket-name" }  
            }  
        }  
    ]  
}
```

3. (可选) 如果 Amazon SQS 队列已启用服务器端加密 (SSE)，则可将以下策略添加到关联的自定义 AWS Key Management Service (AWS KMS) 客户主密钥 (CMK)。您必须将策略添加到自定义 CMK，因为无法修改 Amazon SQS 的默认 AWS 托管 CMK。有关将 SSE 用于 Amazon SQS 以及 AWS KMS 的更多信息，请参阅[使用服务器端加密 \(SSE\) 和 AWS KMS 保护数据](#)。

```
{  
    "Version": "2012-10-17",  
    "Id": "example-ID",  
    "Statement": [  
        {  
            "Sid": "example-statement-ID",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "s3.amazonaws.com"  
            },  
            "Action": [  
                "kms:GenerateDataKey",  
                "kms:Decrypt"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

4. 记录队列 ARN

您创建的 SQS 队列是您的 AWS 账户中的另一项资源，它有唯一的 Amazon 资源名称 (ARN)。在下一步骤中，您需要用到此 ARN。ARN 格式如下：

```
arn:aws:sqs:aws-region:account-id:queue-name
```

## 步骤 3：将通知配置添加到存储桶

您可以使用 Amazon S3 控制台或以编程方式使用 AWS 开发工具包启用存储桶通知。选择任何一个选项以配置存储桶通知。此部分提供使用适用于 Java 和 .NET 的 AWS 开发工具包的代码示例。

### 步骤 3 (选项 a)：使用控制台在存储桶上启用通知

使用 Amazon S3 控制台，添加请求 Amazon S3 执行以下操作的通知配置：

- 将 s3:ObjectCreated:\* 类型的事件发布到您的 Amazon SQS 队列。
- 将 s3:ReducedRedundancyLostObject 类型的事件发布到您的 Amazon SNS 主题。

在您保存通知配置后，Amazon S3 将发布测试消息，您将通过电子邮件收到该消息。

有关说明，请参阅[如何为 S3 存储桶启用和配置事件通知？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

### 步骤 3 (选项 b)：使用适用于 .NET 的 AWS 开发工具包在存储桶上启用通知

下面的 C# 代码示例提供一个完整的代码列表，用于为存储桶添加通知配置。您需要更新该代码，提供您的存储桶名称和 SNS 主题 ARN。有关如何创建和测试有效示例的信息，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

#### Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

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

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableNotificationAsync().Wait();
        }

        static async Task EnableNotificationAsync()
        {
            try
            {
                PutBucketNotificationRequest request = new PutBucketNotificationRequest
                {
```

```
        BucketName = bucketName
    };

    TopicConfiguration c = new TopicConfiguration
    {
        Events = new List<EventType> { EventType.ObjectCreatedCopy },
        Topic = snsTopic
    };
    request.TopicConfigurations = new List<TopicConfiguration>();
    request.TopicConfigurations.Add(c);
    request.QueueConfigurations = new List<QueueConfiguration>();
    request.QueueConfigurations.Add(new QueueConfiguration()
    {
        Events = new List<EventType> { EventType.ObjectCreatedPut },
        Queue = sqsQueue
    });

    PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered on server. Message:'{0}' ",
e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown error encountered on server. Message:'{0}' ",
e.Message);
}
}
}
```

## 步骤 3 (选项 c)：使用AWS SDK for Java在存储桶上启用通知

以下示例说明如何将通知配置添加到存储桶。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

### Example

```
import java.io.IOException;
import java.util.EnumSet;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketNotificationConfiguration;
import com.amazonaws.services.s3.model.TopicConfiguration;
import com.amazonaws.services.s3.model.QueueConfiguration;
import com.amazonaws.services.s3.model.S3Event;
import com.amazonaws.services.s3.model.SetBucketNotificationConfigurationRequest;

public class EnableNotificationOnABucket {

    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        String clientRegion = "*** Client region ***";
        String snsTopicARN = "*** SNS Topic ARN ***";
        String sqsQueueARN = "*** SQS Queue ARN ***";
```

```
try {
    AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(clientRegion)
        .build();
    BucketNotificationConfiguration notificationConfiguration = new
BucketNotificationConfiguration();

    // Add an SNS topic notification.
    notificationConfiguration.addConfiguration("snsTopicConfig",
        new TopicConfiguration(snsTopicARN,
    EnumSet.of(S3Event.ObjectCreated)));

    // Add an SQS queue notification.
    notificationConfiguration.addConfiguration("sqSQueueConfig",
        new QueueConfiguration(sqSQueueARN,
    EnumSet.of(S3Event.ObjectCreated)));

    // Create the notification configuration request and set the bucket
notification configuration.
    SetBucketNotificationConfigurationRequest request = new
SetBucketNotificationConfigurationRequest(
        bucketName, notificationConfiguration);
    s3Client.setBucketNotificationConfiguration(request);
}
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();
}
}
```

## 步骤 4：测试设置

现在，您可以通过将对象上传到存储桶来测试设置，并在 Amazon SQS 控制台中验证事件通知。有关说明，请参阅《Amazon Simple Queue Service 开发人员指南》的“开始使用”部分中的[接收消息](#)。

## 示例演练 2：为存储桶配置通知 (消息目标：AWS Lambda)

有关使用 Amazon S3 通知和 AWS Lambda 的示例，请参阅 AWS Lambda Developer Guide 中的[使用 AWS Lambda 和 Amazon S3](#)。

## 事件消息结构

Amazon S3 发送以发布事件的通知消息是具有以下结构的 JSON 消息。请注意以下几点：

- 如果您希望通过联系 Amazon S3 支持来跟踪请求，则 `responseElements` 键值很有用。`x-amz-request-id` 和 `x-amz-id-2` 都可帮助 Amazon S3 跟踪单个请求。这些值与 Amazon S3 为响应您的原始 PUT 请求（该请求启动了事件）而返回的值相同。

- **s3** 键提供事件中涉及的存储桶和对象的相关信息。对象键名称值进行了 URL 编码。例如，“red flower.jpg”变成了“red+flower.jpg”( S3 返回“application/x-www-form-urlencoded”作为响应中的内容类型 )。
- **sequencer** 键提供了确定事件顺序的方法。无法保证事件通知按事件发生的顺序到达。但是，来自创建对象 (PUT) 和删除对象的事件的通知包含 sequencer，可用于确定给定对象键的事件的顺序。

如果将来自同一对象键上的两个事件通知的 sequencer 字符串进行比较，就会发现 sequencer 十六进制值较大的事件通知是后发生的事件。如果您正在使用事件通知来维护 Amazon S3 对象的单独数据库或索引，则您可能需要在处理每个事件通知时比较和存储 sequencer 值。

请注意：

- sequencer 不能用于确定不同对象键上的事件的顺序。
- 排序器可以有不同的长度。因此，为了比较这些值，您首先要用零填补较短的值的右侧，然后进行字母表顺序比较。

```
{  
    "Records": [  
        {  
            "eventVersion": "2.0",  
            "eventSource": "aws:s3",  
            "awsRegion": "us-east-1",  
            "eventTime": "The time, in ISO-8601 format, for example, 1970-01-01T00:00:00.000Z,  
            when S3 finished processing the request",  
            "eventName": "event-type",  
            "userIdentity": {  
                "principalId": "Amazon-customer-ID-of-the-user-who-caused-the-event"  
            },  
            "requestParameters": {  
                "sourceIPAddress": "ip-address-where-request-came-from"  
            },  
            "responseElements": {  
                "x-amz-request-id": "Amazon S3 generated request ID",  
                "x-amz-id-2": "Amazon S3 host that processed the request"  
            },  
            "s3": {  
                "s3SchemaVersion": "1.0",  
                "configurationId": "ID found in the bucket notification configuration",  
                "bucket": {  
                    "name": "bucket-name",  
                    "ownerIdentity": {  
                        "principalId": "Amazon-customer-ID-of-the-bucket-owner"  
                    },  
                    "arn": "bucket-ARN"  
                },  
                "object": {  
                    "key": "object-key",  
                    "size": "object-size",  
                    "eTag": "object eTag",  
                    "versionId": "object version if bucket is versioning-enabled, otherwise  
null",  
                    "sequencer": "a string representation of a hexadeciml value used to  
determine event sequence,  
                        only used with PUTs and DELETES"  
                }  
            },  
            {  
                // Additional events  
            }  
        ]  
    }  
}
```

以下是示例消息：

- 测试消息 - 当您在存储桶上配置事件通知时，Amazon S3 会发送以下测试消息：

```
{  
    "Service": "Amazon S3",  
    "Event": "s3:TestEvent",  
    "Time": "2014-10-13T15:57:02.089Z",  
    "Bucket": "bucketname",  
    "RequestId": "5582815E1AEA5ADF",  
    "HostId": "8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"  
}
```

- 使用 PUT 请求创建对象时的示例消息 - 以下消息是 Amazon S3 发送以发布 s3:ObjectCreated:Put 事件的消息示例：

```
{  
    "Records": [  
        {  
            "eventVersion": "2.0",  
            "eventSource": "aws:s3",  
            "awsRegion": "us-east-1",  
            "eventTime": "1970-01-01T00:00:00.000Z",  
            "eventName": "ObjectCreated:Put",  
            "userIdentity": {  
                "principalId": "AIDAJDPLRKLG7UEEXAMPLE"  
            },  
            "requestParameters": {  
                "sourceIPAddress": "127.0.0.1"  
            },  
            "responseElements": {  
                "x-amz-request-id": "C3D13FE58DE4C810",  
                "x-amz-id-2": "FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/  
JRWeUWerMUE5JgHvANOjpD"  
            },  
            "s3": {  
                "s3SchemaVersion": "1.0",  
                "configurationId": "testConfigRule",  
                "bucket": {  
                    "name": "mybucket",  
                    "ownerIdentity": {  
                        "principalId": "A3NL1KOZZKEExample"  
                    },  
                    "arn": "arn:aws:s3:::mybucket"  
                },  
                "object": {  
                    "key": "HappyFace.jpg",  
                    "size": 1024,  
                    "eTag": "d41d8cd98f00b204e9800998ecf8427e",  
                    "versionId": "096fKKXTRTtl3on89fvO.nfljtsv6qko",  
                    "sequencer": "0055AED6DCD90281E5"  
                }  
            }  
        }  
    ]  
}
```

# 跨区域复制 (CRR)

跨区域复制是一项存储桶级别配置，该功能支持跨不同 AWS 区域中的存储桶自动异步复制对象。我们将这些存储桶分别称为源 存储桶和目标 存储桶。这些存储桶可由不同的 AWS 账户拥有。

要激活此功能，您可以将一个复制配置添加到源存储桶，来指示 Amazon S3 根据此配置复制对象。在复制配置中，您需要提供如下所示的信息：

- 您希望 Amazon S3 将对象复制到的目标存储桶。
- 您要复制的对象。您可以请求 Amazon S3 复制所有对象，也可以通过在配置中提供一个键名称前缀来请求复制部分对象。例如，您可以将跨区域复制配置为仅复制具有键名称前缀 Tax/ 的对象。这会使 Amazon S3 复制具有 Tax/doc1 或 Tax/doc2 等键的对象，但不复制具有 Legal/doc3 键的对象。
- 默认情况下，Amazon S3 使用源对象的存储类来创建对象副本。您可以选择指定要用于目标存储桶中的对象副本的存储类。

有可供您指定的其他可选配置。有关更多信息，请参阅 [其他跨区域复制配置 \(p. 451\)](#)。

除非您在复制配置中发出特定请求，否则目标存储桶中的对象副本与源存储桶中的对象完全相同。例如：

- 副本具有相同的键名称和元数据 (例如，创建时间、用户定义的元数据和版本 ID)。
- 除非您在复制配置中明确指定其他存储类，否则 Amazon S3 将使用与源对象相同的存储类来存储对象副本。
- 假定对象副本仍由源对象拥有者拥有，当 Amazon S3 最初复制对象时，它还将复制相应的对象访问控制列表 (ACL)。

Amazon S3 使用安全套接字层 (SSL) 跨 AWS 区域加密传输中的所有数据。

源存储桶中的对象只能复制到一个目标存储桶。在 Amazon S3 复制一个对象后，无法再次复制该对象。例如，您可以在现有复制配置中更改目标存储桶，但 Amazon S3 不会再次复制它。

## 使用案例方案

您可能基于各种原因为存储桶配置跨区域复制，其中包括：

- 合规性要求 – 虽然 Amazon S3 默认跨多个地理位置较远的可用区存储数据，但是合规性要求所规定的数据存储距离可能更远。通过跨区域复制，可以在远距离 AWS 区域之间复制数据以满足这些合规性要求。
- 最大限度减少延迟 – 客户处于两个地理位置。为了最大限度缩短访问对象时的延迟，可以在地理位置与用户接近的 AWS 区域中维护对象副本。
- 操作原因 – 您在两个不同 AWS 区域中具有分析同一组对象的计算集群。您可能选择在这些区域中维护对象副本。

- 在不同的所有权下维护对象副本 – 无论谁拥有此源存储桶或源对象，您都可以指示 Amazon S3 将副本的所有权更改为拥有目标存储桶的 AWS 账户。您可以选择执行此操作来限制对对象副本的访问权。这也称作复制配置的拥有者覆盖 选项。

## 要求

跨区域复制的要求：

- 源存储桶和目标存储桶必须已启用版本控制。有关版本控制的更多信息，请参阅[使用版本控制 \(p. 380\)](#)。
- 源存储桶和目标存储桶必须处于不同的 AWS 区域。有关可以在其中创建存储桶的 AWS 区域的列表，请参阅 AWS General Reference 中的[区域和终端节点](#)。
- Amazon S3 必须有权代表您将对象从源存储桶复制到目标存储桶。

您可以通过创建 IAM 角色授予这些权限。有关 IAM 角色的更多信息，请参阅[创建 IAM 角色 \(p. 446\)](#)。

### Important

要传递您创建的用于授予 Amazon S3 复制权限的 IAM 角色，您必须具有 `iam:PassRole` 权限。有关更多信息，请参阅 IAM 用户指南 中的[授予向 AWS 服务传递角色的用户权限](#)。

- 如果源存储桶拥有者也是对象的拥有者，则其拥有复制对象的完全权限。如果不是，则对象拥有者必须通过对象 ACL 向存储桶拥有者授予 READ 和 READ\_ACP 权限。有关 Amazon S3 操作的更多信息，请参阅在[策略中指定权限 \(p. 282\)](#)。有关资源和所有权的更多信息，请参阅[Amazon S3 资源 \(p. 243\)](#)。

如果您要在跨账户方案 (其中，源存储桶和目标存储桶由不同的 AWS 账户拥有) 中设置复制配置，则以下附加要求将适用：

- IAM 角色 必须有权复制目标存储桶中的对象。目标存储桶拥有者可以通过存储桶策略授予这些权限。有关示例，请参阅[演练 2：配置跨区域复制（其中源存储桶和目标存储桶由不同 AWS 账户拥有）\(p. 458\)](#)。
- 在复制配置中，您可以选择指示 Amazon S3 将对象副本的所有权更改为拥有目标存储桶的 AWS 账户。有关相关的附加要求，请参阅[跨区域复制的其他配置：更改副本拥有者 \(p. 451\)](#)。

## 相关主题

[复制和不复制的内容 \(p. 444\)](#)

[设置跨区域复制 \(p. 445\)](#)

[查看跨区域复制状态 \(p. 472\)](#)

[跨区域复制：其他注意事项 \(p. 474\)](#)

[演练 1：配置跨区域复制（其中源存储桶和目标存储桶由同一 AWS 账户拥有）\(p. 457\)](#)

[演练 2：配置跨区域复制（其中源存储桶和目标存储桶由不同 AWS 账户拥有）\(p. 458\)](#)

## 复制和不复制的内容

本部分说明在对存储桶添加复制配置之后，Amazon S3 会复制以及不复制的内容。

### 复制的内容

Amazon S3 会复制以下内容：

- 添加复制配置之后创建的任何新对象（除了下一部分中介绍的对象）。
- 除了未加密对象之外，Amazon S3 会复制使用 Amazon S3 托管密钥 (SSE-S3) 或 AWS KMS 托管密钥 (SSE-KMS) 加密的对象。对象的复制副本也将使用服务器端加密进行加密，该服务器端加密的类型与用于源对象的类型相同，即 SSE-S3 或 SSE-KMS。有关服务器端加密的更多信息，请参阅[使用服务器端加密保护数据 \(p. 345\)](#)。
- 除了对象之外，Amazon S3 还会复制对象元数据。
- Amazon S3 仅复制存储桶拥有者有权读取对象和访问控制列表 (ACL) 的源存储桶中的对象。有关资源所有权的更多信息，请参阅[关于资源拥有者 \(p. 243\)](#)。
- 除非您已指示 Amazon S3 更改跨账户方案中的副本所有权（请参阅[跨区域复制的其他配置：更改副本拥有者 \(p. 451\)](#)），否则将复制任何对象 ACL 更新。

在 Amazon S3 能够使两个 ACL 同步之前，可能存在一些延迟。这仅适用于在向存储桶添加复制配置之后创建的对象。

- Amazon S3 会复制对象标签（如果有）。

### 删除操作和跨区域复制

如果从源存储桶中删除对象，则跨区域复制行为如下所示：

- 如果进行 DELETE 请求而不指定对象版本 ID，则 Amazon S3 会添加删除标记，而跨区域复制会将该标记复制到目标存储桶。有关版本控制和删除标记的更多信息，请参阅[使用版本控制 \(p. 380\)](#)。
- 如果 DELETE 请求指定了要删除的特定对象版本 ID，则 Amazon S3 会在源存储桶中删除该对象版本，但不会在目标存储桶中复制删除操作（换句话说，它不会从目标存储桶中删除同一对象版本）。此行为可防止恶意删除数据。

### 不复制的内容

Amazon S3 不复制以下内容：

- Amazon S3 不会以回溯方式复制在添加复制配置之前存在的对象。

- 不会复制以下加密的对象：
  - 使用客户提供的加密密钥通过服务器端加密 (SSE-C) 创建的对象。
  - 使用 AWS KMS 托管加密密钥通过服务器端加密 (SSE-KMS) 创建的对象，除非您明确启用此选项。

有关服务器端加密的更多信息，请参阅[使用服务器端加密保护数据 \(p. 345\)](#)。

- 存储桶拥有者没有权限的源存储桶中的对象。当对象拥有者与存储桶拥有者不同时，会出现此情况。有关对象拥有者如何向存储桶拥有者授予权限的信息，请参阅[在授予上传对象的交叉账户许可的同时，确保存储桶拥有者拥有完全控制 \(p. 309\)](#)。
- 对存储桶级别子资源进行的更新不会进行复制。例如，您可以更改源存储桶上的生命周期配置或向源存储桶添加通知配置。这些更改不会应用于目标存储桶。这使您可以在源存储桶和目标存储桶中具有不同的存储桶配置。
- 仅复制客户操作。不复制生命周期配置执行的操作。有关生命周期配置的更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

例如，如果仅对源存储桶启用了生命周期配置，则 Amazon S3 会为过期对象创建删除标记，但不会复制这些标记。但是，如果希望对源存储桶和目标存储桶进行相同的生命周期配置，则可以对两个存储桶使用相同的生命周期配置。

- 源存储桶中作为另一个跨区域复制所建副本的对象不会进行复制。

假设您配置的跨区域复制中，存储桶 A 是源，而存储桶 B 是目标。现在假设您添加另一个跨区域复制，其中存储桶 B 是源，而存储桶 C 是目标。在这种情况下，存储桶 B 中作为存储桶 A 中对象的副本的对象不会复制到存储桶 C。

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

[设置跨区域复制 \(p. 445\)](#)

[查看跨区域复制状态 \(p. 472\)](#)

## 设置跨区域复制

要设置跨区域复制，需要两个存储桶—源和目标。这两个存储桶必须启用版本控制，并且处于不同的 AWS 区域。有关可以在其中创建存储桶的 AWS 区域的列表，请参阅 AWS General Reference 中的[区域和终端节点](#)。源存储桶中的对象只能复制到一个目标存储桶。

### Important

如果您在不受版本控制的存储桶中具有对象到期生命周期策略，并且希望在启用版本控制时保持相同的永久删除行为，则必须添加非当前版本到期策略。非当前版本到期生命周期策略将管理在受版本控制的存储桶中删除非当前对象版本的行为。(启用版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。)有关更多信息，请参阅[如何为 S3 存储桶创建生命周期策略？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

## 主题

- [为由同一个 AWS 账户拥有的存储桶设置跨区域复制 \(p. 446\)](#)
- [为由其他 AWS 账户拥有的存储桶设置跨区域复制 \(p. 450\)](#)
- [相关主题 \(p. 451\)](#)

# 为由同一个 AWS 账户拥有的存储桶设置跨区域复制

如果这两个存储桶由同一个 AWS 账户拥有，则可以通过执行以下操作，设置从源存储桶到目标存储桶的跨区域复制：

- 在账户中创建 IAM 角色。此角色向 Amazon S3 角色授予代表您复制对象的权限。
- 对源存储桶添加复制配置。

## 创建 IAM 角色

Amazon S3 将源存储桶中的对象复制到目标存储桶。您必须通过 IAM 角色向 Amazon S3 授予必要权限。

### Note

默认情况下，所有 Amazon S3 资源（存储桶、对象和相关子资源）都是私有的：只有资源拥有者可以访问资源。因此，Amazon S3 需要从源存储桶读取对象以及将它们复制到目标存储桶的权限。

创建 IAM 角色时，需将以下策略附加到角色：

- 一个信任策略，您在其中将 Amazon S3 标识为可代入角色的服务委托人，如下所示：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "s3.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

有关 IAM 角色的更多信息，请参阅 IAM 用户指南 中的 [IAM 角色](#)。

- 一个访问策略，您在其中向角色授予代表您执行复制任务的权限。在 Amazon S3 代入角色时，它将拥有您在此策略中指定的权限。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:GetReplicationConfiguration",  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::source-bucket"  
            ]  
        }  
    ]  
}
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::source-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::destination-bucket/*"
        }
    ]
}
```

访问策略授予以下操作的权限：

- `s3:GetReplicationConfiguration` 和 `s3>ListBucket` – 源 存储桶上的权限允许 Amazon S3 检索复制配置和列出存储桶（当前权限模型需要 `s3>ListBucket` 权限来访问删除标记）。
- `s3GetObjectVersion` 和 `s3GetObjectVersionAcl` – 针对所有对象授予的这些操作的权限允许 Amazon S3 获取对象的特定对象版本和访问控制列表（ACL）。
- `s3:ReplicateObject` 和 `s3:ReplicateDelete` – 目标 存储桶中对象上的这些操作的权限允许 Amazon S3 在目标存储桶中复制对象或删除标记。有关删除标记的信息，请参阅[删除操作和跨区域复制 \(p. 444\)](#)。

**Note**

目标 存储桶上的 `s3:ReplicateObject` 操作的权限还允许复制对象标签。因此，Amazon S3 还复制对象标签（您无需明确授予 `s3:ReplicateTags` 操作的权限）。

- `s3GetObjectVersionTagging` – 源 存储桶中对象上的此操作的权限允许 Amazon S3 读取复制的对象标签（请参阅[对象标签 \(p. 96\)](#)）。如果 Amazon S3 未获得此权限，则它将复制对象，但不复制对象标签（如果有）。

有关 Amazon S3 操作的列表，请参阅[在策略中指定权限 \(p. 282\)](#)。

**Important**

您只能授予您拥有其权限的资源的权限。更具体地说，拥有 IAM 角色的 AWS 账户必须拥有其向 IAM 角色授予的操作的权限。

例如，假定源存储桶包含由另一个 AWS 账户拥有的对象。对象拥有者必须通过对象 ACL 向拥有 IAM 角色的 AWS 账户明确授予必要的权限。否则，这些对象的跨区域复制将失败（因为根据角色策略中授予的权限，Amazon S3 无法访问这些对象）。有关 ACL 权限的信息，请参阅[访问控制列表 \(ACL\) 概述 \(p. 333\)](#)。

在您了解有关其他 CRR 配置的更多信息后，您可以向 Amazon S3 授予其他资源的权限。一般规则仍将适用，即拥有 IAM 角色的 AWS 账户必须拥有其向 IAM 角色授予的操作的权限。

## 添加复制配置

向存储桶添加复制配置时，Amazon S3 会将该配置存储为 XML。以下是示例配置。有关 XML 结构的更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [PUT Bucket 复制](#)。

### Important

将复制配置添加到存储桶时，您必须具有 `iam:PassRole` 权限才能传递授予 Amazon S3 复制权限的 IAM 角色。IAM 角色由在复制配置 XML 中的 `<Role>` 元素中使用的 Amazon 资源名称 (ARN) 指定。有关更多信息，请参阅 IAM 用户指南中的[授予向 AWS 服务传递角色的用户权限](#)。

Example 1：包含一个规则的复制配置

考虑以下复制配置：

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::AcctID:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix></Prefix>

    <Destination><Bucket>arn:aws:s3:::destinationbucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

除了要让 Amazon S3 代入的 IAM 角色之外，配置还指定一个规则，如下所示：

- 规则状态，指示规则是有效的。
- 空前缀，指示规则适用于存储桶中的所有对象。
- 目标存储桶，在其中复制对象。

您可以选择为对象副本指定存储类，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix></Prefix>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
      <StorageClass>storage-class</StorageClass>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

如果 `<Rule>` 不指定存储类别，则 Amazon S3 将使用源对象的存储类别创建对象副本。

您可以指定 Amazon S3 支持的任何存储类，但 `GLACIER` 存储类除外。如果您要将对象转换为 `GLACIER` 存储类，请使用生命周期配置。有关生命周期管理的更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。有关存储类别的更多信息，请参阅[存储类别 \(p. 90\)](#)。

Example 2：包含两个规则的复制配置

考虑以下复制配置：

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Prefix>Tax</Prefix>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
    </Destination>
    ...
  </Rule>
  <Rule>
    <Prefix>Project</Prefix>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
    </Destination>
    ...
  </Rule>

</ReplicationConfiguration>
```

在复制配置中：

- 每个规则均指定一个不同的键名称前缀，用于标识要将规则应用于的源存储桶中的一组单独的对象。随后，Amazon S3 仅复制带特定键前缀的对象。例如，Amazon S3 会复制具有键名称 Tax/doc1.pdf 和 Project/project1.txt 的对象，但不会复制任何具有键名称 PersonalDoc/documentA 的对象。
- 两个规则指定了同一个目标存储桶。
- 两个规则均已启用。

您不能指定重叠前缀，如下所示：

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::AcctID:role/role-name</Role>

  <Rule>
    <Prefix>TaxDocs</Prefix>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
    </Destination>
  </Rule>
  <Rule>
    <Prefix>TaxDocs/2015</Prefix>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

两个规则指定重叠前缀 Tax/ 和 Tax/2015，这是不允许的。

### Example 3：示例演练

当源存储桶和目标存储桶由同一个 AWS 账户拥有时，您可以使用 Amazon S3 控制台设置跨区域复制。假设您拥有源存储桶和目标存储桶（都启用了版本控制），则可以使用控制台对源存储桶添加复制配置。有关更多信息，请参阅以下主题：

- 演练 1：配置跨区域复制（其中源存储桶和目标存储桶由同一 AWS 账户拥有）(p. 457)
- Amazon Simple Storage Service 控制台用户指南 中的[启用跨区域复制](#)。

## 为由其他 AWS 账户拥有的存储桶设置跨区域复制

在跨账户方案中设置复制配置时，除了执行[上一个部分](#)中概述的相同配置之外，目标存储桶拥有者还必须添加一个存储桶策略，以便向源存储桶拥有者授予执行复制操作的权限。

```
{  
    "Version": "2008-10-17",  
    "Id": "PolicyForDestinationBucket",  
    "Statement": [  
        {  
            "Sid": "1",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "SourceBucket-AcctID"  
            },  
            "Action": [  
                "s3:ReplicateDelete",  
                "s3:ReplicateObject"  
            ],  
            "Resource": "arn:aws:s3:::destinationbucket/*"  
        },  
        {  
            "Sid": "2",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "SourceBucket-AcctID"  
            },  
            "Action": "s3>List*",  
            "Resource": "arn:aws:s3:::destinationbucket"  
        }  
    ]  
}
```

有关示例，请参阅[演练 2：配置跨区域复制（其中源存储桶和目标存储桶由不同 AWS 账户拥有）\(p. 458\)](#)。

如果向源存储桶中的对象添加标签，请注意以下事项：

- 如果源存储桶拥有者向 Amazon S3 授予 s3:GetObjectVersionTagging 和 s3:ReplicateTags 操作的权限来复制对象标签（通过 IAM 角色），则 Amazon S3 将复制标签以及对象。有关 IAM 角色的信息，请参阅[创建 IAM 角色 \(p. 446\)](#)。
- 如果目标存储桶拥有者不希望复制标签，则该所有者可以向目标存储桶策略添加以下语句来显式拒绝 s3:ReplicateTags 操作的权限。

```
...  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Principal": {  
                "AWS": "arn:aws:iam::SourceBucket-AcctID:root"  
            },  
            "Action": ["s3:ReplicateTags"],  
            "Resource": "arn:aws:s3:::destinationbucket/*"  
        }  
    ]  
...
```

## 更改副本所有权

您可以选择指示 Amazon S3 将副本所有权更改为拥有目标存储桶的 AWS 账户。这也称作复制配置的拥有者覆盖 选项。有关更多信息，请参阅[跨区域复制的其他配置：更改副本拥有者 \(p. 451\)](#)。

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

[复制和不复制的内容 \(p. 444\)](#)

[演练 1：配置跨区域复制（其中源存储桶和目标存储桶由同一 AWS 账户拥有）\(p. 457\)](#)

[演练 2：配置跨区域复制（其中源存储桶和目标存储桶由不同 AWS 账户拥有）\(p. 458\)](#)

[查看跨区域复制状态 \(p. 472\)](#)

[在 Amazon S3 中排查跨区域复制问题 \(p. 473\)](#)

## 其他跨区域复制配置

### 主题

- [跨区域复制的其他配置：更改副本拥有者 \(p. 451\)](#)
- [CRR 其他配置：复制使用 AWS KMS 托管加密密钥通过服务器端加密 \(SSE\) 创建的对象 \(p. 453\)](#)

此部分描述了与跨区域复制相关的可选配置。有关核心复制的信息，请参阅[设置跨区域复制 \(p. 445\)](#)。

## 跨区域复制的其他配置：更改副本拥有者

无论谁拥有此源存储桶或源对象，您都可以指示 Amazon S3 将副本的所有权更改为拥有目标存储桶的 AWS 账户。您可以选择执行此操作来限制对对象副本的访问权。这也称作复制配置的拥有者覆盖 选项。

### Warning

仅在源和目标存储桶由不同 AWS 账户拥有时添加拥有者覆盖选项。

有关在跨账户方案中设置复制配置的信息，请参阅[为由其他 AWS 账户拥有的存储桶设置跨区域复制 \(p. 450\)](#)。此部分仅提供附加信息，用于指示 Amazon S3 将副本所有权更改为拥有目标存储桶的 AWS 账户。

- 将 `<Account>` 和 `<AccessControlTranslation>` 元素添加为 `<Destination>` 元素的子元素，如下示例所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix></Prefix>
    <Destination>
      <Bucket>arn:aws:s3:::destination-bucket</Bucket>
      <Account>destination-bucket-owner-account-id</Account>
      <AccessControlTranslation>
        <Owner>Destination</Owner>
      </AccessControlTranslation>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

```
</Rule>  
</ReplicationConfiguration>
```

- 向 IAM 角色添加更多权限以允许 Amazon S3 更改副本所有权。

允许目标存储桶中所有副本上的 `s3:ObjectOwnerOverrideToBucketOwner` 操作的 IAM 角色权限，如以下策略声明中所述。

```
...  
{  
    "Effect": "Allow",  
    "Action": [  
        "s3:ObjectOwnerOverrideToBucketOwner"  
    ],  
    "Resource": "arn:aws:s3:::destination-bucket/*"  
}  
...
```

- 在目标存储桶的存储桶策略中，添加 `s3:ObjectOwnerOverrideToBucketOwner` 操作的权限，以允许拥有源存储桶权限的 AWS 账户的副本所有权更改（在生效时，接受对象副本的所有权）。您可以将以下策略语句添加到您的存储桶策略。

```
...  
{  
    "Sid": "1",  
    "Effect": "Allow",  
    "Principal": {"AWS": "source-bucket-account-id"},  
    "Action": ["s3:ObjectOwnerOverrideToBucketOwner"],  
    "Resource": "arn:aws:s3:::destination-bucket/*"  
}  
...
```

### Warning

仅在两个存储桶由不同 AWS 账户拥有时，将此拥有者覆盖选项添加到复制配置。Amazon S3 不会检查存储桶由相同账户还是不同账户拥有。如果您在两个存储桶均由同一个 AWS 账户拥有时添加此选项，仍会应用拥有者覆盖。也就是说，Amazon S3 向目标存储桶拥有者授予完全权限，不将后续的更新复制到源对象访问控制列表（ACL）。副本拥有者可以使用 `PUT ACL` 请求直接对与副本关联的 ACL 进行更改，但不能通过复制进行更改。

有关示例，请参阅[演练 3：将副本拥有者更改为目标存储桶拥有者 \(p. 463\)](#)。

在跨账户方案中，源和目标存储桶由不同的 AWS 账户拥有，以下情况适用：

- 使用可选的拥有者覆盖选项创建复制配置，默认情况下，源对象拥有者也拥有副本。相应地，与对象版本一起，Amazon S3 还会复制与对象版本关联的 ACL。

您可以添加可选的拥有者覆盖配置，指示 Amazon S3 将副本拥有者更改为拥有目标存储桶的 AWS 账户。在这种情况下，由于拥有者不同，Amazon S3 仅复制对象版本而不复制 ACL（同样，Amazon S3 不复制对源对象 ACL 的任何后续更改）。Amazon S3 在副本上设置 ACL，将完全控制权限授予目标存储桶拥有者。

- 更新复制配置（启用/禁用拥有者覆盖选项）– 假设您已将复制配置添加到了存储桶。Amazon S3 将对象版本复制到目标存储桶。随之一起，Amazon S3 还会复制对象 ACL 并将其与对象副本关联。

- 现在，假设您更新复制配置并添加拥有者覆盖选项。Amazon S3 复制对象版本时，它会放弃与源对象关联的 ACL，而是改为在副本上设置 ACL，将完全控制权限授予目标存储桶拥有者。对源对象 ACL 的任何后续更改不会复制。

对于在设置拥有者覆盖选项之前复制的对象版本，不应用此更改。也就是说，设置拥有者覆盖之前所复制的源对象 ACL 上的任何更新，仍将继续复制（因为对象及其副本继续具有相同的拥有者）。

- 现在，假设您以后禁用了拥有者覆盖配置。Amazon S3 继续复制任何新对象版本并将对象 ACL 与目标关联。当您禁用拥有者覆盖时，它不应用到在复制配置中设置了拥有者覆盖时所复制的对象（Amazon S3 所做的对象所有权更改仍然保持有效）。也就是说，对于在您设置了拥有者覆盖的情况下所复制的对象版本上的 ACL，接下来不会进行复制。

## CRR 其他配置：复制使用 AWS KMS 托管加密密钥通过服务器端加密 (SSE) 创建的对象

您的源存储桶中可以有使用 AWS KMS 托管密钥通过服务器端加密创建的对象。默认情况下，Amazon S3 不会复制 AWS KMS 加密的对象。如果您希望 Amazon S3 复制这些对象以及[基本复制配置](#)，则必须执行以下操作：

- 提供您希望 Amazon S3 用来加密对象副本的目标存储桶区域的 AWS KMS 托管密钥。
- 向 IAM 角色授予其他权限，以便 Amazon S3 能够使用 AWS KMS 密钥访问对象。

### 主题

- [在复制配置中指定其他信息 \(p. 453\)](#)
- [IAM 角色的附加权限 \(p. 454\)](#)
- [跨账户方案：附加权限 \(p. 456\)](#)
- [相关注意事项 \(p. 457\)](#)

## 在复制配置中指定其他信息

在[基本复制配置](#)中，添加以下其他信息。

- 此功能（供 Amazon S3 用来复制使用 AWS KMS 托管密钥加密的对象）要求客户必须通过添加 `<SourceSelectionCriteria>` 元素明确选择加入。

```
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
```

- 通过添加 `<EncryptionConfiguration>` 元素提供您希望 Amazon S3 用来加密对象副本的 AWS KMS 密钥：

```
<EncryptionConfiguration>
  <ReplicaKmsKeyId>The AWS KMS key ID (S3 can use to encrypt object replicas).</ReplicaKmsKeyId>
</EncryptionConfiguration>
```

### Important

请注意，AWS KMS 密钥区域必须与目标存储桶的区域相同。确保 AWS KMS 密钥有效。PUT 存储桶复制 API 不会检查无效的 AWS KMS 密钥。您将获得 200 OK 响应，但如果 AWS KMS 密钥无效，则复制将失败。

以下是包含可选配置元素的跨区域复制配置的示例：

```
<ReplicationConfiguration>
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Prefix>prefix1</Prefix>
    <Status>Enabled</Status>
    <SourceSelectionCriteria>
      <SseKmsEncryptedObjects>
        <Status>Enabled</Status>
      </SseKmsEncryptedObjects>
    </SourceSelectionCriteria>
    <Destination>
      <Bucket>arn:aws:s3:::destination-bucket</Bucket>
      <EncryptionConfiguration>
        <ReplicaKmsKeyID>The AWS KMS key ID (that S3 can use to encrypt object replicas).</ReplicaKmsKeyID>
      </EncryptionConfiguration>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

此复制配置包含一个规则。该规则应用于具有指定键前缀的对象。Amazon S3 使用 AWS KMS 密钥 ID 来加密这些对象副本。

## IAM 角色的附加权限

Amazon S3 需要附加权限才能复制使用 AWS KMS 托管密钥通过服务器端加密创建的对象。您必须向 IAM 角色授予以下附加权限：

- 授予源对象的 s3:GetObjectVersionForReplication 操作的权限。此操作的权限允许 Amazon S3 复制未加密的对象以及使用 SSE-S3 (Amazon S3 托管加密密钥) 或 AWS KMS 托管加密 (SSE-KMS) 密钥通过服务器端加密创建的对象。

#### Note

s3:GetObjectVersion 操作的权限允许复制未加密的对象和 SSE-S3 加密的对象。但是，它不允许复制使用 AWS KMS 托管加密密钥创建的对象。

#### Note

建议您使用 s3:GetObjectVersionForReplication 操作而不是 s3:GetObjectVersion 操作，因为它仅向 Amazon S3 提供跨区域复制所需的最低权限。

- 授以下 AWS KMS 操作的权限：
  - 用于加密源对象的 AWS KMS 密钥的 kms:Decrypt 权限。
  - 用于加密对象副本的 AWS KMS 密钥的 kms:Encrypt 权限。

建议您使用 AWS KMS 条件密钥将这些权限限制为特定的存储桶和对象，如以下示例策略声明中所示：

```
{
  "Action": ["kms:Decrypt"],
  "Effect": "Allow",
  "Condition": {
```

```

    "StringLike": {
        "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
        "kms:EncryptionContext:aws:s3:arn": [
            "arn:aws:s3:::source-bucket-name/prefix1*",
        ]
    }
},
"Resource": [
    "List of AWS KMS key IDs that was used to encrypt source objects.",
]
},
{
    "Action": ["kms:Encrypt"],
    "Effect": "Allow",
    "Condition": {
        "StringLike": {
            "kms:ViaService": "s3.destination-bucket-region.amazonaws.com",
            "kms:EncryptionContext:aws:s3:arn": [
                "arn:aws:s3:::destination-bucket-name/prefix1*",
            ]
        }
    },
    "Resource": [
        "List of AWS KMS key IDs, that you want S3 to use to encrypt object replicas."
    ]
}
}

```

对于策略中列出的 AWS KMS 密钥，拥有 IAM 角色的 AWS 账户必须具有这些 AWS KMS 操作 (kms:Encrypt 和 kms:Decrypt) 的权限。如果 AWS KMS 密钥由另一个 AWS 账户拥有，则密钥拥有者必须向拥有 IAM 角色的 AWS 账户授予这些权限。有关管理对这些密钥的访问的更多信息，请参阅 AWS Key Management Service Developer Guide 中的[在 AWS KMS 中使用 IAM 策略](#)。

以下是一个完整 IAM 策略，该策略授予必要权限来复制未加密的对象以及使用 Amazon S3 托管加密密钥和 AWS KMS 托管加密密钥通过服务器端加密创建的对象。

#### Note

不会复制使用客户提供的加密密钥通过服务器端加密 (SSE-C) 创建的对象。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::source-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersionForReplication",
                "s3:GetObjectVersionAcl"
            ],
            "Resource": [
                "arn:aws:s3:::source-bucket/prefix1*"
            ]
        }
    ]
}
```

```
{
    "Effect": "Allow",
    "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete"
    ],
    "Resource": "arn:aws:s3:::destination-bucket/prefix1*"
},
{
    "Action": [
        "kms:Decrypt"
    ],
    "Effect": "Allow",
    "Condition": {
        "StringLike": {
            "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
            "kms:EncryptionContext:aws:s3:arn": [
                "arn:aws:s3:::source-bucket-name/prefix1*"
            ]
        }
    },
    "Resource": [
        "List of AWS KMS key IDs used to encrypt source objects."
    ]
},
{
    "Action": [
        "kms:Encrypt"
    ],
    "Effect": "Allow",
    "Condition": {
        "StringLike": {
            "kms:ViaService": "s3.destination-bucket-region.amazonaws.com",
            "kms:EncryptionContext:aws:s3:arn": [
                "arn:aws:s3:::destination-bucket-name/prefix1*"
            ]
        }
    },
    "Resource": [
        "List of AWS KMS key IDs that you want S3 to use to encrypt object replicas."
    ]
}
}
```

## 跨账户方案：附加权限

在跨账户方案中，目标 AWS KMS 密钥必须是客户主密钥 (CMK)。密钥拥有者必须使用下列方法之一向源存储桶拥有者授予密钥的使用权限：

- 使用 IAM 控制台。
  1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 [https://console.aws.amazon.com/iam/。](https://console.aws.amazon.com/iam/)
  2. 选择 Encryption keys。
  3. 选择 AWS KMS 密钥。
  4. 在 Key Policy、Key Users、External Accounts 中，选择 Add External Account。
  5. 在 arn:aws:iam:: 框中指定源存储桶账户 ID。
  6. 选择 Save Changes。
- 使用 AWS CLI。有关更多信息，请参阅“AWS CLI 命令参考”中的 [put-key-policy](#)。有关基础 API 的信息，请参阅 [AWS Key Management Service API Reference](#) 中的 [PutKeyPolicy](#)。

## 相关注意事项

在启用 CRR 后，在使用 AWS KMS 加密添加大量新对象时，您可能会遇到限制 (HTTP 503 速度变慢错误)。这与 AWS KMS 支持的每秒 KMS 事务数限制相关。有关更多信息，请参阅 AWS Key Management Service Developer Guide 中的 [限制](#)。

在此情况下，我们建议您通过在 AWS 支持中心创建一个案例来请求增加您的 AWS KMS API 速率限制。有关更多信息，请参阅 <https://console.aws.amazon.com/support/home#/>。

## 跨区域复制示例

此部分提供了以下用于设置跨区域复制的示例演练。

### 主题

- 演练 1：配置跨区域复制 (其中源存储桶和目标存储桶由同一 AWS 账户拥有) (p. 457)
- 演练 2：配置跨区域复制 (其中源存储桶和目标存储桶由不同 AWS 账户拥有) (p. 458)
- 跨区域复制：其他演练 (p. 462)
- 使用控制台设置跨区域复制 (p. 468)
- 使用 AWS SDK for Java 设置跨区域复制 (p. 468)
- 使用适用于 .NET 的 AWS 开发工具包设置跨区域复制 (p. 470)

## 演练 1：配置跨区域复制 (其中源存储桶和目标存储桶由同一 AWS 账户拥有)

在此部分中，您将在不同的 AWS 区域中创建两个存储桶 (源和目标)，对这两个存储桶启用版本控制，然后对源存储桶配置跨区域复制。

### 1. 创建两个存储桶。

- a. 在一个 AWS 区域中创建源 存储桶。例如，美国西部 (俄勒冈) (us-west-2)。有关说明，请参阅[如何创建 S3 存储桶？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。
  - b. 在另一个 AWS 区域中创建目标 存储桶。例如，美国东部 (弗吉尼亚北部) (us-east-1)。
2. 对这两个存储桶启用版本控制。有关说明，请参阅[如何为 S3 存储桶启用或暂停版本控制？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

#### Important

如果您在不受版本控制的存储桶中具有对象到期生命周期策略，并且希望在启用版本控制时保持相同的永久删除行为，则必须添加非当前版本到期策略。非当前版本到期生命周期策略将管理在受版本控制的存储桶中删除非当前对象版本的行为。(启用版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。)有关更多信息，请参阅[如何为 S3 存储桶创建生命周期策略？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。

3. 对源 存储桶启用跨区域复制。您决定是要复制所有对象还是仅复制具有特定前缀的对象 (如果使用控制台，则等同于考虑是否仅复制特定文件夹中的对象)。有关说明，请参阅[如何为 S3 存储桶启用和配置跨区域复制？](#) (在 Amazon Simple Storage Service 控制台用户指南 中)。
4. 按如下所示测试设置：
  - a. 在源存储桶中创建对象，并验证 Amazon S3 是否在目标存储桶中复制了对象。Amazon S3 复制对象所需的时间量取决于对象大小。有关查找复制状态的信息，请参阅[查看跨区域复制状态 \(p. 472\)](#)。
  - b. 在源存储桶中更新对象的访问控制列表 (ACL)，并验证更改是否出现在目标存储桶中。有关说明，请参阅 Amazon Simple Storage Service 控制台用户指南 中的[设置存储桶和对象访问权限](#)。

- c. 更新对象的元数据，并验证更改是否出现在目标存储桶中。有关说明，请参阅[如何向 S3 对象添加元数据？](#)(在 Amazon Simple Storage Service 控制台用户指南 中)。

请记住，副本是源存储桶中对象的精确副本。

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

[演练 2：配置跨区域复制 \(其中源存储桶和目标存储桶由不同 AWS 账户拥有\) \(p. 458\)](#)

[复制和不复制的内容 \(p. 444\)](#)

[查看跨区域复制状态 \(p. 472\)](#)

## 演练 2：配置跨区域复制 (其中源存储桶和目标存储桶由不同 AWS 账户拥有)

在此演练中，您设置跨区域复制，其中源存储桶和目标存储桶由不同的 AWS 账户拥有。

由于存储桶由不同的 AWS 账户拥有，因此，您必须执行一个额外步骤来设置跨区域复制 – 目标存储桶拥有者必须创建一个存储桶策略，以便向源存储桶拥有者授予执行复制操作的权限。

在本练习中，您将使用控制台执行所有步骤，但创建 IAM 角色和出于以下原因向源存储桶添加复制配置除外：

- Amazon S3 控制台支持在两个存储桶由同一个 AWS 账户拥有时设置复制配置。但是，在跨账户方案中，您必须指定由另一个 AWS 账户拥有的目标存储桶，并且 Amazon S3 控制台 UI 仅显示您的账户中的存储桶。
- 在 IAM 控制台中，Amazon S3 未在 AWS Service Roles 列表中。您可以选择创建一个 IAM 角色，但选择另一个服务角色类型(如 AWS Lambda)。在创建此角色后，您可以修改信任策略来指定可代入此角色的 Amazon S3 服务委托人(而不是 Lambda 服务委托人)。在本练习中，您使用 AWS CLI 创建此角色。

1. 使用两个不同的 AWS 账户创建两个存储桶。根据跨区域复制要求，您在不同的 AWS 区域中创建这两个存储桶并对它们启用版本控制。
  - a. 在一个 AWS 区域中创建源存储桶。例如，账户 A 中的 美国西部 (俄勒冈) (us-west-2)。有关说明，请转至[如何创建 S3 存储桶？](#)(在 Amazon Simple Storage Service 控制台用户指南 中)。
  - b. 在另一个 AWS 区域中创建目标存储桶。例如，账户 B 中的 美国东部 (弗吉尼亚北部) (us-east-1)。
  - c. 对这两个存储桶启用版本控制。有关说明，请参阅[如何为 S3 存储桶启用或暂停版本控制？](#)(在 Amazon Simple Storage Service 控制台用户指南 中)。

### Important

如果您在不受版本控制的存储桶中具有对象到期生命周期策略，并且希望在启用版本控制时保持相同的永久删除行为，则必须添加非当前版本到期策略。非当前版本到期生命周期策略将管理在受版本控制的存储桶中删除非当前对象版本的行为。(启用版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。)有关更多信息，请参阅[如何为 S3 存储桶创建生命周期策略？](#)(在 Amazon Simple Storage Service 控制台用户指南 中)。

2. 在目标存储桶上添加存储桶策略以允许源存储桶拥有者复制对象。

{

```
"Version": "2008-10-17",
"Id": "",
"Statement": [
  {
    "Sid": "Stmt123",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::AWS-ID-Account-A:root"
    },
    "Action": ["s3:ReplicateObject", "s3:ReplicateDelete"],
    "Resource": "arn:aws:s3:::destination-bucket/*"
  }
]
```

有关说明，请参阅[如何添加 S3 存储桶策略？](#)（在 Amazon Simple Storage Service 控制台用户指南中）。

3. 向 Amazon S3 授予代表源存储桶拥有者复制对象的权限。

在对源存储桶配置跨区域复制后，Amazon S3 将代表您复制对象。源存储桶拥有者可使用 IAM 角色向 Amazon S3 授予必要权限。在此步骤中，您将在账户 A 中创建一个 IAM 角色。

使用 AWS CLI 创建该 IAM 角色。有关如何设置 AWS CLI 的信息，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。本练习假定您已使用以下两个配置文件配置 AWS CLI：accountA 和 accountB。

- a. 复制以下策略并将其保存到名为 s3-role-trust-policy.json 的文件。此策略会向 Amazon S3 授予代入该角色的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- b. 复制以下策略并将其保存到名为 s3-role-permissions-policy.json 的文件。此访问策略授予对各种 Amazon S3 存储桶和对象操作的权限。在以下步骤中，您将策略添加到所创建的 IAM 角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionAcl"
      ],
      "Resource": [
        "arn:aws:s3:::source-bucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3>ListBucket",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::source-bucket"
      ]
    }
  ]
}
```

```
        "s3:GetReplicationConfiguration"
    ],
    "Resource": [
        "arn:aws:s3:::source-bucket"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete"
    ],
    "Resource": "arn:aws:s3:::destination-bucket/*"
}
]
```

- c. 运行以下 AWS CLI 命令以创建角色：

```
$ aws iam create-role \
--role-name crrRole \
--assume-role-policy-document file://S3-role-trust-policy.json \
--profile accountA
```

- d. 运行以下 AWS CLI 命令以创建策略：

```
$ aws iam create-policy \
--policy-name crrRolePolicy \
--policy-document file://S3-role-permissions-policy.json \
--profile accountA
```

- e. 记下 create-policy 命令返回的策略 Amazon 资源名称 (ARN)。

- f. 运行以下 AWS CLI 命令以将策略附加到角色：

```
$ aws iam attach-role-policy \
--role-name crrRole \
--policy-arn policy-arn \
--profile accountA
```

现在，您的账户 A 中已有 Amazon S3 可代入的 IAM 角色。它拥有必要的 Amazon S3 操作的权限，以便 Amazon S3 能够将对象从特定的源存储桶复制到目标存储桶。在将跨区域复制添加到账户 A 中的源存储桶时，可以指定此角色。

4. 在账户 A 中的源存储桶上添加复制配置，指示 Amazon S3 将具有前缀 Tax/ 的对象复制到目标存储桶，如以下示例配置所示。

**Important**

将复制配置添加到存储桶时，您必须具有 `iam:PassRole` 权限才能传递授予 Amazon S3 复制权限的 IAM 角色。IAM 角色由在复制配置 XML 中的 `<Role>` 元素中使用的 Amazon 资源名称 (ARN) 指定。有关更多信息，请参阅 IAM 用户指南中的[授予向 AWS 服务传递角色的用户权限](#)。

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Role>arn:aws:iam::AWS-ID-Account-A:role/role-name</Role>
<Rule>
    <Status>Enabled</Status>
    <Prefix>Tax/<Prefix>
    <Destination><Bucket>arn:aws:s3:::destination-bucket</Bucket></Destination>
</Rule>
</ReplicationConfiguration>
```

在此示例中，您可以使用 AWS CLI、Amazon S3 控制台或 AWS 开发工具包添加复制配置。

- 使用 AWS CLI.

AWS CLI 要求以 JSON 形式指定复制配置。请将以下 JSON 保存在一个文件 (`replication.json`) 中。

```
{  
    "Role": "arn:aws:iam::AWS-ID-Account-A:role/role-name",  
    "Rules": [  
        {  
            "Prefix": "Tax",  
            "Status": "Enabled",  
            "Destination": {  
                "Bucket": "arn:aws:s3:::destination-bucket"  
            }  
        }  
    ]  
}
```

通过提供存储桶名称和角色 ARN 来更新 JSON。随后运行 AWS CLI 命令以向源存储桶添加复制配置：

```
$ aws s3api put-bucket-replication \  
--bucket source-bucket \  
--replication-configuration file://replication.json \  
--profile accountA
```

有关如何设置 AWS CLI 的说明，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

账户 A 可以使用 `get-bucket-replication` 命令检索复制配置：

```
$ aws s3api get-bucket-replication \  
--bucket source-bucket \  
--profile accountA
```

- 使用 Amazon S3 控制台.

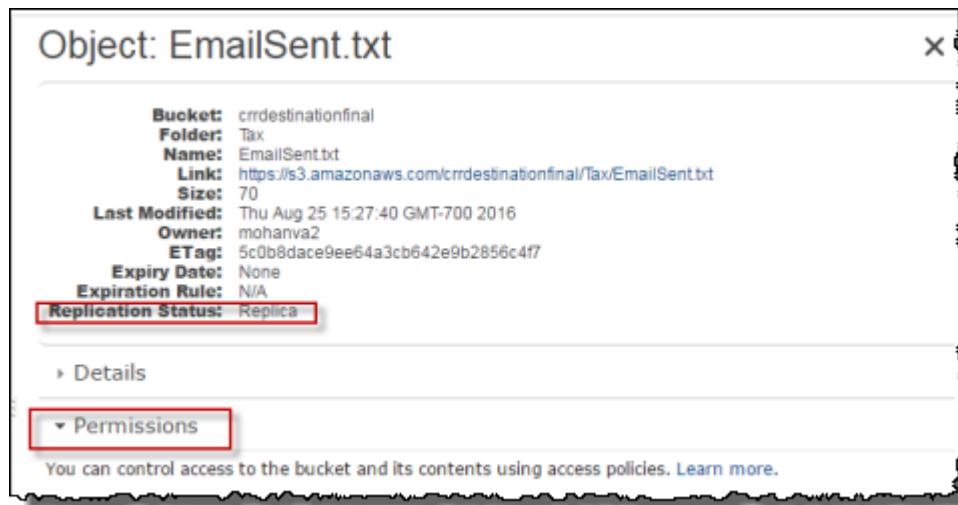
有关使用控制台的说明，请参阅[如何为 S3 存储桶启用和配置跨区域复制？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

- 使用 AWS SDK for Java.

有关代码示例，请参阅[使用AWS SDK for Java设置跨区域复制 \(p. 468\)](#)。

5. 测试设置。在 控制台中，执行以下操作：

- 在源存储桶中，创建一个名为 Tax 的文件夹。
- 将对象添加到源存储桶中的文件夹。
  - 验证 Amazon S3 是否已复制账户 B 拥有的目标存储桶中的对象。
  - 在对象属性中，注意 Replication Status 将设置为“副本”(将其标识为副本对象)。
  - 在对象属性中，权限部分未显示任何权限(副本仍将由源存储桶拥有者拥有，并且目标存储桶拥有者不具有对象副本的权限)。您可以添加可选配置以指示 Amazon S3 更改副本所有权。有关示例请查看[演练 3：将副本拥有者更改为目标存储桶拥有者 \(p. 463\)](#)。



Amazon S3 复制对象所需的时间量取决于对象大小。有关查找复制状态的信息，请参阅[查看跨区域复制状态 \(p. 472\)](#)。

- 在源存储桶中更新对象的 ACL，并验证更改是否出现在目标存储桶中。

有关说明，请参阅[如何在对象上设置权限？](#) (在 Amazon Simple Storage Service 控制台用户指南中)。

- 更新对象的元数据。例如，更改存储类。验证更改是否出现在目标存储桶中。

有关说明，请参阅[如何向 S3 对象添加元数据？](#) (在 Amazon Simple Storage Service 控制台用户指南中)。

请记住，副本是源存储桶中对象的精确副本。

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

[复制和不复制的内容 \(p. 444\)](#)

[查看跨区域复制状态 \(p. 472\)](#)

[演练 1：配置跨区域复制（其中源存储桶和目标存储桶由同一 AWS 账户拥有）\(p. 457\)](#)

## 跨区域复制：其他演练

### 主题

- [演练 3：将副本拥有者更改为目标存储桶拥有者 \(p. 463\)](#)
- [CRR 演练 4：指示 Amazon S3 复制使用 AWS KMS 托管加密密钥通过服务器端加密创建的对象 \(p. 465\)](#)

此部分提供了在存储桶跨区域复制配置中添加可选选项的示例。

## 演练 3：将副本拥有者更改为目标存储桶拥有者

在本练习中，您更新练习 2 ([演练 2：配置跨区域复制（其中源存储桶和目标存储桶由不同 AWS 账户拥有）\(p. 458\)](#)) 中的复制配置以指示 Amazon S3 将副本拥有者更改为拥有目标存储桶的 AWS 账户。有关更改副本所有权的更多信息，请参阅[跨区域复制的其他配置：更改副本拥有者 \(p. 451\)](#)。

1. 完成演练 2。有关说明，请参阅[演练 2：配置跨区域复制（其中源存储桶和目标存储桶由不同 AWS 账户拥有）\(p. 458\)](#)。
2. 通过添加 `<AccessControlTranslation>` 元素更新复制配置规则，如下所示：

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix></Prefix>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
      <Account>destination-bucket-owner-account-id</Account>
      <StorageClass>storage-class</StorageClass>
      <AccessControlTranslation>
        <Owner>Destination</Owner>
      </AccessControlTranslation>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

在此示例中，您可以使用 AWS CLI 或 AWS 开发工具包添加复制配置。由于控制台不支持在其他 AWS 账户中指定目标存储桶，因此您不能使用控制台。

- 使用 AWS CLI.

AWS CLI 要求以 JSON 形式指定复制配置。请将以下 JSON 保存在一个文件 (`replication.json`) 中。

```
{
  "Role": "arn:aws:iam::AWS-ID-Account-A:role/role-name",
  "Rules": [
    {
      "Prefix": "Tax",
      "Status": "Enabled",
      "Destination": {
        "Bucket": "arn:aws:s3:::destination-bucket",
        "AccessControlTranslation" : {
          "Owner" : "Destination"
        }
      }
    }
  ]
}
```

通过提供存储桶名称和角色 Amazon 资源名称 (ARN) 来更新 JSON。随后运行 AWS CLI 命令以向源存储桶添加复制配置：

```
$ aws s3api put-bucket-replication \
--bucket source-bucket \
--replication-configuration file://replication.json \
--profile accountA
```

有关如何设置 AWS CLI 的说明，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

您可以使用 `get-bucket-replication` 命令检索复制配置：

```
$ aws s3api get-bucket-replication \
--bucket source-bucket \
--profile accountA
```

- 使用 AWS SDK for Java.

有关代码示例，请参阅 [使用AWS SDK for Java设置跨区域复制 \(p. 468\)](#)。

3. 在 IAM 控制台中，选择您创建的 IAM 角色，并通过添加 `s3:ObjectOwnerOverrideToBucketOwner` 操作的权限来更新关联的权限策略。

将显示更新后的策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetReplicationConfiguration",
        "s3>ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::source-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3.GetObjectVersionForReplication",
        "s3GetObjectVersionAcl"
      ],
      "Resource": [
        "arn:aws:s3:::source-bucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ObjectOwnerOverrideToBucketOwner"
      ],
      "Resource": "arn:aws:s3:::destination-bucket/*"
    }
  ]
}
```

4. 在 Amazon S3 控制台中，选择目标存储桶并更新存储桶策略，如下所示：

- 向源对象拥有者授予 `s3:ObjectOwnerOverrideToBucketOwner` 操作的权限。
- 向源存储桶拥有者授予 `s3>ListBucket` 和 `s3>ListBucketVersions` 操作的权限。

以下存储桶策略显示其他权限。

```
{
  "Version": "2008-10-17",
  "Id": "PolicyForDestinationBucket",
  "Statement": [
```

```
{  
    "Sid":"1",  
    "Effect":"Allow",  
    "Principal":{  
        "AWS":"source-bucket-owner-aws-account-id"  
    },  
    "Action": [  
        "s3:ReplicateDelete",  
        "s3:ReplicateObject",  
        "s3:ObjectOwnerOverrideToBucketOwner"  
    ],  
    "Resource": "arn:aws:s3:::destinationbucket/*"  
},  
{  
    "Sid":"2",  
    "Effect":"Allow",  
    "Principal":{  
        "AWS":"source-bucket-owner-aws-account-id"  
    },  
    "Action": [  
        "s3>ListBucket",  
        "s3>ListBucketVersions"  
    ],  
    "Resource": "arn:aws:s3:::destinationbucket"  
}  
]  
}
```

5. 在 Amazon S3 控制台中测试复制配置：

- a. 将对象上传到源存储桶 (在 Tax 文件夹中)。
- b. 验证是否已在目标存储桶中创建副本。对于副本，验证权限。请注意，目标存储桶拥有者现在具有对象副本的完全权限。

## CRR 演练 4：指示 Amazon S3 复制使用 AWS KMS 托管加密密钥通过服务器端加密创建的对象

您的源存储桶中可以有使用 AWS KMS 托管密钥通过服务器端加密创建的对象。默认情况下，Amazon S3 不会复制这些对象。但是，您可以向存储桶复制配置添加可选配置以指示 Amazon S3 复制这些对象。

在本练习中，您首先在跨账户方案 (源存储桶和目标存储桶由不同的 AWS 账户拥有) 中设置复制配置。随后，此部分提供了有关更新配置以指示 Amazon S3 复制使用 AWS KMS 托管密钥加密的对象的说明。

### Note

虽然此示例使用现有演练来在跨账户方案中设置 CRR，但在源存储桶和目标存储桶具有相同的拥有者时也可配置 SSE-KMS 加密的对象的复制。

1. 完成 CRR 演练 2。有关说明，请参阅[演练 2：配置跨区域复制 \(其中源存储桶和目标存储桶由不同 AWS 账户拥有\) \(p. 458\)](#)。
2. 将源存储桶上的复制配置替换为以下内容 (这将添加指示 Amazon S3 复制使用 AWS KMS 密钥加密的源对象的选项)。

```
<ReplicationConfiguration>  
  <Role>IAM role ARN</Role>  
  <Rule>  
    <Prefix>Tax</Prefix>  
    <Status>Enabled</Status>  
    <SourceSelectionCriteria>  
      <SseKmsEncryptedObjects>  
        <Status>Enabled</Status>
```

```
</SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
    <Bucket>arn:aws:s3:::dest-bucket-name</Bucket>
    <EncryptionConfiguration>
        <ReplicaKmsKeyID>AWS KMS key ID to use for encrypting object replicas.</ReplicaKmsKeyID>
    </EncryptionConfiguration>
</Destination>
</Rule>
</ReplicationConfiguration>
```

在此示例中，您可以使用 AWS CLI 或 AWS 开发工具包添加复制配置。

- 使用 AWS CLI.

AWS CLI 要求以 JSON 形式指定复制配置。请将以下 JSON 保存在一个文件 (`replication.json`) 中。

```
{
    "Role": "IAM role ARN",
    "Rules": [
        {
            "Prefix": "Tax",
            "Status": "Enabled",
            "SourceSelectionCriteria": {
                "SseKmsEncryptedObjects" : {
                    "Status" : "Enabled"
                }
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::dest-bucket-name",
                "EncryptionConfiguration" : {
                    "ReplicaKmsKeyID": "AWS KMS key ARN(created in the same region as the destination bucket.)"
                }
            }
        }
    ]
}
```

通过提供存储桶名称和角色 ARN 来更新 JSON。随后运行 AWS CLI 命令以向源存储桶添加复制配置：

```
$ aws s3api put-bucket-replication \
--bucket source-bucket \
--replication-configuration file://replication.json \
--profile accountA
```

有关如何设置 AWS CLI 的说明，请参阅[设置用于示例演练的工具 \(p. 256\)](#)。

账户 A 可以使用 `get-bucket-replication` 命令检索复制配置：

```
$ aws s3api get-bucket-replication \
--bucket source-bucket \
--profile accountA
```

- 使用 AWS SDK for Java.

有关代码示例，请参阅[使用AWS SDK for Java设置跨区域复制 \(p. 468\)](#)。

3. 通过添加 AWS KMS 操作的权限来更新 IAM 角色的权限策略。

```
{  
    "Action": [  
        "kms:Decrypt"  
    ],  
    "Effect": "Allow",  
    "Condition": {  
        "StringLike": {  
            "kms:ViaService": "s3.source-bucket-region.amazonaws.com",  
            "kms:EncryptionContext:aws:s3:arn": [  
                "arn:aws:s3:::source-bucket-name/Tax"  
            ]  
        }  
    },  
    "Resource": [  
        "List of AWS KMS key IDs used to encrypt source objects."  
    ]  
},  
{  
    "Action": [  
        "kms:Encrypt"  
    ],  
    "Effect": "Allow",  
    "Condition": {  
        "StringLike": {  
            "kms:ViaService": "s3.dest-bucket-region.amazonaws.com",  
            "kms:EncryptionContext:aws:s3:arn": [  
                "arn:aws:s3:::dest-bucket-name/Tax"  
            ]  
        }  
    },  
    "Resource": [  
        "List of AWS KMS key IDs that you want S3 to use to encrypt object replicas."  
    ]  
}
```

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:GetObjectVersionForReplication",  
                "s3:GetObjectVersionAcl"  
            ],  
            "Resource": [  
                "arn:aws:s3:::source-bucket/Tax"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListBucket",  
                "s3:GetReplicationConfiguration"  
            ],  
            "Resource": [  
                "arn:aws:s3:::source-bucket"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:ReplicateObject",  
                "s3:PutObjectAcl"  
            ],  
            "Resource": [  
                "arn:aws:s3:::source-bucket/replica-prefix"  
            ]  
        }  
    ]  
}
```

```
        "s3:ReplicateDelete"
    ],
    "Resource": "arn:aws:s3:::dest-bucket/*"
},
{
    "Action": [
        "kms:Decrypt"
    ],
    "Effect": "Allow",
    "Condition": {
        "StringLike": {
            "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
            "kms:EncryptionContext:aws:s3:arn": [
                "arn:aws:s3:::source-bucket/Tax*"
            ]
        }
    },
    "Resource": [
        "List of AWS KMS key IDs used to encrypt source objects."
    ]
},
{
    "Action": [
        "kms:Encrypt"
    ],
    "Effect": "Allow",
    "Condition": {
        "StringLike": {
            "kms:ViaService": "s3.dest-bucket-region.amazonaws.com",
            "kms:EncryptionContext:aws:s3:arn": [
                "arn:aws:s3:::dest-bucket/Tax*"
            ]
        }
    },
    "Resource": [
        "List of AWS KMS key IDs that you want S3 to use to encrypt object replicas."
    ]
}
]
```

4. 测试设置。在控制台中，使用 AWS KMS 托管密钥将一个对象上传到源存储桶（在 /Tax 文件夹中）。验证 Amazon S3 是否已在目标存储桶中复制对象。

## 使用控制台设置跨区域复制

源存储桶和目标存储桶由同一个 AWS 账户拥有时，可以使用 Amazon S3 控制台对源存储桶添加复制配置。有关更多信息，请参阅以下主题：

- [演练 1：配置跨区域复制（其中源存储桶和目标存储桶由同一 AWS 账户拥有）\(p. 457\)](#)
- [如何为 S3 存储桶启用和配置跨区域复制？\(在 Amazon Simple Storage Service 控制台用户指南 中\)。](#)
- [跨区域复制 \(CRR\) \(p. 442\)](#)
- [设置跨区域复制 \(p. 445\)](#)

## 使用AWS SDK for Java设置跨区域复制

当源存储桶和目标存储桶由两个不同的 AWS 账户拥有时，可以使用 AWS CLI 或其中一个 AWS 开发工具包向源存储桶添加复制配置。不能使用控制台添加复制配置，因为在向源存储桶添加复制配置时，控

制台不会提供指定由另一个 AWS 账户拥有的目标存储桶的方法。有关更多信息，请参阅[设置跨区域复制 \(p. 445\)](#)。

以下示例向存储桶添加复制配置，然后检索并验证该配置。有关创建和测试有效示例的说明，请参阅[测试 Amazon S3 Java 代码示例 \(p. 522\)](#)。

```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketReplicationConfiguration;
import com.amazonaws.services.s3.model.ReplicationDestinationConfig;
import com.amazonaws.services.s3.model.ReplicationRule;
import com.amazonaws.services.s3.model.ReplicationRuleStatus;
import com.amazonaws.services.s3.model.StorageClass;

public class CrossRegionReplication {

    public static void main(String[] args) throws IOException {
        String clientRegion = "**** Client region ****";
        String accountId = "**** Account ID ****";
        String roleName = "**** Role name ****";
        String sourceBucketName = "**** Source bucket name ****";
        String destBucketName = "**** Destination bucket name ****";
        String prefix = "Tax/";

        String roleARN = String.format("arn:aws:iam::%s:role/%s", accountId, roleName);
        String destinationBucketARN = "arn:aws:s3:::" + destBucketName;

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Create the replication rule.
            Map<String, ReplicationRule> replicationRules = new HashMap<String,
            ReplicationRule>();
            replicationRules.put("ReplicationRule1",
                new ReplicationRule()
                    .withStatus(ReplicationRuleStatus.Enabled)
                    .withPrefix(prefix)
                    .withDestinationConfig(new
            ReplicationDestinationConfig()
                .withBucketARN(destinationBucketARN)
                .withStorageClass(StorageClass.Standard)));

            // Save the replication rule to the source bucket.
            s3Client.setBucketReplicationConfiguration(sourceBucketName,
                new BucketReplicationConfiguration()
                    .withRoleARN(roleARN)

            .withRules(replicationRules));

            // Retrieve the replication configuration and verify that the configuration
            // matches the rule we just set.
            BucketReplicationConfiguration replicationConfig =
            s3Client.getBucketReplicationConfiguration(sourceBucketName);
```

```
        ReplicationRule rule = replicationConfig.getRule("ReplicationRule1");
        System.out.println("Retrieved destination bucket ARN: " +
rule.getDestinationConfig().getBucketARN());
        System.out.println("Retrieved source-bucket replication rule prefix: " +
rule.getPrefix());
        System.out.println("Retrieved source-bucket replication rule status: " +
rule.getStatus());
    }
    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();
    }
}
}
```

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

[设置跨区域复制 \(p. 445\)](#)

## 使用适用于 .NET 的 AWS 开发工具包设置跨区域复制

源存储桶和目标存储桶由两个不同的 AWS 账户拥有时，可以使用 AWS CLI 或一个 AWS 开发工具包对源存储桶添加复制配置。不能使用控制台添加复制配置，因为在对源存储桶添加复制配置时，控制台不提供指定由另一个 AWS 账户拥有的目标存储桶的方法。有关更多信息，请参阅 [设置跨区域复制 \(p. 445\)](#)。

以下适用于 .NET 的 AWS 开发工具包代码示例首先向存储桶添加复制配置，然后检索该配置。您需要通过提供存储桶名称和 IAM 角色 ARN 来更新代码。有关如何创建和测试有效示例的说明，请参阅 [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-
developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class CrossRegionReplicationTest
    {
        private const string sourceBucket = "*** source bucket ***";
        // Bucket ARN example - arn:aws:s3:::destinationbucket
        private const string destinationBucketArn = "*** destination bucket ARN ***";
        private const string roleArn = "*** IAM Role ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint sourceBucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(sourceBucketRegion);
            EnableReplicationAsync().Wait();
        }
        static async Task EnableReplicationAsync()
```

```
{  
    try  
    {  
        ReplicationConfiguration replConfig = new ReplicationConfiguration  
        {  
            Role = roleArn,  
            Rules =  
            {  
                new ReplicationRule  
                {  
                    Prefix = "Tax",  
                    Status = ReplicationRuleStatus.Enabled,  
                    Destination = new ReplicationDestination  
                    {  
                        BucketArn = destinationBucketArn  
                    }  
                }  
            }  
        };  
  
        PutBucketReplicationRequest putRequest = new PutBucketReplicationRequest  
        {  
            BucketName = sourceBucket,  
            Configuration = replConfig  
        };  
  
        PutBucketReplicationResponse putResponse = await  
s3Client.PutBucketReplicationAsync(putRequest);  
  
        // Verify configuration by retrieving it.  
        await RetrieveReplicationConfigurationAsync(s3Client);  
    }  
    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);  
    }  
}  
private static async Task RetrieveReplicationConfigurationAsync(IAmazonS3 client)  
{  
    // Retrieve the configuration.  
    GetBucketReplicationRequest getRequest = new GetBucketReplicationRequest  
    {  
        BucketName = sourceBucket  
    };  
    GetBucketReplicationResponse getResponse = await  
client.GetBucketReplicationAsync(getRequest);  
    // Print.  
    Console.WriteLine("Printing replication configuration information...");  
    Console.WriteLine("Role ARN: {0}", getResponse.Configuration.Role);  
    foreach (var rule in getResponse.Configuration.Rules)  
    {  
        Console.WriteLine("ID: {0}", rule.Id);  
        Console.WriteLine("Prefix: {0}", rule.Prefix);  
        Console.WriteLine("Status: {0}", rule.Status);  
    }  
}
```

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

[设置跨区域复制 \(p. 445\)](#)

# 查看跨区域复制状态

您可以使用 Amazon S3 清单功能获取存储桶中的所有对象的复制状态。随后，Amazon S3 会将一个 .csv 文件传送给配置的目标存储桶。有关 Amazon S3 清单的更多信息，请参阅 [Amazon S3 清单 \(p. 235\)](#)。

如果您要获取单个对象的 CRR 状态，请阅读以下内容：

在跨区域复制中，您有一个源存储桶（对它配置复制）和一个目标存储桶（Amazon S3 将对象复制到其中）。当您请求这些存储桶中的对象（GET 对象）或对象元数据（HEAD 对象）时，Amazon S3 将在响应中返回 `x-amz-replication-status` 标头，如下所示：

- 如果请求源存储桶中的对象 – Amazon S3 将返回 `x-amz-replication-status` 标头（如果请求中的对象符合复制条件）。

例如，假设您在复制配置中指定了对象前缀 `TaxDocs`，从而请求 Amazon S3 复制具有键名称前缀 `TaxDocs` 的对象。那么，使用此键名称前缀上传的任何对象（例如 `TaxDocs/document1.pdf`）都符合复制条件。对于使用此键名称前缀请求的任何对象，Amazon S3 将返回 `x-amz-replication-status` 标头与对象复制状态的以下值之一：`PENDING`、`COMPLETED` 或 `FAILED`。

### Note

在上传对象后，如果对象复制失败，将没有机制来重试失败的复制。您必须再次为 Amazon S3 上传此对象以复制该对象。

- 如果请求目标存储桶中的对象 – Amazon S3 将 `x-amz-replication-status` 标头和值 `REPLICA`（如果请求中的对象是 Amazon S3 创建的副本）。

您可以在控制台中、使用 AWS CLI 或以编程方式使用 AWS 开发工具包查找对象复制状态。

- 在控制台中，选择对象，然后选择 Properties 以查看对象属性（包括复制状态）。
- 可以使用如下所示的 `head-object` AWS CLI 命令检索对象元数据信息：

```
aws s3api head-object --bucket source-bucket --key object-key --version-id object-version-id
```

该命令会返回对象元数据信息（包括 `ReplicationStatus`），如以下示例响应所示：

```
{  
    "AcceptRanges": "bytes",  
    "ContentType": "image/jpeg",  
    "LastModified": "Mon, 23 Mar 2015 21:02:29 GMT",  
    "ContentLength": 3191,  
    "ReplicationStatus": "COMPLETED",  
    "VersionId": "jfnW.HIMOFYiD_9rGbSkmroXsFj3fqZ.",  
    "ETag": "\"6805f2cfc46c0f04559748bb039d69ae\"",  
    "Metadata": {}  
}
```

- 可以使用 AWS 开发工具包检索对象的复制状态。以下是使用 AWS SDK for Java 和适用于 .NET 的 AWS 开发工具包的代码片段。

- AWS SDK for Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
    bucketName);
metadataRequest.setKey(key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
    metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

- 适用于 .NET 的 AWS 开发工具包

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
{
    BucketName = sourceBucket,
    Key        = objectKey
};

GetObjectMetadataResponse getmetadataResponse =
    client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
    getmetadataResponse.ReplicationStatus);
```

#### Note

如果您决定从启用了复制的源存储桶中删除对象，则应在删除之前检查对象的复制状态，以确保复制了对象。

如果对源存储桶启用了生命周期配置，则 Amazon S3 将搁置所有生命周期操作，直到它将对象状态标记为 COMPLETED 或 FAILED。

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

## 在 Amazon S3 中排查跨区域复制问题

在配置跨区域复制之后，如果您未看到目标存储桶中创建的对象副本，请尝试以下故障排除方法：

- Amazon S3 复制对象所需的时间取决于对象大小。对于大型对象，可能需要几个小时。如果所涉对象较大，请稍后再次检查复制的对象是否出现在目标存储桶中。
- 在源存储桶上的复制配置中：
  - 验证目标存储桶的 Amazon 资源名称 (ARN) 是否正确。
  - 验证键名称前缀是否正确。例如，如果将配置设置为复制具有前缀 Tax 的对象，则仅复制具有 Tax/document1 或 Tax/document2 等键名称的对象。不会复制具有键名称 document3 的对象。
  - 验证状态是否为 enabled。
- 如果目标存储桶由另一个 AWS 账户拥有，请验证存储桶拥有者是否对目标存储桶设置了允许源存储桶拥有者复制对象的存储桶策略。
- 如果对象副本未出现在目标存储桶中，请注意以下情况：
  - 若源存储桶中的某对象本身是另一个复制配置所创建的副本，Amazon S3 不会复制该副本。例如，如果您设置从存储桶 A 到存储桶 B 再到存储桶 C 的复制配置，则 Amazon S3 不会复制存储桶 B 中的对象副本。
  - 存储桶拥有者可以向其他 AWS 账户授予上传对象的权限。默认情况下，存储桶拥有者对于其他账户创建的对象没有任何权限。复制配置仅复制存储桶拥有者具有访问权限的对象。存储桶拥有者可以向其他

AWS 账户授予创建对象的权限，从而有条件地要求针对这些对象的明确访问权限。有关策略示例，请参阅[在授予上传对象的交叉账户许可的同时，确保存储桶拥有者拥有完全控制 \(p. 309\)](#)。

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

# 跨区域复制：其他注意事项

除了复制配置之外，Amazon S3 还支持多种其他存储桶配置选项，包括：

- 对存储桶配置版本控制。有关更多信息，请参阅[使用版本控制 \(p. 380\)](#)。
- 为网站托管配置存储桶。有关更多信息，请参阅[在 Amazon S3 上托管静态网站 \(p. 401\)](#)。
- 通过存储桶策略或 ACL 配置存储桶访问。有关详细信息，请参阅[使用存储桶策略和用户策略 \(p. 279\)](#) 和参阅[使用 ACL 管理访问 \(p. 333\)](#)。
- 配置存储桶以存储访问日志。有关更多信息，请参阅[Amazon S3 服务器访问日志记录 \(p. 506\)](#)。
- 为存储桶中的对象配置生命周期。有关更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

本主题说明存储桶复制配置如何影响其他存储桶配置的行为。

## 生命周期配置和对象副本

Amazon S3 复制对象所需的时间取决于对象大小。对于大型对象，可能需要几个小时。虽然副本可能需要一段时间才能出现在目标存储桶中，但副本的创建时间与源存储桶中的对应对象相同。因此，如果您对目标存储桶设置了生命周期策略，请注意，生命周期规则遵循对象的原始创建时间，而不是副本在目标存储桶中出现的时间。

如果您在不受版本控制的存储桶中具有对象过期生命周期策略，并且希望在启用版本控制时保持相同的永久删除行为，则必须添加非当前过期策略以管理启用版本控制的存储桶中非当前对象版本的删除。

复制配置需要启用版本控制的存储桶。在存储桶上启用版本控制时，请注意以下几点：

- 如果您拥有对象 `Expiration` 生命周期策略，在启用版本控制后，则应添加 `NonCurrentVersionExpiration` 策略以保持相同的永久删除行为（这是启用版本控制之前的情况）。
- 如果拥有 `Transition` 生命周期策略，则在启用版本控制后，应考虑添加 `NonCurrentVersionTransition` 策略。

## 版本控制配置和复制配置

对存储桶配置复制时，源存储桶和目标存储桶都必须启用版本控制。对源存储桶和目标存储桶启用版本控制并对源存储桶配置复制后，请注意以下几点：

- 如果您尝试对源存储桶禁用版本控制，则 Amazon S3 会返回错误。您必须先删除复制配置，然后才能对源存储桶禁用版本控制。
- 如果您对目标存储桶禁用版本控制，则 Amazon S3 会停止复制。

## 日志记录配置和复制配置

请注意以下几点：

- 如果您让 Amazon S3 将日志传送到已启用复制的存储桶，则 Amazon S3 将复制日志对象。
- 如果您已对源存储桶或目标存储桶启用服务器访问日志 ([Amazon S3 服务器访问日志记录 \(p. 506\)](#)) 或 AWS CloudTrail 日志 ([使用 AWS CloudTrail 记录 Amazon S3 API 调用 \(p. 488\)](#))，则 Amazon S3 将在日志中包含与 CRR 相关的请求。例如，Amazon S3 将记录它复制的每个对象。

## CRR 和目标区域

在 CRR 配置中，源存储桶和目标存储桶必须处于不同的 AWS 区域。您可以根据您的业务需求或成本注意事项选择目标存储桶区域。例如，区域间数据传输费用因区域配对而异。例如，假定 美国东部（弗吉尼亚北部）(us-east-1) 为您的源存储桶区域。如果您选择 美国西部（俄勒冈）(us-west-2) 作为目标存储桶区域，则在选择 美国东部（俄亥俄州）(us-east-2) 区域时，您需要支付更多费用。有关定价信息，请参阅 [Amazon S3 定价](#) 中的“数据传输定价”部分。

## 暂停复制配置

如果您希望 Amazon S3 暂停复制，则可在复制配置中禁用特定规则。如果启用复制并删除向 Amazon S3 授予必要权限的 IAM 角色，则 Amazon S3 将无法复制对象并会将这些对象的复制状态报告为失败。

## 相关主题

[跨区域复制 \(CRR\) \(p. 442\)](#)

# 请求路由选择

## 主题

- [请求重定向和 REST API \(p. 476\)](#)
- [DNS 注意事项 \(p. 479\)](#)

向使用 `<CreateBucketConfiguration>` API 创建的存储桶发送请求的程序必须支持重定向。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

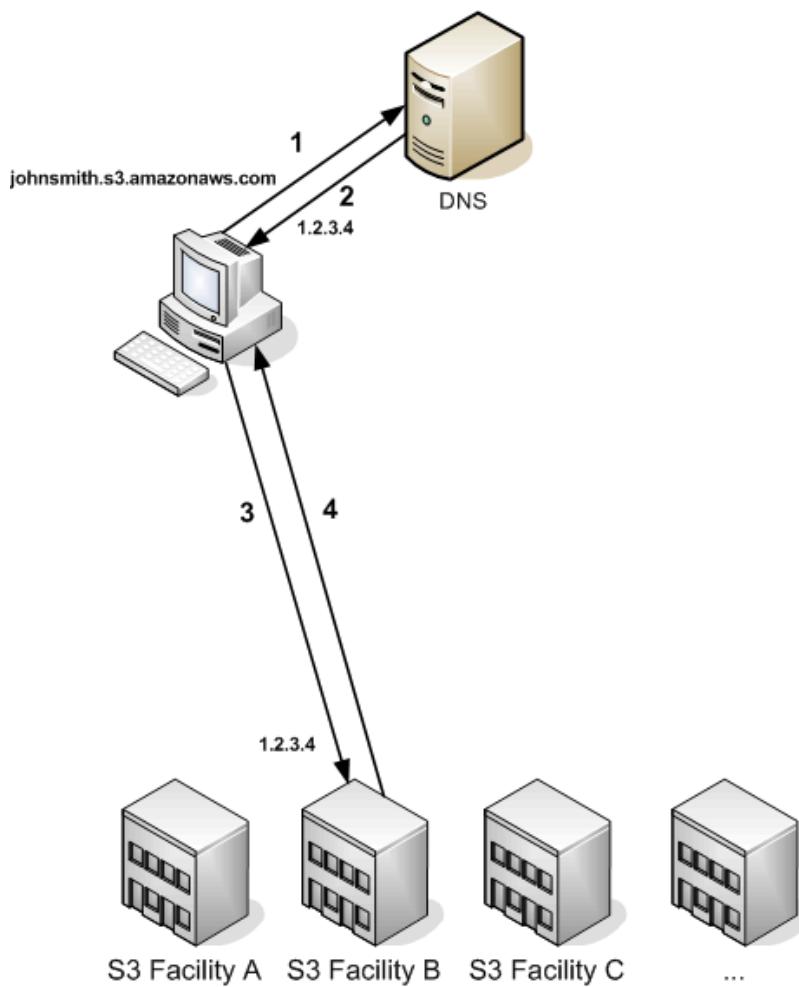
要使用此功能，您必须拥有一个可以处理 Amazon S3 重定向响应的应用程序。唯一的例外是专用于处理存储桶（在不使用 `<CreateBucketConfiguration>` 的情况下创建）的应用程序。有关存储桶位置约束的更多信息，请参阅[访问存储桶 \(p. 51\)](#)。

## 主题

- [DNS 路由选择 \(p. 476\)](#)
- [临时请求重定向 \(p. 477\)](#)
- [永久请求重定向 \(p. 478\)](#)
- [请求重定向示例 \(p. 479\)](#)

## 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。

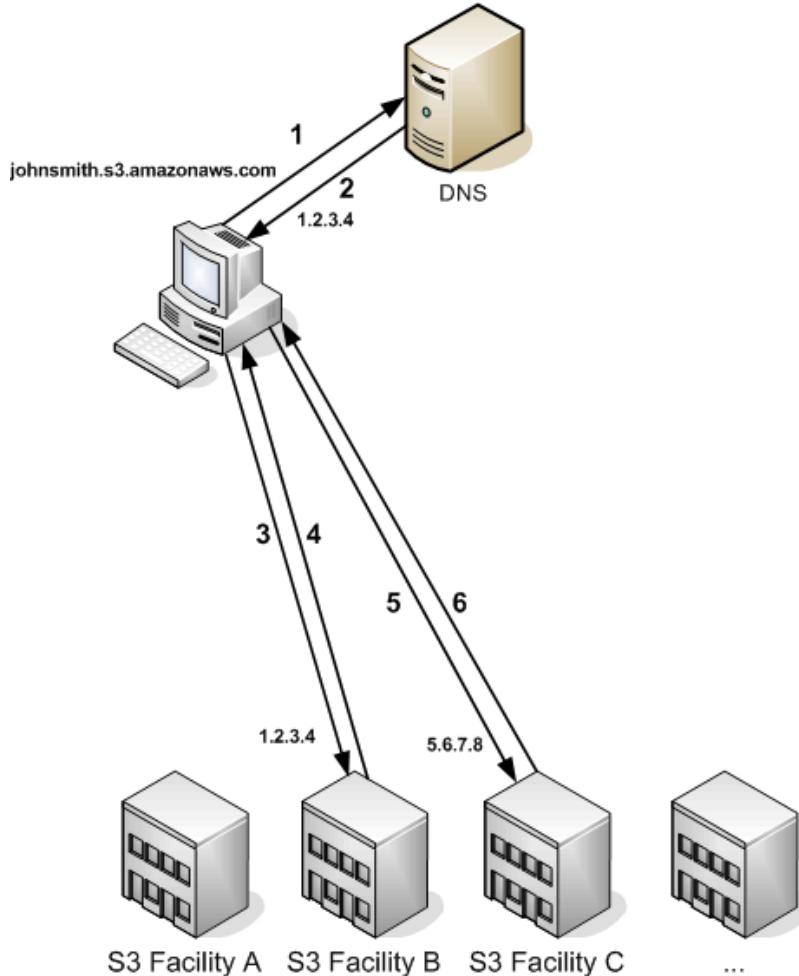
例如，如果您创建新的存储桶并立即向该存储桶发送请求，则根据存储桶的位置约束，您可能收到临时重定向。如果您在美国东部（弗吉尼亚北部）AWS 区域中创建了该存储桶，则看不到此重定向，因为这也是默认的 Amazon S3 终端节点。

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 [访问存储桶 \(p. 51\)](#)。

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://johnsmith.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>johnsmith.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

### SOAP API 临时请求重定向

#### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。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-gztb4pa9sq.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](#)。

# 性能优化

此部分通过以下主题讨论优化性能的 Amazon S3 最佳实践。

## 主题

- [请求速率和性能指南 \(p. 480\)](#)
- [TCP 窗口缩放 \(p. 480\)](#)
- [TCP 选择性认可 \(p. 480\)](#)

## Note

有关高性能优化的更多信息，请参阅 Pittsburgh Supercomputing Center (PSC) 网站上的[启用高性能数据传输](#)。

## 请求速率和性能指南

Amazon S3 自动扩展到高请求速率。例如，您的应用程序可以在存储桶中实现至少每秒每个前缀 3,500 个 PUT/POST/DELETE 请求和 5,500 个 GET 请求。对存储桶中的前缀数量没有限制。以指数方式提高读取或写入性能很简单。例如，如果您在 Amazon S3 存储桶中创建 10 个前缀以并行处理读取，则可以将读取性能扩展到每秒 55,000 个读取请求。

如果 Amazon S3 工作负载将服务器端加密与 AWS Key Management Service (SSE-KMS) 结合使用，请参阅 AWS Key Management Service Developer Guide 中的 [AWS KMS 限制](#)，以获取有关使用案例支持的请求速率的信息。

## GET 密集型工作负载

如果工作负载主要发送 GET 请求，则除了上述准则之外，还应考虑使用 Amazon CloudFront 实现性能优化。通过将 CloudFront 与 Amazon S3 集成，可以在向用户分发内容时同时实现低延迟和高数据传输速率。另外可以减少向 Amazon S3 发送的直接请求数，从而降低成本。

例如，假设有几个十分常用的对象。CloudFront 会从 Amazon S3 提取这些对象并对其进行缓存。CloudFront 随后可以从其缓存中为针对这些对象的更多请求提供服务，从而减少它发送到 Amazon S3 的 GET 请求数。有关更多信息，请参阅 [Amazon CloudFront 产品详细信息页面](#)。

## TCP 窗口缩放

通过支持大于 64 KB 的窗口大小，TCP 窗口缩放使您可以提高操作系统和应用程序层之间，以及应用程序层和 Amazon S3 之间的网络吞吐量性能。启动 TCP 会话时，客户端将公告其支持的接收窗口 WSCALE 因子，然后 Amazon S3 将为上游方向的客户端返回其支持的接收窗口 WSCALE 因子。

尽管 TCP 窗口缩放可以提高性能，但要正确设置它则非常困难。确保在应用程序和内核级别调整了设置。有关 TCP 窗口缩放的更多信息，请参阅您的操作系统文档并转到 [RFC 1323](#)。

## TCP 选择性认可

TCP 选择性认可旨在缩短大量数据包丢失后的恢复时间。大多数较新的操作系统都支持 TCP 选择性认可，但是可能需要先启用。有关 TCP 选择性认可的更多信息，请参阅您的操作系统随附的文档并转到 [RFC 2018](#)。

# 监控 Amazon S3

监控是保持 Amazon S3 和 AWS 解决方案的可靠性、可用性和性能的重要环节。您应从 AWS 解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。但是，在开始监控 Amazon S3 之前，您应创建一个可以回答以下问题的监控计划：

- 您的监控目标是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

## 主题

- [监控工具 \(p. 481\)](#)
- [使用 Amazon CloudWatch 来监控指标 \(p. 482\)](#)
- [存储桶的指标配置 \(p. 487\)](#)
- [使用 AWS CloudTrail 记录 Amazon S3 API 调用 \(p. 488\)](#)

## 监控工具

AWS 为您提供了各种可用于监控 Amazon S3 的工具。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

### 自动监控工具

您可以使用以下自动化监控工具来监控 Amazon S3 并在出现错误时进行报告：

- Amazon CloudWatch 警报 – 按您指定的时间段观察单个指标，并根据相对于给定阈值的指标值在若干时间段内执行一项或多项操作。该操作是向 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略发送通知。CloudWatch 警报将不会调用操作，因为这些操作处于特定状态；该状态必须改变并在指定数量的时间段内一直保持。有关更多信息，请参阅 [使用 Amazon CloudWatch 来监控指标 \(p. 482\)](#)。
- AWS CloudTrail 日志监控 – 在账户间共享日志文件，通过将 CloudTrail 日志文件发送到 CloudWatch Logs 对它们进行实时监控，在 Java 中编写日志处理应用程序，以及验证您的日志文件在被 CloudTrail 交付后未发生更改。有关更多信息，请参阅 [使用 AWS CloudTrail 记录 Amazon S3 API 调用 \(p. 488\)](#)。

### 手动监控工具

监控 Amazon S3 时的另一个重要环节是手动监控 CloudWatch 警报未涵盖的那些项。Amazon S3、CloudWatch、Trusted Advisor 和其他 AWS 控制台控制面板均提供 AWS 环境状态的概览视图。您可能需要启用服务器访问日志记录，以跟踪针对存储桶的访问请求。每个访问日志记录都提供有关单个访问请求的详细信息，如请求者、存储桶名称、请求时间、请求操作、响应状态和错误代码（如果有）。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 \(p. 506\)](#)。

- Amazon S3 控制面板显示：
  - 您的存储桶及其包含的对象和属性。
- CloudWatch 主页显示：

- 当前警报和状态。
- 警报和资源的图表。
- 服务运行状况。

此外，您还可以使用 CloudWatch 执行以下操作：

- 创建[自定义控制面板](#)以监控您关心的服务。
- 绘制指标数据图，以排除问题并弄清楚趋势。
- 搜索并浏览您所有的 AWS 资源指标。
- 创建和编辑警报以接收有关问题的通知。
- AWS Trusted Advisor 可帮助您监控 AWS 资源，以提高性能、可靠性、安全性和成本效益。四个 Trusted Advisor 检查可供所有用户使用；超过 50 个检查可供具有“商业”或“企业”支持计划的用户使用。有关更多信息，请参阅[AWS Trusted Advisor](#)。

Trusted Advisor 具有与 Amazon S3 相关的检查：

- Amazon S3 bucket 的日志记录配置的检查。
- 具有开放访问权限的 Amazon S3 bucket 的安全性检查。
- 未启用版本控制或已暂停版本控制的 Amazon S3 bucket 的容错检查。

## 使用 Amazon CloudWatch 来监控指标

Amazon S3 的 Amazon CloudWatch 指标可帮助您了解和提高使用 Amazon S3 的应用程序的性能。将 CloudWatch 与 Amazon S3 结合使用的方法有两种。

- 存储桶的每日存储指标 - 您可以使用 CloudWatch 监控 bucket 存储，此工具可收集来自 Amazon S3 存储的数据并将其处理为便于读取的每日指标。Amazon S3 的这些存储指标每天报告一次并提供给所有客户，无需额外费用。
- 请求指标 - 您可以选择监控 Amazon S3 请求，以快速确定运行问题并对其采取措施。这些指标在要处理的某些延迟后每隔 1 分钟提供一次。这些 CloudWatch 指标的计费费率与 Amazon CloudWatch 自定义指标的相同。有关 CloudWatch 定价的信息，请参阅[Amazon CloudWatch 定价](#)。要了解有关如何选择获取这些指标的更多信息，请参阅[存储桶的指标配置 \(p. 487\)](#)。

启用后，将为所有对象操作报告请求指标。默认情况下，这些 1 分钟指标在 Amazon S3 存储桶级别提供。您还可以为使用共享前缀或对象标签收集的指标定义筛选条件，这样您就可以使各个指标筛选条件满足特定业务应用程序、工作流或内部组织的需求。

所有 CloudWatch 统计数据会保留十五个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的运行情况。有关 CloudWatch 的更多信息，请参阅[什么是 Amazon CloudWatch、Amazon CloudWatch Events 和 Amazon CloudWatch Logs？](#) (位于 Amazon CloudWatch 用户指南中)。

## 指标与维度

下面列出了 Amazon S3 发送到 CloudWatch 的存储指标和维度。

## 存储桶的 Amazon S3 CloudWatch 每日存储指标

AWS/S3 命名空间包含存储桶的以下每日存储指标。

指标	说明
BucketSizeBytes	存储桶中存储的标准存储类的数据量 (以字节为单位)：标准 - 不经常访问 (Standard_IA) 存储类、OneZone - 不经常访问 (OneZone_IA)、低冗余存储 (RRS) 类或 Glacier (GLACIER) 存储类

指标	说明
	<p>有效的存储类型筛选条件 : StandardStorage、GlacierS3ObjectOverhead、StandardIAStorage、Standard和GlacierObjectOverhead (请参阅 StorageType 维度)</p> <p>单位 : 字节</p> <p>有效统计数据 : Average</p>
NumberOfObjects	<p>存储桶中存储的所有存储类的对象的总数</p> <p>有效的存储类型筛选条件 : AllStorageTypes (请参阅 StorageType 维度)</p> <p>单位 : 计数</p> <p>有效统计数据 : Average</p>

AWS/S3 命名空间包含以下请求指标。

## Amazon S3 CloudWatch 请求指标

指标	描述
AllRequests	<p>向 Amazon S3 存储桶提出的 HTTP 请求 (不论类型如何) 的总数。如果您要将某个指标配置用于某个筛选条件，那么该指标将仅返回向符合该筛选条件要求的存储桶中的对象提出的 HTTP 请求。</p> <p>单位 : 计数</p> <p>有效统计数据 : Sum</p>
GetRequests	<p>向 Amazon S3 存储桶中的对象提出的 HTTP GET 请求的数量。这不包括列表操作。</p> <p>单位 : 计数</p> <p>有效统计数据 : Sum</p> <p><b>Note</b></p> <p>面向分页列表的请求 (例如 <a href="#">List Multipart Uploads</a>、<a href="#">List Parts</a>、<a href="#">Get Bucket Object versions</a> 和其他请求) 未包含在此指标中。</p>
PutRequests	<p>向 Amazon S3 存储桶中的对象提出的 HTTP PUT 请求的数量。</p> <p>单位 : 计数</p> <p>有效统计数据 : Sum</p>
DeleteRequests	<p>向 Amazon S3 存储桶中的对象提出的 HTTP DELETE 请求的数量。这还包括 <a href="#">Delete Multiple Objects</a> 请求。此指标显示请求的数量，而不是删除的对象的数量。</p> <p>单位 : 计数</p> <p>有效统计数据 : Sum</p>
HeadRequests	向 Amazon S3 存储桶提出的 HTTP HEAD 请求的数量。

指标	描述
	<p>单位 : 计数</p> <p>有效统计数据 : Sum</p>
PostRequests	<p>向 Amazon S3 存储桶提出的 HTTP POST 请求的数量。</p> <p>单位 : 计数</p> <p>有效统计数据 : Sum</p> <p>Note</p> <p><a href="#">删除多个对象和 SELECT 对象内容请求未包含在此指标中。</a></p>
SelectRequests	<p>为 Amazon S3 存储桶中的对象发出的 Amazon S3 <a href="#">SELECT 对象内容</a>请求的数量。</p> <p>单位 : 计数</p> <p>有效统计数据 : Sum</p>
SelectScannedBytes	<p>使用 Amazon S3 存储桶中的 Amazon S3 <a href="#">SELECT 对象内容</a>请求扫描的数据的字节数。</p> <p>单位 : 字节</p> <p>有效统计数据 : Average (每个请求的字节数)、Sum (每个周期的字节数)、Sample Count、Min、Max</p>
SelectReturnedBytes	<p>使用 Amazon S3 存储桶中的 Amazon S3 <a href="#">SELECT 对象内容</a>请求返回的数据的字节数。</p> <p>单位 : 字节</p> <p>有效统计数据 : Average (每个请求的字节数)、Sum (每个周期的字节数)、Sample Count、Min、Max</p>
ListRequests	<p>列出存储桶内容的 HTTP 请求的数量。</p> <p>单位 : 计数</p> <p>有效统计数据 : Sum</p>
BytesDownloaded	<p>为向 Amazon S3 存储桶提出的请求下载的字节数 (请求的响应包含正文)。</p> <p>单位 : 字节</p> <p>有效统计数据 : Average (每个请求的字节数)、Sum (每个周期的字节数)、Sample Count、Min、Max</p>
BytesUploaded	<p>包含向 Amazon S3 存储桶提出的请求正文的已上传字节数。</p> <p>单位 : 字节</p> <p>有效统计数据 : Average (每个请求的字节数)、Sum (每个周期的字节数)、Sample Count、Min、Max</p>

指标	描述
4xxErrors	<p>向 Amazon S3 存储桶提出的值为 0 或 1 的 HTTP 4xx 客户端错误状态代码请求数。<code>average</code> 统计数据显示了错误率，<code>sum</code> 统计数据显示了每个周期内该类型的错误的计数。</p> <p>单位：计数</p> <p>有效统计数据：Average (每个请求的报告数)、Sum (每个周期的报告数)、Min、Max、Sample Count</p>
5xxErrors	<p>向 Amazon S3 存储桶提出的值为 0 或 1 的 HTTP 5xx 服务器错误状态代码请求数。<code>average</code> 统计数据显示了错误率，<code>sum</code> 统计数据显示了每个周期内该类型的错误的计数。</p> <p>单位：计数</p> <p>有效统计数据：Average (每个请求的报告数)、Sum (每个周期的报告数)、Min、Max、Sample Count</p>
FirstByteLatency	<p>从 Amazon S3 存储桶收到完整请求到开始返回响应的每请求时间。</p> <p>单位：毫秒</p> <p>有效统计数据：Average、Sum、Min、Max、Sample Count</p>
TotalRequestLatency	<p>从收到第一个字节到将最后一个字节发送到 Amazon S3 存储桶的已用每请求时间。这包括接收请求正文和发送响应正文所耗的时间 (未包含在 <code>FirstByteLatency</code> 中)。</p> <p>单位：毫秒</p> <p>有效统计数据：Average、Sum、Min、Max、Sample Count</p>

## Amazon S3 CloudWatch 维度

下列维度用于筛选 Amazon S3 指标。

维度	说明
BucketName	此维度筛选您仅为已识别存储桶请求的数据。
StorageType	此维度按存储类型筛选您已存储在存储桶中的数据。类型包括适用于 STANDARD 存储类的 StandardStorage、适用于 STANDARD_IA 存储类的 StandardIAStorage、适用于 ONEZONE_IA 存储类的 OneZoneIAStorage、适用于 REDUCED_REDUNDANCY 存储类的 ReducedRedundancyStorage、适用于 GLACIER 存储类的 GlacierStorage 和 AllStorageTypes。AllStorageTypes 类型包含 STANDARD、STANDARD_IA、ONEZONE_IA、REDUCED_REDUNDANCY 和 GLACIER 存储类。
FilterId	此维度将筛选您为存储桶上的请求指标指定的指标配置，例如前缀或标签。您将在创建指标配置时指定筛选条件 ID。有关更多信息，请参阅 <a href="#">存储桶的指标配置</a> 。

## 访问 CloudWatch 指标

您可以按照以下步骤查看 Amazon S3 的存储指标。请注意，要获取涉及的 Amazon S3 指标，您必须设置开始和结束时间戳。对于任意给定 24 小时内的指标，请将时间段设置为 86400 秒（一天的秒数）。另外，请记得设置 BucketName 和 StorageType 维度。

例如，如果您使用 AWS CLI 以字节为单位获取特定存储桶尺寸的平均值，则可以使用以下命令：

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --namespace AWS/S3
--start-time 2016-10-19T00:00:00Z --end-time 2016-10-20T00:00:00Z --statistics Average
--unit Bytes --region us-west-2 --dimensions Name=BucketName,Value=ExampleBucket
Name=StorageType,Value=StandardStorage --period 86400 --output json
```

此示例将产生以下输出：

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:00:00Z",
            "Average": 1025328.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "BucketSizeBytes"
}
```

### 使用 CloudWatch 控制台查看指标

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics。
3. 选择 S3 命名空间。
4. (可选) 要查看某个指标，请在搜索字段中键入该指标的名称。
5. (可选) 要按照 StorageType 维度进行筛选，请在搜索字段中键入存储类的名称。

### 使用 AWS CLI 查看存储的针对您 AWS 账户的有效指标列表

- 在命令提示符处，输入以下命令：

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

## 相关资源

- [Amazon CloudWatch Logs API Reference](#)
- [Amazon CloudWatch 用户指南](#)
- AWS CLI Command Reference 中的 [list-metrics](#) 操作。
- AWS CLI Command Reference 中的 [get-metric-statistics](#) 操作。
- [存储桶的指标配置 \(p. 487\)](#)。

## 存储桶的指标配置

利用 Amazon S3 的 CloudWatch 请求指标，您可以接收 1 分钟 CloudWatch 指标、设置 CloudWatch 警报和访问 CloudWatch 控制面板，以查看 Amazon S3 存储的近乎实时的运行状况和性能。对于依赖于云存储的应用程序而言，这些指标使您可以快速识别运行问题并采取措施。启用后，这些 1 分钟指标默认在 Amazon S3 存储桶级别提供。

如果您要获取某个存储桶中的对象的 CloudWatch 请求指标，则必须为该存储桶创建指标配置。您还可以为使用共享前缀或对象标签收集的指标定义筛选条件，这样您就可以使各个指标筛选条件满足特定业务应用程序、工作流或内部组织的需求。

有关可用 CloudWatch 指标的更多信息和存储指标与请求指标之间的区别，请参阅 [使用 Amazon CloudWatch 来监控指标 \(p. 482\)](#)。

在使用指标配置时，请记住以下几点：

- 每个存储桶最多可以配置 1000 个指标。
- 您可以利用筛选条件选择存储桶中的哪些对象可纳入指标配置。通过筛选共享前缀或对象标签，您可以使各个指标筛选条件满足特定业务应用程序、工作流或内部组织的需求。要请求整个存储桶的指标，请创建一种不带筛选条件的指标配置。
- 只有启用请求指标才需要指标配置。存储桶级每日存储指标始终保持开启状态，并且免费提供。目前不能为筛选过的对象子集获取日常存储指标。
- 每个指标配置均可启用全套[可用请求指标 \(p. 483\)](#)。只有存在适用于您的存储桶或筛选条件请求类型，才会报告针对特定操作的指标（例如 PostRequests）。
- 将报告对象级操作的请求指标，并且还将报告列出存储桶内容的操作，例如 [GET Bucket \(List Objects\)](#)、[GET Bucket Object Versions](#) 和 [列出分段上传](#)，但不报告存储桶的其他操作。

## 最大努力 CloudWatch 指标传输

系统将以最大努力传输 CloudWatch 指标。大多数具有请求指标的针对 Amazon S3 对象的请求会导致将数据点发送到 CloudWatch。

因此无法保证指标的完整性和及时性。可能返回特定请求的数据点，其时间戳晚于实际处理请求的时间。1 分钟的数据点在通过 CloudWatch 提供之前可能会延迟，或者根本不会提供该数据点。您可以通过 CloudWatch 请求指标近乎实时地了解有关存储桶流量性质方面的信息。这并不意味着会完整记录所有请求。

根据此功能的最大努力性质，在[账单和成本管理控制面板](#)提供的报告中可能有一个或多个访问请求不会出现在存储桶指标中。

## 筛选指标配置

当使用 CloudWatch 指标配置时，您可以选择将配置筛选到一个存储桶中的多组相关对象。您可以基于以下一个或多个因素在存储桶中筛选包含在指标配置中的对象：

- 对象键名称前缀 - 尽管 Amazon S3 数据模型是一种扁平结构，但您仍可以使用前缀推断层次结构。Amazon S3 控制台支持这些带有文件夹概念的前缀。如果您按前缀进行筛选，有相同前缀的对象将包含在指标配置中。
- 标签 - 您可以为对象添加标签和键值名称对。标签可让您轻松查找和整理对象。这些标签还可用作指标配置的筛选条件。

如果您指定了一项筛选条件，则仅在单个对象运行的请求可以满足筛选条件，并包含在报告的指标中。类似 [Delete Multiple Objects](#) 这样的请求和 List 请求不会为有筛选条件的配置返回指标。

要请求更复杂的筛选，请选择两个或更多元素。只有拥有所有这些元素的对象才会包含在指标配置中。如果未设置筛选条件，则存储桶中的所有对象都会包含在指标配置中。

## 如何添加指标配置

您可以通过 Amazon S3 控制台、使用 AWS CLI 或使用 Amazon S3 REST API 向存储桶添加指标配置。有关如何在 AWS 管理控制台中执行此操作的信息，请参阅[如何为 S3 存储桶配置请求指标？\(在 Amazon Simple Storage Service 控制台用户指南 中\)](#)。

### 使用 AWS CLI 添加指标配置

1. 安装并设置 AWS CLI。有关说明，请参阅[获取 AWS Command Line Interface 用户指南 中的使用 AWS 命令行界面进行设置](#)。
2. 打开终端。
3. 运行以下命令以添加指标配置：

```
aws s3api put-bucket-metrics-configuration --endpoint http://s3-us-west-2.amazonaws.com  
--bucket bucket-name --id metrics-config-id --metrics-configuration '{"Id":"'metrics-  
config-id'","Filter":{"Prefix":"prefix1"}}
```

4. 要验证配置是否已添加，请执行以下命令：

```
aws s3api get-bucket-metrics-configuration --endpoint http://s3-us-west-2.amazonaws.com  
--bucket bucket-name --id metrics-config-id
```

这将返回以下响应：

```
{  
    "MetricsConfiguration": {  
        "Filter": {  
            "Prefix": "prefix1"  
        },  
        "Id": "metrics-config-id"  
    }  
}
```

您还可以使用 Amazon S3 REST API 以编程方式添加指标配置。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的以下主题：

- [PUT Bucket Metric 配置](#)
- [GET Bucket Metric 配置](#)
- [List Bucket Metric 配置](#)
- [DELETE Bucket Metric 配置](#)

## 使用 AWS CloudTrail 记录 Amazon S3 API 调用

Amazon S3 与 AWS CloudTrail 集成，后者是在 Amazon S3 中记录用户、角色或 AWS 服务所执行操作的服务。CloudTrail 将以事件的形式捕获 Amazon S3 的 API 调用子集，包括来自 Amazon S3 控制台的调用和对 Amazon S3 API 的代码调用。如果您创建了跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 Amazon S3 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history (事件历史记录) 中查看最新事件。通过使用 CloudTrail 收集的信息，您可以确定向 Amazon S3 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，包括如何对其进行配置和启用，请参阅[AWS CloudTrail User Guide](#)

## CloudTrail 中的 Amazon S3 信息

在您创建 AWS 账户时，将针对该账户启用 CloudTrail。当 Amazon S3 中发生受支持的事件活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history (事件历史记录) 中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅[使用 CloudTrail 事件历史记录查看事件](#)。

要持续记录 AWS 账户中的事件（包括 Amazon S3 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。此跟踪在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取操作。有关更多信息，请参阅：

- [创建跟踪概述](#)
- [CloudTrail 支持的服务和集成](#)
- [为 CloudTrail 配置 Amazon SNS 通知](#)
- [接收多个区域中的 CloudTrail 日志文件](#)和[从多个账户中接收 CloudTrail 日志文件](#)。

每个事件或日志条目都包含有关生成请求的人员的信息。身份信息帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅[CloudTrail userIdentity 元素](#)。

日志文件可以在存储桶中存储任意长时间，不过您也可以定义 Amazon S3 生命周期规则以自动存档或删除日志文件。默认情况下，将使用 Amazon S3 服务器端加密 (SSE) 对日志文件进行加密。

## CloudTrail 日志记录跟踪的 Amazon S3 存储桶级操作

默认情况下，CloudTrail 将记录存储桶级操作。Amazon S3 记录与其他 AWS 服务记录一起写入日志文件。CloudTrail 基于时间段和文件大小来确定何时创建新文件并向其写入内容。

本节中的表格列出了可用于 CloudTrail 日志记录的 Amazon S3 存储桶级操作。

### CloudTrail 日志记录跟踪的 Amazon S3 存储桶级操作

REST API 名称	在 CloudTrail 日志中使用的 API 事件名称
<a href="#">DELETE Bucket</a>	DeleteBucket
<a href="#">DELETE Bucket cors</a>	DeleteBucketCors
<a href="#">DELETE Bucket 加密</a>	DeleteBucketEncryption
<a href="#">DELETE Bucket lifecycle</a>	DeleteBucketLifecycle
<a href="#">DELETE Bucket policy</a>	DeleteBucketPolicy
<a href="#">DELETE Bucket 复制</a>	DeleteBucketReplication
<a href="#">DELETE 存储桶标记</a>	DeleteBucketTagging
<a href="#">DELETE Bucket website</a>	DeleteBucketWebsite
<a href="#">GET Bucket acl</a>	GetBucketAcl

REST API 名称	在 CloudTrail 日志中使用的 API 事件名称
GET Bucket cors	GetBucketCors
GET Bucket 加密	GetBucketEncryption
GET Bucket lifecycle	GetBucketLifecycle
GET Bucket location	GetBucketLocation
GET Bucket logging	GetBucketLogging
GET Bucket notification	GetBucketNotification
GET Bucket policy	GetBucketPolicy
GET Bucket 复制	GetBucketReplication
GET Bucket requestPayment	GetBucketRequestPay
GET Bucket 标记	GetBucketTagging
GET Bucket versioning	GetBucketVersioning
GET Bucket website	GetBucketWebsite
GET 服务 (列出所有存储桶)	ListBuckets
PUT Bucket	CreateBucket
PUT Bucket acl	PutBucketAcl
PUT Bucket cors	PutBucketCors
PUT Bucket 加密	PutBucketEncryption
PUT Bucket lifecycle	PutBucketLifecycle
PUT Bucket logging	PutBucketLogging
PUT Bucket notification	PutBucketNotification
PUT 存储桶策略	PutBucketPolicy
PUT Bucket 复制	PutBucketReplication
PUT Bucket requestPayment	PutBucketRequestPay
PUT Bucket 标记	PutBucketTagging
PUT Bucket versioning	PutBucketVersioning
PUT Bucket website	PutBucketWebsite

除了这些 API 操作之外，也可以使用 [OPTIONS 对象](#) 对象级操作。由于此操作将检查存储桶的 CORS 配置，因此它被视为 CloudTrail 日志记录中的存储桶级操作。

## CloudTrail 日志记录跟踪的 Amazon S3 对象级操作

您也可以获取对象级 Amazon S3 操作的 CloudTrail 日志。要执行此操作，请指定跟踪的 Amazon S3 对象。当对象级操作在您的账户中发生时，CloudTrail 将评估您的跟踪设置。如果事件与您在跟踪中指定的对象相

匹配，则记录该事件。有关更多信息，请参阅[如何使用 AWS CloudTrail 数据事件为 S3 存储桶启用对象级别日志记录？](#)（在 Amazon Simple Storage Service 控制台用户指南中）以及 AWS CloudTrail User Guide 中的数据事件。下表列出了 CloudTrail 可以记录的对象级操作：

REST API 名称	在 CloudTrail 日志中使用的 API 事件名称
中止分段上传	AbortMultipartUpload
完成分段上传	CompleteMultipartUpload
DELETE Object	DeleteObject
GET Object	GetObject
GET Object ACL	GetObjectAcl
GET Object 标签	GetObjectTagging
GET Object torrent	GetObjectTorrent
HEAD Object	HeadObject
开始分段上传	CreateMultipartUpload
列出分段	ListParts
POST Object	PostObject
POST Object restore	RestoreObject
PUT Object	PutObject
PUT Object acl	PutObjectAcl
PUT Object 标签	PutObjectTagging
PUT Object – Copy	CopyObject
SELECT 对象内容	SelectObjectContent
上传分段	UploadPart
上传分段 – 复制	UploadPartCopy

除了这些操作之外，在特定情况下，您还可以使用以下存储桶级操作获取 CloudTrail 日志作为对象级 Amazon S3 操作：

- [GET Bucket \(List Objects\) Version 2](#) – 选择在跟踪中指定的前缀。
- [GET Bucket Object versions](#) – 选择在跟踪中指定的前缀。
- [HEAD Bucket](#) – 指定存储桶和空前缀。
- [Delete Multiple Objects](#) – 指定存储桶和空前缀。

## 跨账户情形中的对象级操作

下面是涉及跨账户情形中的对象级 API 调用的特殊使用案例以及报告 CloudTrail 日志的方法。CloudTrail 始终会将日志提供给请求者（进行 API 调用的人）。在设置跨账户访问时，请考虑本节中的示例。

### Note

这些示例假定 CloudTrail 日志已适当配置。

### 示例 1 : CloudTrail 将访问日志提供给存储桶拥有者

仅当存储桶拥有者具有对同一对象 API 的权限时，CloudTrail 才会将访问日志提供给存储桶拥有者。请考虑以下跨账户情形：

- 账户 A 拥有一个存储桶。
- 账户 B (请求者) 尝试访问该存储桶中的对象。

CloudTrail 始终将对象级 API 访问日志提供给请求者。此外，仅当存储桶拥有者具有对该对象执行同一 API 操作的权限时，CloudTrail 才会将相同的日志提供给存储桶拥有者。

#### Note

如果存储桶拥有者也是对象拥有者，存储桶拥有者将获得对象访问日志。否则，存储桶拥有者必须通过对象 ACL 获取权限，以便让相同的对象 API 获取相同的对象访问 API 日志。

### 示例 2 : CloudTrail 没有使设置对象 ACL 时使用的电子邮件地址激增

请考虑以下跨账户情形：

- 账户 A 拥有一个存储桶。
- 账户 B (请求者) 使用电子邮件地址发送了一个设置对象 ACL 授权的请求。有关 ACL 的信息，请参阅[访问控制列表 \(ACL\) 概述 \(p. 333\)](#)。

该请求获取日志以及电子邮件信息。但是，存储桶拥有者 (如果他们像在示例 1 中一样有资格接收日志) 将获取报告该事件的 CloudTrail 日志。但是，存储桶拥有者不会获取 ACL 配置信息，尤其是被授权者电子邮件和授权。日志为存储桶拥有者提供的唯一信息是账户 B 进行了 ACL API 调用。

## CloudTrail 对 Amazon S3 SOAP API 调用的跟踪

CloudTrail 跟踪 Amazon S3 SOAP API 调用。Amazon S3HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。有关 Amazon S3 SOAP 支持的更多信息，请参阅[附录 A : 使用 SOAP API \(p. 527\)](#)。

#### Important

SOAP 不支持较新的 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

CloudTrail 日志记录跟踪的 Amazon S3 SOAP 操作

SOAP API 名称	在 CloudTrail 日志中使用的 API 事件名称
ListAllMyBuckets	ListBuckets
CreateBucket	CreateBucket
DeleteBucket	DeleteBucket
GetBucketAccessControlPolicy	GetBucketAcl
SetBucketAccessControlPolicy	PutBucketAcl
GetBucketLoggingStatus	GetBucketLogging
SetBucketLoggingStatus	PutBucketLogging

# 使用 CloudTrail 日志和 Amazon S3 服务器访问日志及 CloudWatch Logs

您可以将 AWS CloudTrail 日志与 Amazon S3 服务器访问日志搭配使用。CloudTrail 日志为您提供了对 Amazon S3 存储桶级和对象级操作的详细 API 跟踪，而 Amazon S3 服务器访问日志则可让您了解关于您存储在 Amazon S3 中的数据的对象级操作。有关服务器访问日志的更多信息，请参阅 [Amazon S3 服务器访问日志记录 \(p. 506\)](#)。

此外，您还可以将 CloudTrail 日志与用于 Amazon S3 的 CloudWatch 搭配使用。CloudTrail 与 CloudWatch 日志集成在一起，将 CloudTrail 捕获的 S3 存储桶级 API 活动传输到您指定的 CloudWatch 日志组中的 CloudWatch 日志流。您可以创建用于监控特定 API 活动的 CloudWatch 警报，并在此特定 API 活动发生时收到电子邮件通知。有关用于监控特定 API 活动的 CloudWatch 警报的更多信息，请参阅 [AWS CloudTrail User Guide](#)。有关将 CloudWatch 与 Amazon S3 配合使用的更多信息，请参阅[使用 Amazon CloudWatch 来监控指标 \(p. 482\)](#)。

## 示例：Amazon S3 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一条 CloudTrail 日志条目，它演示了 [DELETE 存储桶策略](#)、[PUT 存储桶 acl](#) 和 [GET 存储桶版本控制](#) 操作。

```
{  
    "Records": [  
        {  
            "eventVersion": "1.03",  
            "userIdentity": {  
                "type": "IAMUser",  
                "principalId": "111122223333",  
                "arn": "arn:aws:iam::111122223333:user/myUserName",  
                "accountId": "111122223333",  
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",  
                "userName": "myUserName"  
            },  
            "eventTime": "2015-08-26T20:46:31Z",  
            "eventSource": "s3.amazonaws.com",  
            "eventName": "DeleteBucketPolicy",  
            "awsRegion": "us-west-2",  
            "sourceIPAddress": "127.0.0.1",  
            "userAgent": "[ ]",  
            "requestParameters": {  
                "bucketName": "myawsbucket"  
            },  
            "responseElements": null,  
            "requestID": "47B8E8D397DCE7A6",  
            "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",  
            "eventType": "AwsApiCall",  
            "recipientAccountId": "111122223333"  
        },  
        {  
            "eventVersion": "1.03",  
            "userIdentity": {  
                "type": "IAMUser",  
                "principalId": "111122223333",  
                "arn": "arn:aws:iam::111122223333:user/myUserName",  
                "accountId": "111122223333",  
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",  
                "userName": "myUserName"  
            },  
            "eventTime": "2015-08-26T20:46:31Z",  
            "eventSource": "s3.amazonaws.com",  
            "eventName": "PutBucketVersioning",  
            "awsRegion": "us-west-2",  
            "sourceIPAddress": "127.0.0.1",  
            "userAgent": "[ ]",  
            "requestParameters": {  
                "versioningConfiguration": {  
                    "status": "Enabled",  
                    "sse": "AES256",  
                    "sseKmsMasterKeyArn": "arn:aws:kms:us-west-2:111122223333:key/12345678-1234-1234-1234-123456789012"  
                }  
            },  
            "responseElements": null,  
            "requestID": "47B8E8D397DCE7A6",  
            "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",  
            "eventType": "AwsApiCall",  
            "recipientAccountId": "111122223333"  
        },  
        {  
            "eventVersion": "1.03",  
            "userIdentity": {  
                "type": "IAMUser",  
                "principalId": "111122223333",  
                "arn": "arn:aws:iam::111122223333:user/myUserName",  
                "accountId": "111122223333",  
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",  
                "userName": "myUserName"  
            },  
            "eventTime": "2015-08-26T20:46:31Z",  
            "eventSource": "s3.amazonaws.com",  
            "eventName": "GetBucketVersioning",  
            "awsRegion": "us-west-2",  
            "sourceIPAddress": "127.0.0.1",  
            "userAgent": "[ ]",  
            "requestParameters": {  
                "versioningConfiguration": {}  
            },  
            "responseElements": {  
                "versioningConfiguration": {  
                    "status": "Enabled",  
                    "sse": "AES256",  
                    "sseKmsMasterKeyArn": "arn:aws:kms:us-west-2:111122223333:key/12345678-1234-1234-1234-123456789012"  
                }  
            },  
            "requestID": "47B8E8D397DCE7A6",  
            "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",  
            "eventType": "AwsApiCall",  
            "recipientAccountId": "111122223333"  
        }  
    ]  
}
```

```
        "userName": "myUserName"
    },
    "eventTime": "2015-08-26T20:46:31Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "PutBucketAcl",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "",
    "userAgent": "[ ]",
    "requestParameters": {
        "bucketName": "",
        "AccessControlPolicy": {
            "AccessControlList": {
                "Grant": {
                    "Grantee": {
                        "xsi:type": "CanonicalUser",
                        "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
                        "ID": "d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
                    },
                    "Permission": "FULL_CONTROL"
                }
            },
            "xmlns": "http://s3.amazonaws.com/doc/2006-03-01/",
            "Owner": {
                "ID": "d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
            }
        }
    },
    "responseElements": null,
    "requestID": "BD8798EACDD16751",
    "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
},
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/myUserName",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "myUserName"
    },
    "eventTime": "2015-08-26T20:46:31Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "GetBucketVersioning",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "",
    "userAgent": "[ ]",
    "requestParameters": {
        "bucketName": "myawsbucket"
    },
    "responseElements": null,
    "requestID": "07D681279BD94AED",
    "eventID": "f2b287f3-0df1-4961-a2f4-c4bdfed47657",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
]
```

## 相关资源

- [AWS CloudTrail User Guide](#)
- [CloudTrail 事件参考](#)

# 将 BitTorrent 与 Amazon S3 配合使用

## 主题

- [BitTorrent 传输的收费方式 \(p. 496\)](#)
- [使用 BitTorrent 来检索存储在 Amazon S3 中的对象 \(p. 496\)](#)
- [使用 Amazon S3 和 BitTorrent 发布内容 \(p. 497\)](#)

BitTorrent 是开放对等的文件分发协议。您可以使用 BitTorrent 协议检索 Amazon S3 中任何可公开访问的对象。本节描述了您可能会使用 BitTorrent 将您的数据分发至 Amazon S3 之外的原因以及应如何执行此操作。

Amazon S3 支持 BitTorrent 协议，因此开发人员可以在分发大规模内容时节约成本。Amazon S3 可用于简单可靠地存储任何数据。用于 Amazon S3 数据的默认分发机制是通过客户端/服务器下载。在客户端/服务器分发中，整个对象将以点对点的方式从 Amazon S3 传输到所有请求该对象的授权用户。尽管客户端/服务器传输适用于各种使用案例，但它并非对每个人都是最优的。尤其是，随着下载对象的用户数的增加，客户端/服务器分发的成本将直线上升。这使得分发受欢迎的对象的成本非常昂贵。

BitTorrent 通过将正在下载对象的客户端用作分发服务器来解决此问题：每个客户端从 Amazon S3 和其他客户端下载对象的某些部分，同时将相同部分的对象上传到其他关注的“对等方”。发布者可以获得的好处是：对于受欢迎的大型文件，Amazon S3 实际提供的数据量要显著少于通过客户端/服务器下载时需要为同一客户端提供的数量。传输的数据越少，对对象的发布者而言就意味着成本更低。

## Note

您仅可以为大小小于 5 GB 的对象获取 torrent 文件。

## BitTorrent 传输的收费方式

在 Amazon S3 中使用 BitTorrent 不会产生额外的费用。通过 BitTorrent 协议传输数据的计费费率与客户端/服务器传输相同。确切地讲，每次下载 BitTorrent 时，客户端都会从 Amazon S3 “seeder”请求“piece”对象，如同使用了 REST 或 SOAP 协议为该部分创建了匿名请求一样，费用将逐渐增加。这些费用将以相同的方式出现在您的 Amazon S3 账单和使用报告上。区别在于，如果大量客户端同时通过 BitTorrent 请求相同的对象，则 Amazon S3 为满足这些客户端需求而必须提供的数据量要低于客户端/服务器传输。这是因为 BitTorrent 客户端可在它们中间同时进行上传和下载。

## Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

通过使用 BitTorrent 而实现的数据传输节省可能会有很大的差异，具体取决于您的对象受欢迎程度。不受欢迎的对象需要更多地使用“seeder”来为客户端提供服务，因此对于这些对象而言，BitTorrent 分发成本和客户端/服务器分发成本之间的差异很小。特别是，如果只有一个客户端并且该客户端一次仅下载一个特殊的对象，那么 BitTorrent 传输的成本与直接下载的成本相同。

## 使用 BitTorrent 来检索存储在 Amazon S3 中的对象

Amazon S3 中任何可以匿名读取的对象也可以通过 BitTorrent 进行下载。需要使用 BitTorrent 客户端应用程序才能执行此操作。Amazon 不会分发 BitTorrent 客户端应用程序，但是会提供许多免费的客户

端。已对 Amazon S3BitTorrent 执行进行了测试，可以与正式 BitTorrent 客户端结合使用 (请访问 <http://www.bittorrent.com/>)。

BitTorrent 下载将从 .torrent 文件开始。这个小型文件向 BitTorrent 客户端描述了要下载的数据和开始查找该数据的位置。.torrent 文件是要下载的实际对象大小的一小部分。一旦您为 BitTorrent 客户端应用程序中添加了 Amazon S3 生成的 .torrent 文件，它将立即从 Amazon S3 and 从任意的“peer”BitTorrent 客户端开始下载。

可以轻松地为任何公开可用的对象检索 .torrent 文件。只需将“?torrent”查询字符串参数添加到对象的 REST GET 请求的末尾即可。无需进行身份验证。安装 BitTorrent 客户端后，使用 BitTorrent 下载来下载对象可能和您在 Web 浏览器中打开此 URL 一样简单。

不存在使用 SOAP API 为 Amazon S3 对象取回 .torrent 的机制。

#### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

#### Example

本示例将为“quotes”存储桶中的“Nelson”对象检索 Torrent 文件。

#### Sample Request

```
GET /quotes/Nelson?torrent HTTP/1.0
Date: Wed, 25 Nov 2009 12:00:00 GMT
```

#### Sample Response

```
HTTP/1.1 200 OK
x-amz-request-id: 7CD745EBB7AB5ED9
Date: Wed, 25 Nov 2009 12:00:00 GMT
Content-Disposition: attachment; filename=Nelson.torrent;
Content-Type: application/x-bittorrent
Content-Length: 537
Server: AmazonS3

<body: a Bencoded dictionary as defined by the BitTorrent specification>
```

## 使用 Amazon S3 和 BitTorrent 发布内容

使用 BitTorrent 时，存储在 Amazon S3 中的所有匿名可读的对象将自动可供下载。将对象上的 ACL 更改为允许匿名 READ 操作的过程在 [管理对 Amazon S3 资源的访问权限 \(p. 242\)](#) 中进行了描述。

您可以将客户端定向至您的 BitTorrent 可访问对象，方法是直接向它们提供 .torrent 文件或发布一个指向对象的 ?torrent URL 的链接。需要特别注意的是，第一次请求时 (通过 REST ?torrent 资源)，描述 Amazon S3 对象的 .torrent 文件是按需生成的。为对象生成 .torrent 所需的时间与对象的大小成正比。对于大型对象，此时间可能会很长。因此，在发布 ?torrent 链接之前，建议为其本身创建首个请求。在生成 .torrent 文件时，Amazon S3 可能需要几分钟的时间才能响应此首个请求。除非您更新了所涉及的对象，否则 .torrent 的后续请求将会很快。在分发 ?torrent 链接之前遵循此步骤，可确保客户拥有流畅的 BitTorrent 下载体验。

要停止使用 BitTorrent 分发文件，只需移除对文件的匿名访问权限即可。可以通过从 Amazon S3 删除该文件，或将您的访问控制策略修改为禁止匿名读取来完成此操作。完成此操作后，Amazon S3 不再是文件的 BitTorrent 网络中的“seeder”，并且也不会再通过 ?torrent REST API 提供 .torrent 文件。但是在为您的文件发布了 .torrent 后，此操作可能不会停止对象的公共下载 (仅在使用 BitTorrent 对等网络时发生)。

# 处理 REST 和 SOAP 错误

## 主题

- [REST 错误响应 \(p. 498\)](#)
- [SOAP 错误响应 \(p. 499\)](#)
- [Amazon S3 排错最佳实践 \(p. 500\)](#)

本节描述 REST 和 SOAP 错误，以及如何处理它们。

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

## REST 错误响应

### 主题

- [响应标头 \(p. 498\)](#)
- [错误响应 \(p. 499\)](#)

如果 REST 请求导致错误，则 HTTP 回复将包含：

- 作为响应正文的 XML 错误文档
- Content-Type：application/xml
- 合适的 3xx、4xx 或 5xx HTTP 状态代码

下面是 REST 错误响应的示例。

```
<?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>
```

有关 Amazon S3 错误的更多信息，请参阅 [ErrorCodeList](#)。

## 响应标头

下面是所有操作返回的响应标头：

- **x-amz-request-id**: 系统分配给每个请求的唯一 ID。在极少的情况下，如果您在使用 Amazon S3 时遇到问题，Amazon 将使用此信息来帮助排查问题。
- **x-amz-id-2**: 将帮助我们排查问题的特殊令牌。

## 错误响应

### 主题

- [错误代码 \(p. 499\)](#)
- [错误消息 \(p. 499\)](#)
- [更多详细信息 \(p. 499\)](#)

若 Amazon S3 请求出现错误，客户端将收到一个错误响应。准确的错误响应格式应该特定于 API：例如，REST 错误响应的格式和 SOAP 错误响应的格式不同。但是，所有错误响应都有一些共同的元素。

#### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

## 错误代码

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。许多错误代码在 SOAP 和 REST API 上都比较常见，但也有部分是特定于 API 的。例如，NoSuchKey 是通用的，但是 UnexpectedContent 仅在响应无效的 REST 请求时发生。在所有案例中，SOAP 错误代码会带有一个前缀（如错误代码表格中所示），以便 NoSuchKey 错误实际将作为 Client.NoSuchKey 返回到 SOAP 中。

#### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

## 错误消息

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。

## 更多详细信息

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。

## SOAP 错误响应

#### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。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。

#### Example

```
<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>
```

有关 Amazon S3 错误的更多信息，请参阅 [ErrorCodeList](#)。

## Amazon S3 排错最佳实践

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，并作出相应的反应。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，请将这些错误发布到我们的 Amazon S3 开发人员论坛 <https://forums.aws.amazon.com/> 或注册 AWS Premium Support <https://aws.amazon.com/premiumsupport/>。

### 隔离错误

#### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。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，您还应该参考供用户阅读的<详细信息> XML 错误响应的元素。

# Amazon S3 疑难解答

本节介绍了如何排除 Amazon S3 的故障，并说明了在联系 AWS Support 时如何获取您需要的请求 ID。

## 主题

- [根据征兆对 Amazon S3 进行故障排除 \(p. 502\)](#)
- [获取 AWS Support 需要的 Amazon S3 请求 ID \(p. 502\)](#)
- [相关主题 \(p. 504\)](#)

## 根据征兆对 Amazon S3 进行故障排除

以下主题列出了各种征兆，帮助您解决在使用 Amazon S3 时可能遇到的某些问题。

### 征兆

- [启用版本控制后，Amazon S3 对存储桶请求的 HTTP 503 响应显著增加 \(p. 502\)](#)
- [访问具有 CORS 设置的存储桶时出现意外行为 \(p. 502\)](#)

## 启用版本控制后，Amazon S3 对存储桶请求的 HTTP 503 响应显著增加

如果您注意到启用版本控制后，Amazon S3 对存储桶的 PUT 或 DELETE 对象请求的 HTTP 503 慢速响应数量显著增加，那么存储桶中可能有一个或多个对象有数以百万计的版本。如果您的对象有数以百万计的版本，Amazon S3 会自动限制对该存储桶的请求，以防止客户的请求流量过多，但也可能会妨碍对该存储桶的其他请求。

要确定哪些 S3 对象有数以百万计的版本，可以使用 Amazon S3 清单工具。清单工具可以生成一份报告，提供存储桶中对象的平面文件列表。有关更多信息，请参阅 [Amazon S3 清单 \(p. 235\)](#)。

Amazon S3 团队鼓励客户调查重复覆盖同一 S3 对象的应用程序（可能会为该对象创建数百万个版本），确定应用程序是否正常工作。如果您在使用中发现一个或多个 S3 对象需要数百万个版本，请通过 [AWS Support](#) 与 AWS Support 团队联系，讨论您的使用案例，并帮助我们协助您确定最佳解决方案。

## 访问具有 CORS 设置的存储桶时出现意外行为

如果您在访问具有跨源资源共享 (CORS) 配置的存储桶时遇到意外行为，请参阅 [排查 CORS 问题 \(p. 141\)](#)。

## 获取 AWS Support 需要的 Amazon S3 请求 ID

当您因在 Amazon S3 中遇到错误或意外行为而需要联系 AWS Support 时，您将需要获取与失败操作关联的请求 ID。获取这些请求 ID 将使 AWS Support 能够帮助您解决所遇到的问题。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）中返回并通过详细日志访问。可通过大量常见方法来获取您的请求 ID。

在您恢复这些日志后，复制并保留这两个值，因为您在联系 AWS Support 时需要它们。有关联系 AWS Support 的信息，请参阅[联系我们](#)。

## 主题

- 使用 HTTP 获得请求 ID (p. 503)
- 使用 Web 浏览器获得请求 ID (p. 503)
- 使用 AWS SDK 获得请求 ID (p. 503)
- 使用 AWS CLI 获得请求 ID (p. 504)

## 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并运行该工具，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 对将与以下示例类似。

```
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 浏览器都具有允许您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，您需要使用开发人员工具来查看 HTTP 响应标头。有关适用于特定浏览器的开发人员工具的信息，请参阅 AWS 开发人员论坛中的 Amazon S3 疑难解答 - 如何恢复 S3 请求 ID。

## 使用 AWS SDK 获得请求 ID

以下部分包含有关使用 AWS 开发工具包配置日志记录的信息。当您可以对每个请求和响应启用详细日志记录时，您不应在生产系统中启用日志记录，因为大量请求/响应会导致应用程序明显变慢。

对于 AWS 开发工具包请求，请求 ID 对将与以下示例类似。

```
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
```

## 使用适用于 PHP 的开发工具包 获得请求 ID

可使用 PHP 配置日志记录。有关更多信息，请参阅[如何查看已通过线路发送的数据？](#)，该文章位于适用于 PHP 的 AWS 开发工具包 的常见问题中。

## 使用适用于 Java 的开发工具包 获得请求 ID

可以为特定请求或响应启用日志记录，这将允许您仅捕获和返回相关标头。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，然后再执行实际请求。调用 `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()` 获取记录的请求或响应。

#### Example

```
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 AWS SDK for Java Developer Guide 中的“适用于 Java 的 AWS 开发工具包调用的日志记录”主题中的[详细线路日志记录](#)。

## 使用适用于 .NET 的 AWS 开发工具包 获得请求 ID

您可以使用内置 `System.Diagnostics` 日志记录工具配置 适用于 .NET 的 AWS 开发工具包 中的日志记录。有关更多信息，请参阅[使用适用于 .NET 的 AWS 开发工具包进行日志记录](#) AWS 开发人员博客文章。

#### Note

默认情况下，返回的日志仅包含错误信息。配置文件需要已添加 `AWSLogMetrics` (可以选择添加 `AWSResponseLogging`) 来获取请求 ID。

## 使用 SDK for Python 获得请求 ID

可以通过将以下行添加到代码中以将调试信息输出到文件中，来配置 Python 中的日志记录。

```
import logging
logging.basicConfig(filename="mylog.log", level=logging.DEBUG)
```

如果使用的是适用于 AWS 的 Boto Python 接口，则可按照[此处](#)的 Boto 文档将调试级别设置为 2。

## 使用适用于 Ruby 的开发工具包 获得请求 ID

您可以使用版本 1、版本 2 或版本 3 的 适用于 Ruby 的开发工具包 来获取请求 ID。

- 使用 适用于 Ruby 的开发工具包 - 版本 1 – 可以使用以下代码行来全局启用 HTTP 线路日志记录。

```
s3 = AWS::S3.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

- 使用 适用于 Ruby 的开发工具包 的版本 2 或版本 3 – 可以使用以下代码行来全局启用 HTTP 线路日志记录。

```
s3 = Aws::S3::Client.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

## 使用 AWS CLI 获得请求 ID

可以通过将 `--debug` 添加到命令中来获取 AWS CLI 中的请求 ID。

## 相关主题

有关其他疑难解答和支持主题，请参阅以下内容：

- [排查 CORS 问题 \(p. 141\)](#)
- [处理 REST 和 SOAP 错误 \(p. 498\)](#)
- [AWS Support 文档](#)

有关第三方工具的疑难解答信息，请参阅 AWS 开发人员论坛中的[获取 Amazon S3 请求 ID](#)。

# Amazon S3 服务器访问日志记录

服务器访问日志记录详细地记录对存储桶提出的各种请求。对于许多应用程序而言，服务器访问日志很有用。例如，访问日志信息可能在安全和访问权限审核方面很有用。它还可以帮助您了解您的客户群并了解您的 Amazon S3 账单。

## 主题

- [如何启用服务器访问日志记录 \(p. 506\)](#)
- [日志对象键格式 \(p. 507\)](#)
- [如何传输日志？\(p. 507\)](#)
- [最大努力服务器日志传输 \(p. 507\)](#)
- [存储桶日志记录状态更改将逐渐生效 \(p. 508\)](#)
- [使用控制台启用日志记录 \(p. 508\)](#)
- [以编程方式启用日志记录 \(p. 508\)](#)
- [服务器访问日志格式 \(p. 511\)](#)
- [删除 Amazon S3 日志文件 \(p. 517\)](#)

## 如何启用服务器访问日志记录

要跟踪针对您的存储桶的访问请求，您可以启用服务器访问日志记录。每个访问日志记录都提供有关单个访问请求的详细信息，如请求者、存储桶名称、请求时间、请求操作、响应状态和错误代码（如果相关）。

### Note

对 Amazon S3 存储桶启用日志记录无需额外费用；但是，系统传输给您的任何日志文件都会产生普通存储费用。（您可以随时删除日志文件。）日志文件传输不会产生数据传输费，但对已传输的日志文件的访问则与任何其他数据传输同等计费。

默认情况下，日志记录处于禁用状态。启用了日志记录时，日志会保存到与源存储桶相同的 AWS 区域的存储桶中。

要启用访问日志记录，您必须执行以下操作：

- 在希望 Amazon S3 传输访问日志的存储桶上添加日志记录配置，打开日志传输。我们将此存储桶称为源存储桶。
- 向 Amazon S3 日志传输组授予对用于保存访问日志的存储桶的写入权限。我们将此存储桶称为目标存储桶。

要打开日志传输，请提供以下日志记录配置信息：

- 您希望 Amazon S3 在其中将访问日志保存为对象的目标存储桶的名称。您可以让日志传输至您拥有的且与源存储桶位于同一区域中的任何存储桶，包括源存储桶本身。

我们建议您将访问日志保存在不同的存储桶中，以方便管理这些日志。如果您选择将访问日志保存在源存储桶中，我们建议您为所有日志对象键指定前缀，以便对象名称以通用字符串开头，且日志对象更易于识别。

当源存储桶和目标存储桶是同一存储桶时，将为写入该存储桶的日志创建额外的日志。此行为对于您的使用案例而言可能并不理想，因为它会导致您的存储账单金额小幅增加。此外，有关日志的额外日志可能会导致更难以找到您所查找的日志。

#### Note

源存储桶和目标存储桶必须由同一个 AWS 账户拥有，且存储桶必须都位于同一区域。

- (可选) 供 Amazon S3 分配给所有日志对象键的前缀。通过该前缀可更方便地查找日志对象。

例如，如果您指定前缀值 logs/，则 Amazon S3 创建的每个日志对象的键都以 logs/ 前缀开头，如以下示例所示：

```
logs/2013-11-01-21-32-16-E568B2907131C0C0
```

当您删除日志时，键前缀可以有所帮助。例如，您可以设置一个生命周期配置规则，让 Amazon S3 删除具有特定键前缀的对象。有关更多信息，请参阅 [删除 Amazon S3 日志文件 \(p. 517\)](#)。

- (可选) 使其他人可以访问生成的日志的权限。默认情况下，存储桶拥有者始终拥有日志对象的完全访问权限。您可以选择向其他用户授予访问权限。

有关启用服务器访问日志记录的更多信息，请参阅 [使用控制台启用日志记录 \(p. 508\)](#) 和 [以编程方式启用日志记录 \(p. 508\)](#)。

## 日志对象键格式

Amazon S3 将以下对象键格式用于在目标存储桶中上传的日志对象：

```
TargetPrefixYYYY-mm-DD-HH-MM-SS-UniqueString
```

在键中，YYYY、mm、DD、HH、MM 和 SS 分别为日志文件传输时间中表示年、月、日、小时、分钟和秒的数字。

在特定时间传输的日志文件可包含在该时间前的任何时刻编写的记录。无法知道是否已传输特定时间间隔内的所有日志记录。

键的 UniqueString 部分用于防止覆盖文件。它没有意义，日志处理软件应忽略它。

## 如何传输日志？

Amazon S3 定期收集访问日志记录，在日志文件中整合这些记录，然后将日志文件作为日志对象上传到目标存储桶。如果多个启用了日志记录的源存储桶使用相同的目标存储桶，此目标存储桶中将保留所有这些源存储桶的访问日志。但是，每个日志对象只会报告特定源存储桶的访问日志记录。

Amazon S3 使用特殊的日志传输账户（称为日志传输组）写入访问日志。这些写入受常规的访问控制限制。您必须通过在目标存储桶的访问控制列表（ACL）中添加授权条目，向日志传输组授予对该存储桶的写入权限。如果您使用 Amazon S3 控制台在存储桶上启用日志记录，则该控制台会同时在源存储桶上启用日志记录，并在目标存储桶上更新 ACL，以便向日志传输组授予写入权限。

## 最大努力服务器日志传输

服务器访问日志记录会以最大努力进行传输。针对已正确配置了日志记录的存储桶的大多数请求会导致传输一条日志记录。大多数日志记录将在记录后的几小时内传输，但可以更频繁地传输这些记录。

因此不能保证服务器日志记录的完整性和即时性。特殊请求的日志记录可能会在实际处理了请求之后进行传输，也可能完全不会传输。服务器日志的用途在于向您提供有关存储桶流量性质方面的信息。丢失日志记录的情况十分少见，但是服务器日志记录不旨在完整记录所有请求。

根据服务器日志记录功能的最大努力性质，在 AWS 门户上提供的使用率报告 (AWS 管理控制台上的账单和成本管理报告) 中可能有一个或多个访问请求不会出现在传输的服务器日志中。

## 存储桶日志记录状态更改将逐渐生效

存储桶日志记录状态的更改需要一定时间才能实际影响日志文件的传输。例如，如果您为某个存储桶启用了日志记录，那么将记录在以下时间内发送的请求，而不会记录其他请求。如果您将日志记录的目标存储桶从存储桶 A 更改为存储桶 B，则在接下来的一个小时里仍可能有一些日志传输到存储桶 A，但其他日志则会传输到新的目标存储桶 B。无论如何，新的设置将最终生效，并且您无需执行任何操作。

## 使用控制台启用日志记录

有关在 [AWS 管理控制台](#) 中启用 Amazon S3 服务器访问日志记录 (p. 506) 的信息，请参阅[如何为 S3 存储桶启用服务器访问日志记录？](#)(在 Amazon Simple Storage Service 控制台用户指南 中)。

当您在存储桶上启用日志记录时，控制台会同时在源存储桶上启用日志记录，并在目标存储桶的访问控制列表 (ACL) 中添加授权，向日志传输组授予写入权限。

有关以编程方式启用日志记录的信息，请参阅[以编程方式启用日志记录 \(p. 508\)](#)。

有关日志记录格式的信息(包括字段列表及其描述)，请参阅[服务器访问日志格式 \(p. 511\)](#)。

## 以编程方式启用日志记录

您可以使用 Amazon S3 API 或 AWS 开发工具包，以编程方式启用或禁用日志记录。为此，请同时在存储桶上启用日志记录并向日志传输组授予向目标存储桶写入日志的权限。

### 主题

- [启用日志记录 \(p. 508\)](#)
- [向日志传输组授予 WRITE 和 READ\\_ACP 权限 \(p. 509\)](#)
- [示例：适用于 .NET 的 AWS 开发工具包 \(p. 509\)](#)
- [更多信息 \(p. 511\)](#)

## 启用日志记录

要启用日志记录，请提交 [PUT Bucket 日志记录](#) 请求，以在源存储桶上添加日志记录配置。该请求指定目标存储桶，以及要用于所有日志对象键的前缀(可选)。以下示例将 logbucket 标识为目标存储桶，将 logs/ 标识为前缀。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>logbucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

日志对象由日志传输账户编写和拥有，存储桶拥有者可获得对日志对象的完全权限。此外，您还可以选择向其他用户授予权限，以便他们可以访问日志。有关更多信息，请参阅[PUT Bucket 日志记录](#)。

Amazon S3 还提供 [GET Bucket 日志记录](#) API，用于检索存储桶的日志记录配置。要删除日志记录配置，请发送 BucketLoggingStatus 为空的 PUT Bucket 日志记录请求。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
</BucketLoggingStatus>
```

您可以使用 Amazon S3 API 或 AWS 开发工具包包装程序库在存储桶上启用日志记录。

## 向日志传输组授予 WRITE 和 READ\_ACP 权限

Amazon S3 作为预定义 Amazon S3 日志传输组的成员，向目标存储桶写入日志文件。这些写入受常规的访问控制限制。您必须通过向目标存储桶的访问控制列表 (ACL) 添加授权，向此组授予 s3:GetObjectAcl 和 s3:PutObject 权限。日志传输组由以下 URL 表示。

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

若要授予 WRITE 和 READ\_ACP 权限，请添加以下授权。有关 ACL 的信息，请参阅[使用 ACL 管理访问 \(p. 333\)](#)。

```
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
  </Grantee>
  <Permission>WRITE</Permission>
</Grant>
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
  </Grantee>
  <Permission>READ_ACP</Permission>
</Grant>
```

有关使用 AWS 开发工具包以编程方式添加 ACL 授权的示例，请参阅[使用 AWS SDK for Java 管理 ACL \(p. 339\)](#) 和[使用适用于 .NET 的 AWS 开发工具包管理 ACL \(p. 341\)](#)。

## 示例：适用于 .NET 的 AWS 开发工具包

以下 C# 示例在存储桶上启用日志记录。您需要创建两个存储桶（一个源存储桶和一个目标存储桶）。该示例首先向日志传输组授予向目标存储桶写入日志的所需权限，然后在源存储桶上启用日志记录。有关更多信息，请参阅[以编程方式启用日志记录 \(p. 508\)](#)。有关如何创建和测试有效示例的说明，请参阅[运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)。

### Example

```
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-s3-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

namespace Amazon.DocSamples.S3
{
    class ServerAccessLoggingTest
    {
        private const string bucketName = "*** bucket name for which to enable logging ***";
        private const string targetBucketName = "*** bucket name where you want access logs stored ***";
```

```
private const string logObjectKeyPrefix = "Logs";
// Specify your bucket region (an example region is shown).
private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
private static IAmazonS3 client;

public static void Main()
{
    client = new AmazonS3Client(bucketRegion);
    EnableLoggingAsync().Wait();
}

private static async Task EnableLoggingAsync()
{
    try
    {
        // Step 1 - Grant Log Delivery group permission to write log to the target
        await GrantPermissionsToWriteLogsAsync();
        // Step 2 - Enable logging on the source bucket.
        await EnableDisableLoggingAsync();
    }
    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);
    }
}

private static async Task GrantPermissionsToWriteLogsAsync()
{
    var bucketACL = new S3AccessControlList();
    var aclResponse = client.GetACL(new GetACLRequest { BucketName =
targetBucketName });
    bucketACL = aclResponse.AccessControlList;
    bucketACL.AddGrant(new S3Grantee { URI = "http://acs.amazonaws.com/groups/s3/
LogDelivery" }, S3Permission.WRITE);
    bucketACL.AddGrant(new S3Grantee { URI = "http://acs.amazonaws.com/groups/s3/
LogDelivery" }, S3Permission.READ_ACP);
    var setACLRequest = new PutACLRequest
    {
        AccessControlList = bucketACL,
        BucketName = targetBucketName
    };
    await client.PutACLAsync(setACLRequest);
}

private static async Task EnableDisableLoggingAsync()
{
    var loggingConfig = new S3BucketLoggingConfig
    {
        TargetBucketName = targetBucketName,
        TargetPrefix = logObjectKeyPrefix
    };

    // Send request.
    var putBucketLoggingRequest = new PutBucketLoggingRequest
    {
        BucketName = bucketName,
        LoggingConfig = loggingConfig
    };
    await client.PutBucketLoggingAsync(putBucketLoggingRequest);
```

```
}
```

## 更多信息

- Amazon S3 服务器访问日志记录 (p. 506)
- AWS CloudFormation 用户指南中的 [AWS::S3::Bucket](#)

## 服务器访问日志格式

服务器访问日志文件由一系列的换行分隔日志记录组成。每个日志记录表示一个请求并由空格分隔的字段组成。以下是含有六份日志记录的示例日志。

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be mybucket [06/  
Feb/2014:00:00:38 +0000] 192.0.2.3  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE  
REST.GET.VERSIONING - "GET /mybucket?versioning HTTP/1.1" 200 - 113 - 7 - "-"  
"S3Console/0.4" -  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be mybucket [06/  
Feb/2014:00:00:38 +0000] 192.0.2.3  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE  
REST.GET.LOGGING_STATUS - "GET /mybucket?logging HTTP/1.1" 200 - 242 - 11 - "-"  
"S3Console/0.4" -  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be mybucket [06/  
Feb/2014:00:00:38 +0000] 192.0.2.3  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE  
REST.GET.BUCKETPOLICY - "GET /mybucket?policy HTTP/1.1" 404 NoSuchBucketPolicy 297 - 38 -  
"-" "S3Console/0.4" -  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be mybucket [06/  
Feb/2014:00:01:00 +0000] 192.0.2.3  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE  
REST.GET.VERSIONING - "GET /mybucket?versioning HTTP/1.1" 200 - 113 - 33 - "-"  
"S3Console/0.4" -  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be mybucket [06/  
Feb/2014:00:01:57 +0000] 192.0.2.3  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be DD6CC733AEXAMPLE  
REST.PUT.OBJECT s3-dg.pdf "PUT /mybucket/s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-"  
"S3Console/0.4" -  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be mybucket [06/  
Feb/2014:00:03:21 +0000] 192.0.2.3  
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be BC3C074D0EXAMPLE  
REST.GET.VERSIONING - "GET /mybucket?versioning HTTP/1.1" 200 - 113 - 28 - "-"  
"S3Console/0.4" -
```

### Note

任何字段都可以设置为 - 以指示数据未知或不可用，或者该字段不适用于此请求。

以下列表介绍了日志记录字段。

### 存储桶拥有者

源存储桶拥有者的规范用户 ID。规范用户 ID 是另一种形式的 AWS 账户 ID。有关规范用户 ID 的更多信息，请参阅 [AWS 账户标识符](#)。有关如何查找您的账户的规范用户 ID 的信息，请参阅 [查找账户的规范用户 ID](#)。

### 示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

### 存储桶

请求处理的存储桶的名称。如果系统收到格式错误的请求且无法确定存储桶，则请求不会显示在任何的服务器访问日志中。

#### 示例条目

```
mybucket
```

### 时间

收到请求的时间。使用 `strftime()` 术语的格式如下所示：[%d/%b/%Y:%H:%M:%S %z]

#### 示例条目

```
[06/Feb/2014:00:00:38 +0000]
```

### 远程 IP

请求者的显式 Internet 地址。中间代理和防火墙可能会隐藏发送请求的计算机的实际地址。

#### 示例条目

```
192.0.2.3
```

### 请求者

请求者的规范用户 ID 或用于未验证请求的 -。如果请求者是 IAM 用户，此字段会返回请求者的 IAM 用户名以及该 IAM 用户所属的 AWS 根账户。此标识符与用于访问控制目的的标识符是相同的。

#### 示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

### 请求 ID

由 Amazon S3 生成的字符串，可用于唯一地标识每个请求。

#### 示例条目

```
3E57427F33A59F07
```

### Operation

此处列出的操作将声明为

SOAP.*operation*、REST.*HTTP\_method.resource\_type*、WEBSITE.*HTTP\_method.resource\_type* 或 BATCH.DELETE.OBJECT。

#### 示例条目

```
REST.PUT.OBJECT
```

### 键

请求的“密钥”部分、已编码的 URL 或“-”(如果操作没有使用密钥参数)。

示例条目

```
/photos/2014/08/puppy.jpg
```

请求-URI

HTTP 请求消息的“请求-URI”部分。

示例条目

```
"GET /mybucket/photos/2014/08/puppy.jpg?x-foo=bar HTTP/1.1"
```

HTTP 状态

响应的数字 HTTP 状态代码。

示例条目

```
200
```

错误代码

Amazon S3 [错误代码 \(p. 499\)](#) 或“-”(如果没有发生任何错误)。

示例条目

```
NoSuchBucket
```

发送的字节数

发送的响应字节数，不包括 HTTP 协议支出或“-”(如果为零)。

示例条目

```
2662992
```

对象大小

所涉及的对象的总大小。

示例条目

```
3462992
```

总时间

从服务器传输请求的毫秒数。该值计算从收到请求到发出响应的最后一个字节的时间。由于网络延迟，从客户端计算出的时间可能会更长。

示例条目

```
70
```

周转时间

Amazon S3 处理您的请求所花费的毫秒数。该值计算从收到您的请求的最后一个字节到发出响应的第一个字节的时间。

#### 示例条目

```
10
```

#### 引用站点

HTTP 引用站点标头的值 (如果存在)。发送请求时，HTTP 用户代理 (例如，浏览器) 通常会将此标头设置为链接的 URL 或嵌入页面。

#### 示例条目

```
"http://www.amazon.com/webservices"
```

#### 用户代理

HTTP 用户代理标头的值。

#### 示例条目

```
"curl/7.15.1"
```

#### 版本 ID

请求中的版本 ID 或“-”(如果操作没有使用 `versionId` 参数)。

#### 示例条目

```
3HL4kqtJvjVBH40Nrjfkd
```

## 自定义访问日志信息

通过向请求中的 URL 添加自定义的查询字符串参数，您可以为请求包含将存储在访问日志记录中的自定义信息。Amazon S3 忽略以“x-”开头的查询字符串参数，但是会将这些参数包含在请求的访问日志记录中，以作为日志记录的 Request-URI 字段的一部分。例如，“`s3.amazonaws.com/mybucket/photos/2014/08/puppy.jpg?x-user=johndoe`”的 GET 请求与“`s3.amazonaws.com/mybucket/photos/2014/08/puppy.jpg`”的同一请求的工作方式相同，唯一的不同是“`x-user=johndoe`”字符串包含在关联日志记录的 Request-URI 字段中。此功能仅在 REST 界面中可用。

## 可扩展服务器访问日志格式的编程注意事项

有时我们可能会通过向每一行的末尾添加新字段来扩展访问日志记录格式。必须写入可解析服务器访问日志的代码以处理后续未知的字段。

## 复制操作的其他日志记录

复制操作包括 GET 和 PUT。出于该原因，我们会在执行复制操作时记录两份记录。前面的表格描述了与操作的 PUT 部分相关的字段。以下列表描述了记录中与复制操作的 GET 部分相关的字段。

#### 存储桶拥有者

用于存储将复制的对象的存储桶的规范用户 ID。规范用户 ID 是另一种形式的 AWS 账户 ID。有关规范用户 ID 的更多信息，请参阅 [AWS 账户标识符](#)。有关如何查找您的账户的规范用户 ID 的信息，请参阅 [查找账户的规范用户 ID](#)。

#### 示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

#### 存储桶

用于存储被复制对象的存储桶的名称。

#### 示例条目

```
mybucket
```

#### 时间

收到请求的时间。使用 `strftime()` 术语的格式如下所示：[%d/%B/%Y:%H:%M:%S %z]

#### 示例条目

```
[06/Feb/2014:00:00:38 +0000]
```

#### 远程 IP

请求者的显式 Internet 地址。中间代理和防火墙可能会隐藏发送请求的计算机的实际地址。

#### 示例条目

```
192.0.2.3
```

#### 请求者

请求者的规范用户 ID 或用于未经验证请求的 -。如果请求者是 IAM 用户，此字段将返回请求者的 IAM 用户名与该 IAM 用户所属的 AWS 根账户。此标识符与用于访问控制目的的标识符是相同的。

#### 示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

#### 请求 ID

由 Amazon S3 生成的字符串，可用于唯一地标识每个请求。

#### 示例条目

```
3E57427F33A59F07
```

#### Operation

此处列出的操作将声明为

SOAP.`operation`、REST.`HTTP_method.resource_type`、WEBSITE.`HTTP_method.resource_type` 或 BATCH.DELETE.OBJECT。

#### 示例条目

```
REST.COPY.OBJECT_GET
```

#### 键

被复制对象的“密钥”或“-”(如果操作没有使用密钥参数)。

示例条目

```
/photos/2014/08/puppy.jpg
```

请求-URI

HTTP 请求消息的“请求-URI”部分。

示例条目

```
"GET /mybucket/photos/2014/08/puppy.jpg?x-foo=bar"
```

HTTP 状态

复制操作的 GET 部分的数字 HTTP 状态代码。

示例条目

```
200
```

错误代码

复制操作的 GET 部分的 Amazon S3 错误代码 ([p. 499](#)) 或“-”(如果没有发生任何错误)。

示例条目

```
NoSuchBucket
```

发送的字节数

发送的响应字节数，不包括 HTTP 协议支出或“-”(如果为零)。

示例条目

```
2662992
```

对象大小

所涉及的对象的总大小。

示例条目

```
3462992
```

总时间

从服务器传输请求的毫秒数。该值计算从收到请求到发出响应的最后一个字节的时间。由于网络延迟，从客户端计算出的时间可能会更长。

示例条目

```
70
```

周转时间

Amazon S3 处理您的请求所花费的毫秒数。该值计算从收到您的请求的最后一个字节到发出响应的第一个字节的时间。

#### 示例条目

```
10
```

#### 引用站点

HTTP 引用站点标头的值 (如果存在)。发送请求时，HTTP 用户代理 (例如，浏览器) 通常会将此标头设置为链接的 URL 或嵌入页面。

#### 示例条目

```
"http://www.amazon.com/webservices"
```

#### 用户代理

HTTP 用户代理标头的值。

#### 示例条目

```
"curl/7.15.1"
```

#### 版本 ID

被复制对象的版本 ID 或“-”(如果 `x-amz-copy-source` 标头没有将 `versionId` 参数指定为复制源的一部分)。

#### 示例条目

```
3HL4kqtJvjVBH40Nrjfkd
```

## 删除 Amazon S3 日志文件

一个已启用服务器访问日志记录的 S3 存储桶可能会随时间推移积累许多服务器日志对象。您的应用程序可能在创建之后的特定时间段内需要这些访问日志，在此之后，您可能希望删除它们。您可以使用 Amazon S3 生命周期配置来设置规则，以便 Amazon S3 可自动对这些对象进行排队，从而在其生命周期结束时进行删除。

您可以使用共享前缀为 S3 存储桶中的一部分对象（即，其名称以通用字符串开头的对象）定义生命周期配置。如果您在服务器访问日志记录配置中指定了前缀，则可以设置生命周期配置规则，以删除具有该前缀的日志对象。例如，如果您的日志对象具有前缀 `logs/`，您可以设置生命周期配置规则，以便在指定的时间段后删除存储桶中具有 `/logs` 前缀的所有对象。有关生命周期配置的更多信息，请参阅[对象生命周期管理 \(p. 104\)](#)。

## 更多信息

[Amazon S3 服务器访问日志记录 \(p. 506\)](#)

# 使用 AWS 开发工具包、CLI 和 Explorer

使用 Amazon S3 开发应用程序时，您可以使用 AWS 开发工具包。AWS 开发工具包包装了底层 REST API，可以简化您的编程任务。还提供 AWS 移动开发工具包和 AWS Amplify JavaScript 库，用于使用 AWS 构建互连移动和 Web 应用程序。

本部分提供使用 AWS 开发工具包开发 Amazon S3 应用程序的概述。本节还描述了如何测试本指南中提供的 AWS 开发工具包代码示例。

## 主题

- [在请求身份验证中指定签名版本 \(p. 519\)](#)
- [设置 AWS CLI \(p. 520\)](#)
- [使用 AWS SDK for Java \(p. 521\)](#)
- [使用适用于 .NET 的 AWS 开发工具包 \(p. 522\)](#)
- [使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例 \(p. 523\)](#)
- [使用适用于 Ruby 的 AWS 开发工具包 - 版本 3 \(p. 524\)](#)
- [使用 AWS SDK for Python \(Boto\) \(p. 525\)](#)
- [使用适用于 iOS 和 Android 的 AWS 移动开发工具包 \(p. 525\)](#)
- [使用 AWS Amplify JavaScript 库 \(p. 525\)](#)

除了 AWS 开发工具包外，AWS Explorer 也适用于 Visual Studio 和 Eclipse for Java IDE。在此情况下，可以将开发工具包和 Explorer 捆绑在一起作为 AWS 工具包。

您也可以使用 AWS 命令行界面 (AWS CLI) 来管理 Amazon S3 存储桶和对象。

## AWS Toolkit for Eclipse

AWS Toolkit for Eclipse 包含 AWS SDK for Java 和 AWS Explorer for Eclipse。AWS Explorer for Eclipse 是适用于 Eclipse for Java IDE 的开源插件，能够让开发人员更为轻松地使用 AWS 开发、调试和部署 Java 应用程序。易于使用的 GUI 使您可以访问和管理您的 AWS 基础设施，包括 Amazon S3。在 Eclipse for Java IDE 环境中，您可以在开发应用程序的同时，执行常见的操作（例如管理您的存储桶和对象，以及设置 IAM 策略）。有关设置说明，请参阅[设置工具包](#)。有关使用 Explorer 的示例，请参阅[如何访问 AWS Explorer](#)。

## AWS Toolkit for Visual Studio

AWS Explorer for Visual Studio 是针对 Microsoft Visual Studio 的扩展，使开发人员能够更容易地使用 Amazon Web Services 来开发、调试和部署 .NET 应用程序。易于使用的 GUI 使您可以访问和管理您的 AWS 基础设施，包括 Amazon S3。在 Visual Studio 环境中，您可以在开发应用程序的同时，执行常见的操作（例如管理您的存储桶和对象，或设置 IAM 策略）。有关设置说明，请转至[设置 AWS Toolkit for Visual Studio](#)。有关通过 Explorer 使用 Amazon S3 的示例，请参阅[从 AWS Explorer 使用 Amazon S3](#)。

## AWS 开发工具包

您只能下载开发工具包。有关下载开发工具包库的信息，请参阅[示例代码库](#)。

## AWS CLI

AWS CLI 是用于管理您的 AWS 服务 (包括 Amazon S3) 的统一工具。有关下载 AWS CLI 的信息 , 请参阅 [AWS Command Line Interface](#)。

## 在请求身份验证中指定签名版本

Amazon S3 在大多数 AWS 区域中只支持签名版本 4。不过，在某些较旧的 AWS 区域中，Amazon S3 同时支持签名版本 4 和签名版本 2。有关所有 Amazon S3 区域的列表以及这些区域支持的签名版本，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 开发工具包使用签名版本 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，您可能需要请求签名版本 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	<p>对于默认配置文件，运行以下命令：</p> <pre>\$ aws configure set default.s3.signature_version s3v4</pre> <p>对于自定义配置文件，运行以下命令：</p> <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
Java 软件开发工具包	<p>在代码中添加以下内容：</p> <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> <p>或者在命令行中指定以下内容：</p> <pre>-Dcom.amazonaws.services.s3.enableV4</pre>
JavaScript 开发工具包	<p>在构建客户端时，将 <code>signatureVersion</code> 参数设置为 v4：</p> <pre>var s3 = new AWS.S3({signatureVersion: 'v4'});</pre>
PHP 开发工具包	<p>在构建 Amazon S3 服务客户端时，将 <code>signature</code> 参数设置为 v4：</p> <pre>&lt;?php \$s3 = new S3Client(['signature' =&gt; 'v4']);</pre>
Python-Boto 开发工具包	<p>在默认配置文件 <code>boto</code> 中指定以下内容：</p> <pre>[s3] use-sigv4 = True</pre>
Ruby 开发工具包	<p>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>：</p> <pre>s3 = AWS::S3::Client.new(:s3_signature_version =&gt; :v4)</pre>

开发工具包	请求使用签名版本 4 进行请求身份验证
	<p>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 v4：</p> <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>
.NET 开发工具包	<p>在创建 Amazon S3 客户端之前将以下内容添加到代码中：</p> <pre>AWSConfigs.S3UseSignatureVersion4 = true;</pre> <p>或将以下内容添加到配置文件中：</p> <pre>&lt;appSettings&gt;     &lt;add key="AWS.S3.UseSignatureVersion4" value="true" /&gt; &lt;/appSettings&gt;</pre>

## 设置 AWS CLI

按照步骤下载和配置 AWS 命令行界面 (AWS CLI)。

### 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。您随后便可以使用一个特殊的 URL 和该 IAM 用户的凭证访问 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](#)。

### 设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅 AWS 命令行界面用户指南中的以下主题：
  - [使用 AWS 命令行界面进行设置](#)
  - [配置 AWS 命令行界面](#)
2. 在 AWS CLI 配置文件中为管理员用户添加一个命名配置文件。在执行 AWS CLI 命令时，您将使用此配置文件。

```
[adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

有关可用 AWS 区域的列表，请参阅 AWS General Reference 中的 [区域和终端节点](#)。

3. 在命令提示符处输入以下命令来验证设置。
  - 尝试 `help` 命令来验证您的计算机上是否安装了 AWS CLI：

```
aws help
```

- 尝试 `s3` 命令来验证用户是否可以访问 Amazon S3。此命令将列出您的账户中的存储桶。AWS CLI 使用 `adminuser` 凭证来验证请求。

```
aws s3 ls --profile adminuser
```

## 使用 AWS SDK for Java

AWS SDK for Java 为 Amazon S3 存储桶和对象操作提供了 API。对于对象操作，除了提供用于在单个操作中上传对象的 API 外，开发工具包还提供 API 来分段上传大型对象。有关更多信息，请参阅 [使用分段上传 API 上传对象 \(p. 155\)](#)。

### 主题

- [Java API 组织 \(p. 521\)](#)
- [测试 Amazon S3 Java 代码示例 \(p. 522\)](#)

AWS SDK for Java 允许您选择使用高级别或低级别 API。

低级别 API。

低级别 API 对应于底层 Amazon S3 REST 操作，例如应用于存储桶和对象的创建、更新和删除操作。如果使用低级别分段上传 API 上传大型对象，它可以提供更好的控制。例如，可使用它暂停和恢复分段上传，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。如果您没有这些要求，请使用高级别 API 上传对象。

高级别 API

对于上传对象，开发工具包通过提供 `TransferManager` 类来提供更高级别的抽象。高级别 API 是更简单的 API，只需几行代码，即可将文件和流上传到 Amazon S3。除非您需要控制上传（如前面的低级别 API 部分所述），否则您应该使用此 API 来上传数据。

对于较小的数据，可使用 `TransferManager` API 通过单个操作上传数据。但是，当数据大小达到特定的阈值后，`TransferManager` 会转为使用分段上传 API。如果可能，`TransferManager` 会使用多个线程来并发上传分段。如果分段上传失败，API 最多会重试三次失败的分段上传。但是，可以使用 `TransferManagerConfiguration` 类来配置这些选项。

### Note

将流用作数据源时，`TransferManager` 类不会执行并发上传。

## Java API 组织

AWS SDK for Java 在以下包中提供 API：

- `com.amazonaws.services.s3` - 提供用于创建 Amazon S3 客户端以及使用存储桶和对象的 API。例如，该 API 可让您创建存储桶、上传对象、获取对象、删除对象和列出键。
- `com.amazonaws.services.s3.transfer` - 提供高级别 API 数据操作。

此高级别 API 旨在简化对象与 Amazon S3 之间的传输。它包含 `TransferManager` 类，该类提供了使用、查询和操作传输的异步方法。它还包括 `TransferManagerConfiguration` 类，您可以使用该类配置用于上传分段的最小分段大小以及使用分段上传时的阈值（字节）。

- `com.amazonaws.services.s3.model` - 提供低级别 API 类来创建请求和处理响应。例如，它包含用于描述获取对象请求的 `GetObjectRequest` 类、用于描述列出键请求的 `ListObjectsRequest` 类以及用于创建分段上传的 `InitiateMultipartUploadRequest` 类。

有关 AWS SDK for Java API 的更多信息，请参阅 [AWS SDK for Java API Reference](#)。

## 测试 Amazon S3 Java 代码示例

本指南中的 Java 示例与 AWS SDK for Java 版本 1.11.321 兼容。有关设置和运行代码示例的说明，请参阅 AWS SDK for Java 开发人员指南中的[AWS SDK for Java 入门](#)。

## 使用适用于 .NET 的 AWS 开发工具包

适用于 .NET 的 AWS 开发工具包为 Amazon S3 存储桶和对象操作提供 API。对于对象操作，除了提供用于在单个操作中上传对象的 API 外，开发工具包还提供 API 来分段上传大型对象 (参阅[使用分段上传 API 上传对象 \(p. 155\)](#))。

### 主题

- [.NET API 组织 \(p. 522\)](#)
- [运行 Amazon S3 .NET 代码示例 \(p. 523\)](#)

适用于 .NET 的 AWS 开发工具包 允许您选择使用高级别或低级别 API。

低级别 API。

低级别 API 对应于底层 Amazon S3 REST 操作，包括应用于存储桶和对象的创建、更新和删除操作。如果使用低级别分段上传 API 上传大型对象 (请参阅[使用分段上传 API 上传对象 \(p. 155\)](#))，它可以提供更好的控制。例如，可使用它暂停和恢复分段上传，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。如果您没有这些要求，请使用高级别 API 来上传对象。

高级别 API

对于上传对象，开发工具包通过提供 TransferUtility 类来提供更高级别的抽象。高级别 API 是更简单的 API，只需几行代码，即可将文件和流上传到 Amazon S3。除非您需要控制上传 (如前面的低级别 API 部分所述)，否则您应该使用此 API 来上传数据。

对于较小的数据，可使用 TransferUtility API 通过单个操作上传数据。但是，当数据大小达到特定的阈值后，TransferUtility 会转为使用分段上传 API。默认情况下，它使用多个线程并发上传分段。如果分段上传失败，API 最多会重试三次失败的分段上传。但是，这些是可配置选项。

### Note

如果您将流用作数据源，TransferUtility 类不会执行并发上传。

## .NET API 组织

使用适用于 .NET 的 AWS 开发工具包写入 Amazon S3 应用程序时，可使用 AWSSDK.dll。此程序集中的以下命名空间提供分段上传 API：

- Amazon.S3.Transfer – 提供高级别 API 来分段上传您的数据。

它包括允许您指定用于上传数据的文件、目录或流的 TransferUtility 类。它还包括 TransferUtilityUploadRequest 和 TransferUtilityUploadDirectoryRequest 类来配置高级设置，例如并发线程数量、分段大小、对象元数据、存储类 (STANDARD、REDUCED\_REDUNDANCY) 和对象访问控制列表 (ACL)。

- Amazon.S3 – 提供适用于低级别 API 的执行。

它提供对应于 Amazon S3 REST 分段上传 API 的方法 (请参阅[使用适用于分段上传的 REST API \(p. 183\)](#))。

- Amazon.S3.Model – 提供低级别 API 来创建请求和处理响应。例如，它提供在启动分段上传时使用的 `InitiateMultipartUploadRequest` 和 `InitiateMultipartUploadResponse` 类，以及在上传分段时使用的 `UploadPartRequest` 和 `UploadPartResponse` 类。
- Amazon.S3.Encryption - 提供 `AmazonS3EncryptionClient`。
- Amazon.S3.Util - 提供各种实用程序类 (如 `AmazonS3Util` 和 `BucketRegionDetector`)。

有关适用于 .NET 的 AWS 开发工具包 API 的更多信息，请参阅[适用于 .NET 的 AWS 开发工具包版本 3 API 参考](#)。

## 运行 Amazon S3 .NET 代码示例

本指南中的 .NET 代码示例与适用于 .NET 的 AWS 开发工具包版本 3.0 兼容。有关设置和运行代码示例的说明，请参阅[适用于 .NET 的 AWS 开发工具包开发人员指南 中的适用于 .NET 的 AWS 开发工具包入门](#)。

# 使用适用于 PHP 的 AWS 开发工具包和运行 PHP 示例

适用于 PHP 的 AWS 开发工具包提供对用于 Amazon S3 存储桶和对象操作的 API 的访问。该开发工具包为您提供了使用服务的低级别 API 或使用高级别抽象的选项。

您可以在[适用于 PHP 的 AWS 开发工具包](#)中找到该开发工具包，其中还包括该开发工具包的安装和入门相关说明。

适用于 PHP 的 AWS 开发工具包使用的相关设置取决于您的环境以及您要如何运行自己的应用程序。要设置您的环境以运行本文档中的示例，请参阅[适用于 PHP 的 AWS 开发工具包 Getting Started Guide](#)。

### 主题

- [适用于 PHP 的 AWS 开发工具包级别 \(p. 523\)](#)
- [运行 PHP 示例 \(p. 524\)](#)
- [相关资源 \(p. 524\)](#)

## 适用于 PHP 的 AWS 开发工具包级别

适用于 PHP 的 AWS 开发工具包 允许您选择使用高级别或低级别 API。

### 低级别 API。

低级别 API 对应于底层 Amazon S3 REST 操作，包括对存储桶和对象执行的创建、更新和删除操作。低级别 API 提供了对这些操作的更多控制。例如，可以批量处理请求并平行执行它们。或者，在使用分段上传 API 时，可以单独管理对象分段。请注意，这些低级别 API 调用将返回包含所有 Amazon S3 响应详细信息的结果。有关分段上传 API 的更多信息，请参阅[使用分段上传 API 上传对象 \(p. 155\)](#)。

### 高级别抽象

高级别抽象旨在简化常用情况。例如，要使用低级别 API 上传大型对象，应先调用 `Aws\S3\S3Client::createMultipartUpload()`，然后调用 `Aws\S3\S3Client::uploadPart()` 方法上传对象分段，再调用 `Aws\S3\S3Client::completeMultipartUpload()` 方法完成上传。可改用高级别 `Aws\S3\MultipartUploader` 对象来简化分段上传的创建。

再如，当枚举存储桶中的对象时，可以使用适用于 PHP 的 AWS 开发工具包的迭代器功能返回所有对象键，无论存储桶中存储了多少对象都是如此。如果使用了低级别 API，响应将最多返回 1000 个键。如果存储桶包含 1000 以上的对象，结果将被截断，您必须管理响应并检查截断。

## 运行 PHP 示例

要针对适用于 PHP 的 AWS 开发工具包版本 3 设置并使用 Amazon S3 示例，请参阅适用于 PHP 的 AWS 开发工具包 Developer Guide 中的[安装](#)。

## 相关资源

- 用于 Amazon S3 的适用于 PHP 的 AWS 开发工具包
- 适用于 PHP 的 AWS 开发工具包文档

# 使用适用于 Ruby 的 AWS 开发工具包 - 版本 3

适用于 Ruby 的 AWS 开发工具包为 Amazon S3 存储桶和对象操作提供 API。对于对象操作，您可以使用 API 在单一操作中上传对象或分批上传大量对象（请参阅[使用适用于 Ruby 的 AWS 开发工具包进行分段上传 \(p. 182\)](#)）。但是，用于单个操作上传的 API 也可以接受大型对象，并在幕后为您管理分段上传，从而减少您需要编写的脚本数量。

## Ruby API 组织

使用适用于 Ruby 的 AWS 开发工具包创建 Amazon S3 应用程序时，必须安装适用于 Ruby 的开发工具包 gem。有关更多信息，请参阅[适用于 Ruby 的 AWS 开发工具包 - 版本 3](#)。安装后，您可以访问包含以下密钥类的 API：

- Aws::S3::Resource - 表示适用于 Ruby 开发工具包的 Amazon S3 的接口并提供用于创建和枚举存储桶的方法。

S3 类提供了用于访问现有存储桶或创建新存储桶的 #buckets 实例方法。

- Aws::S3::Bucket - 表示 Amazon S3 存储桶。

Bucket 类提供了用于访问存储桶中对象的 #object(key) 和 #objects 方法，以及删除存储桶和返回有关存储桶的信息（例如存储桶策略）的方法。

- Aws::S3::Object - 表示由其键标识的 Amazon S3 对象。

Object 类提供了用于获取和设置对象的属性、指定用于存储对象的存储类，以及使用访问控制列表设置对象权限的方法。Object 类还具有用于删除、上传和复制对象的方法。分段上传对象时，此类将向您提供选项以指定上传的分段顺序和分段大小。

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

## 测试 Ruby 脚本示例

开始使用 Ruby 脚本示例的最简单方式是安装最新的适用于 Ruby 的 AWS 开发工具包 gem。有关安装或更新至最新 gem 的信息，请转到[适用于 Ruby 的 AWS 开发工具包 - 版本 3](#)。以下任务将指导您完成 Ruby 脚本示例的创建和测试过程（假设已安装适用于 Ruby 的 AWS 开发工具包）。

创建和测试 Ruby 脚本示例的常规过程

1	要访问 AWS，您必须为您的适用于 Ruby 的开发工具包应用程序提供一组凭证。有关更多信息，请参阅 <a href="#">配置适用于 Ruby 的 AWS SDK</a> 。
---	---

2	创建新的适用于 Ruby 的开发工具包脚本并将以下各行添加到脚本顶部。
	<pre>#!/usr/bin/env ruby  require 'rubygems' require 'aws-sdk-s3'</pre>
	第一行是解释程序指令，且两个 require 语句会将两个所需版本导入到您的脚本。
3	将代码从您正在阅读的部分复制到您的脚本。
4	通过提供任意所需数据更新代码。例如，如果上传某个文件，则提供文件路径和存储桶名称。
5	运行脚本。验证使用 AWS 管理控制台对存储桶和对象进行的更改。有关 AWS 管理控制台的更多信息，请转到 <a href="https://aws.amazon.com/console/">https://aws.amazon.com/console/</a> 。

### Ruby 示例

以下链接包含的示例可帮助您开始使用适用于 Ruby 的开发工具包 - 版本 3：

- [使用适用于 Ruby 的 AWS 开发工具包版本 3 \(p. 57\)](#)
- [使用适用于 Ruby 的 AWS 开发工具包 上传对象 \(p. 154\)](#)

## 使用 AWS SDK for Python (Boto)

Boto 是 Python 包，提供 AWS 接口（包括 Amazon S3）。有关 Boto 的更多信息，请参阅[AWS SDK for Python \(Boto\)](#)。本页上的入门链接提供了入门的分步指南。

## 使用适用于 iOS 和 Android 的 AWS 移动开发工具包

您可将适用于 Android 和 iOS 的 AWS 移动开发工具包与 [AWS Mobile Hub](#) 配合使用，快速轻松地将稳健的云后端集成到您的现有移动应用程序中。您不必成为 AWS 专家即可配置和使用用户登录、数据库、推送通知等功能。

使用 AWS 移动开发工具包可以轻松访问 Amazon S3 以及许多其他 AWS 服务。要开始使用 AWS 移动开发工具包，请参阅[AWS 移动开发工具包入门](#)。

### 更多信息

[使用 AWS Amplify JavaScript 库 \(p. 525\)](#)

## 使用 AWS Amplify JavaScript 库

AWS Amplify 是一个开源 JavaScript 库，可供 Web 和移动开发人员构建支持云的应用程序。AWS Amplify 提供可定制的 UI 组件和声明性接口，可与 S3 存储桶及其他高级类别 AWS 服务配合使用。

要开始使用 AWS Amplify JavaScript 库，请选择下列链接之一：

- [适用于 Web 的 AWS Amplify 库入门](#)

- [适用于 React Native 的 AWS Amplify 库入门](#)

有关 AWS Amplify 的更多信息，请参阅 [GitHub 上的 AWS Amplify](#)。

## 更多信息

[使用适用于 iOS 和 Android 的 AWS 移动开发工具包 \(p. 525\)](#)

# 附录

此 Amazon Simple Storage Service 开发人员指南附录包括下列部分。

## 主题

- [附录 A : 使用 SOAP API \(p. 527\)](#)
- [附录 B : 对请求进行身份验证 \( AWS 签名版本 2 \) \(p. 529\)](#)

## 附录 A : 使用 SOAP API

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

本节包含 Amazon S3 SOAP API 的特定信息。

### 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。

## 常见 SOAP API 元素

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。我们建议您使用 REST API 或 AWS 开发工具包。

您可以使用 SOAP 1.1 通过 HTTP 与 Amazon S3 进行交互。在 <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl> 上提供了有关 Amazon S3 WSDL (以计算机可读的方式描述 Amazon S3 API) 的信息。在 <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.xsd> 中提供了有关 Amazon S3 架构的信息。

大多数用户将使用为他们的语言和开发环境定制的 SOAP 工具包与 Amazon S3 进行交互。不同的工具包将以不同的方式提供 Amazon S3 API。请参照您的特定工具包文档来了解如何使用它。本节通过在 Amazon S3 SOAP 操作显示“在线”时呈现 XML 请求和响应，以独立于工具包的方式演示这些操作。

## 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

## 对 SOAP 请求进行身份验证

### Note

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。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，以便确立创建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 [创建请求 \(p. 9\)](#)。

- **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。如果此时戳超过了 Amazon S3 服务器上的时间 15 分钟，授权将失败。
- **Signature**：“AmazonS3” + OPERATION + Timestamp 串联的 RFC 2104 HMAC-SHA1 摘要 (参阅 <http://www.ietf.org/rfc/rfc2104.txt>)，该摘要将您的 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="http://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**

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。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。AccessControlList 元素在 [管理对 Amazon S3 资源的访问权限 \(p. 242\)](#) 中进行了描述。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="http://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="http://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</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。

## 附录 B：对请求进行身份验证 ( AWS 签名版本 2 )

### 主题

- [使用 REST API 对请求进行身份验证 \(p. 531\)](#)
- [签署和对 REST 请求进行身份验证 \(p. 532\)](#)
- [使用 POST \( AWS 签名版本 2 \) 的基于浏览器的上传 \(p. 541\)](#)

### Note

本主题说明使用签名版本 2 对请求进行身份验证。Amazon S3 目前支持最新的签名版本 4，所有区域均支持此版本；它是新 AWS 区域唯一支持的版本。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的[对请求进行身份验证 \(AWS 签名版本 4\)](#)。



## 使用 REST API 对请求进行身份验证

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

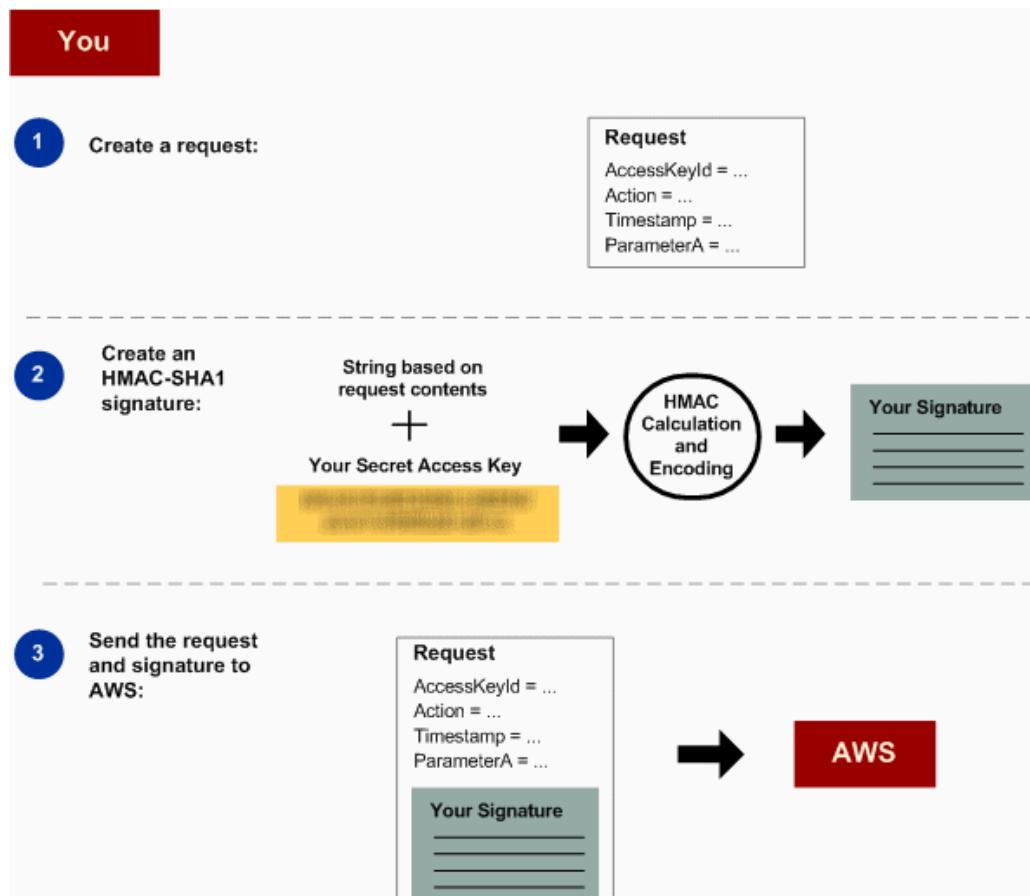
- AWS 访问密钥 ID – 每个请求必须包含用于发送请求的身份的访问密钥 ID。
- 签名 – 每个请求必须包含一个有效的请求签名，否则请求将被拒绝。

将使用您的秘密访问密钥（它是一个共享的密钥，只有您和 AWS 知道）计算出请求签名。

- 时间戳 – 每个请求必须包含请求的创建日期和时间，并使用 UTC 中的字符串表示。
- 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。

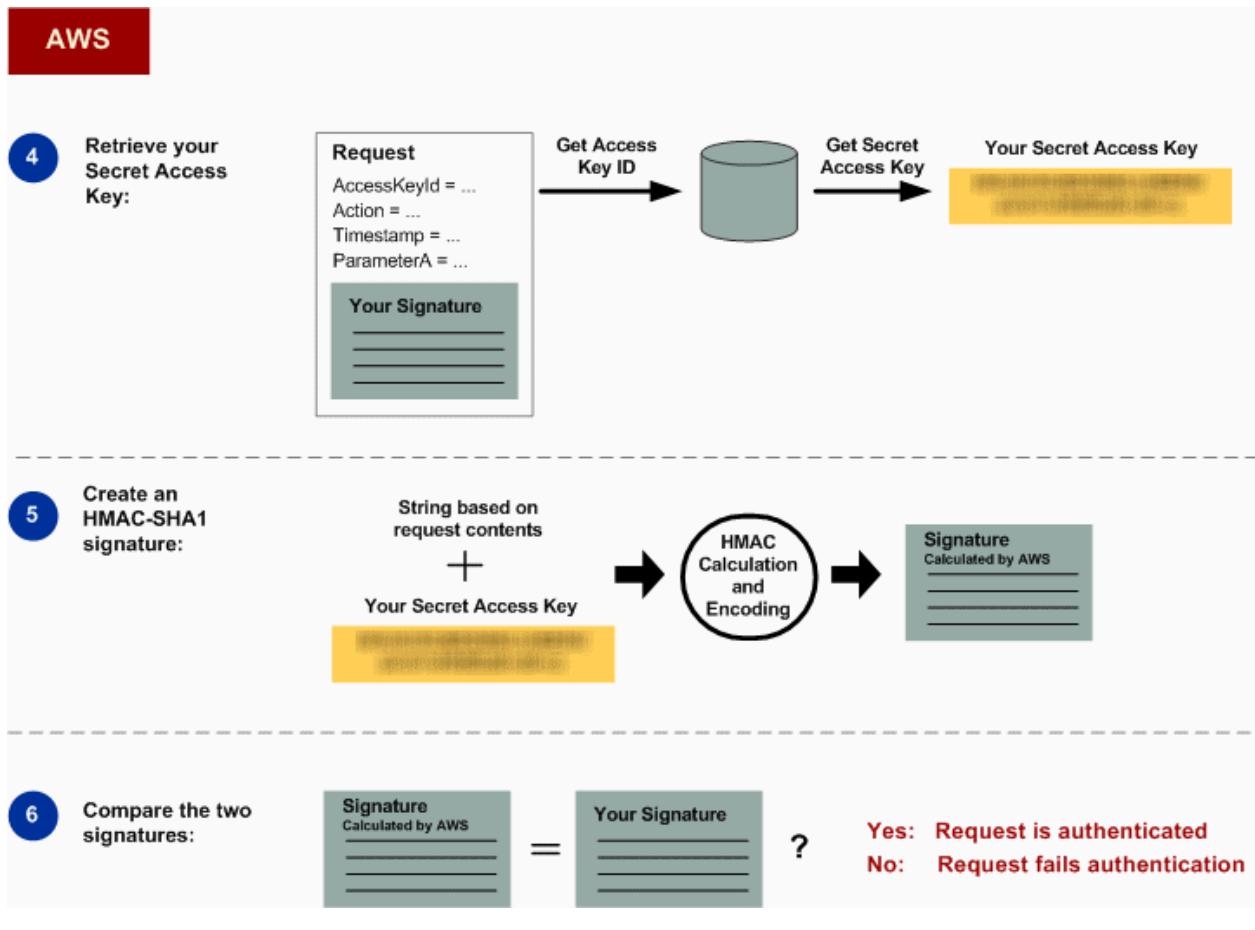
下面是针对发送到 Amazon S3 的请求进行身份验证的常规步骤。此处假设您已拥有了必要的安全凭证、访问密钥 ID 和秘密访问密钥。



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。



4

Amazon S3 使用访问密钥 ID 来查找秘密访问密钥。

5

通过与用于计算您在请求中发送的签名相同的算法，Amazon S3 可根据请求数据和秘密访问密钥计算出签名。

6

如果由 Amazon S3 生成的签名与您在请求中发送的签名相匹配，将认为请求是真实的。如果比较签名这一操作失败，那么请求将被丢弃，同时 Amazon S3 将返回错误响应。

## 详细的身份验证信息

有关 REST 身份验证的详细信息，请参阅 [签署和对 REST 请求进行身份验证 \(p. 532\)](#)。

## 签署和对 REST 请求进行身份验证

### 主题

- [使用临时安全凭证 \(p. 533\)](#)
- [身份验证标头 \(p. 534\)](#)
- [用于签名的请求规范 \(p. 534\)](#)

- 构建 CanonicalizedResource 元素 (p. 534)
- 构建 CanonicalizedAmzHeaders 元素 (p. 535)
- 位置和命名 HTTP 标头 StringToSign 元素的对比 (p. 536)
- 时间戳要求 (p. 536)
- 身份验证示例 (p. 536)
- REST 请求签名问题 (p. 539)
- 查询字符串请求身份验证替代项 (p. 539)

#### Note

本主题说明使用签名版本 2 对请求进行身份验证。Amazon S3 现在支持最新的签名版本 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，请参阅 Amazon Simple Storage Service API Reference 中的[对请求进行身份验证 \(AWS 签名版本 4\)](#)。

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）将为相关的存储桶拥有者保留在存储桶中创建对象的权利。作为开发人员，您需要创建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 \( AWS 签名版本 2 \) 的基于浏览器的上传 \(p. 541\)](#)。

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: johnsmith.s3.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:frJIUN8DYpKDtOLCwo//yllqDzg=
```

## 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[创建请求 \(p. 9\)](#)），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，请参阅 AWS Security Token Service API Reference 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认证信息，而不是授权信息。) 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，会向开发人员颁发 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，则请求者证明拥有 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( YourSecretAccessKeyID, 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", "?logging", or "?torrent"];

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(YourSecretAccessKeyID)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：

### 启动过程

- 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 \(p. 42\)](#)。

	<p>对于虚拟托管类型请求“<a href="https://johnsmith.s3.amazonaws.com/photos/puppy.jpg">https://johnsmith.s3.amazonaws.com/photos/puppy.jpg</a>”，CanonicalizedResource 是“/johnsmith”。</p> <p>对于路径类型请求“<a href="https://s3.amazonaws.com/johnsmith/photos/puppy.jpg">https://s3.amazonaws.com/johnsmith/photos/puppy.jpg</a>”，CanonicalizedResource 是“”。</p>
3	<p>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)。</p> <p>对于虚拟托管类型请求“<a href="https://johnsmith.s3.amazonaws.com/photos/puppy.jpg">https://johnsmith.s3.amazonaws.com/photos/puppy.jpg</a>”，CanonicalizedResource 是“/johnsmith/photos/puppy.jpg”。</p> <p>对于路径类型请求“<a href="https://s3.amazonaws.com/johnsmith/photos/puppy.jpg">https://s3.amazonaws.com/johnsmith/photos/puppy.jpg</a>”，CanonicalizedResource 是“/johnsmith/photos/puppy.jpg”。此时，虚拟托管类型和路径类型请求的 CanonicalizedResource 相同。</p> <p>对于不寻址存储桶的请求 (例如，<a href="#">GET Service</a>)，请附加“/”。</p>
4	<p>如果请求将寻址子资源 (例如，?versioning、?location、?acl、?torrent、?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“&amp;”进行分隔 (例如，?acl&amp;versionId=#)。</p> <p>构建 CanonicalizedResource 元素时必须包含的子资源为： acl、lifecycle、location、logging、notification、partNumber、policy、requestPayment、torrent、uploadId、upload 和 website。</p> <p>如果请求指定覆盖响应标头值的查询字符串参数 (请参阅 <a href="#">Get Object</a>)，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。</p> <p>为多对象删除请求创建 CanonicalizedResource 时，必须包括 delete 查询字符串参数。</p>

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 \(p. 42\)](#)。

## 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。

### CanonicalizedAmzHeaders 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	按照 RFC 2616 中第 4.2 节中的规定，将相同名称的标头字段合并为一个“header-name:comma-separated-value-list”对，各值之间不留空格。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 (包括换行符) 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”改为“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、日期和 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。相比之下，“x-amz-”元素则属于命名标头。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 \(p. 539\)](#)。

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 RFC 2616 格式 (<http://www.ietf.org/rfc/rfc2616.txt>) 中的任意格式。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 StringToSign 时使用 Date 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

本示例将从 johnsmith 存储桶获取一个对象。

请求	StringToSign
GET /photos/puppy.jpg HTTP/1.1 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000	GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n

请求	StringToSign
<pre>Authorization: AWS AKIAIOSFODNN7EXAMPLE: bWq2s1WEIj+Ydj0vQ697zp+IXMU=</pre>	/johnsmith/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)。

## 对象 PUT

本示例将对象放置在 johnsmith 存储桶中。

请求	StringToSign
<pre>PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000  Authorization: AWS AKIAIOSFODNN7EXAMPLE: MyyxerY7whkBe+bq8fHCL/2kKUg=</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /johnsmith/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

本示例列出了 johnsmith 存储桶中的内容。

请求	StringToSign
<pre>GET /?prefix=photos&amp;max-keys=50&amp;marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:42:41 +0000  Authorization: AWS AKIAIOSFODNN7EXAMPLE: htDYFYduRNen8P9ZfE/s9SuKy0U=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n /johnsmith/</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“johnsmith”存储桶提取访问控制策略子资源。

请求	StringToSign
<pre>GET /?acl HTTP/1.1 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:44:46 +0000  Authorization: AWS AKIAIOSFODNN7EXAMPLE:</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n</pre>

请求	StringToSign
<code>C2WLPFtWHVgbEmeEG93a4cG37dM=</code>	<code>/johnsmith/?acl</code>

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“johnsmith”存储桶删除对象。

请求	StringToSign
<pre>DELETE /johnsmith/photos/puppy.jpg HTTP/1.1 User-Agent: dotnet Host: s3.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:lx3byBScXR6KzyMaifNkardMwNk=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /johnsmith/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.johnsmith.net:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000  x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw==\n X-Amz-Meta-ReviewedBy: joe@johnsmith.net X-Amz-Meta-ReviewedBy: jane@johnsmith.net 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: iIyl83RwaSoYIEdxDQcA4OnAnc=</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@johnsmith.net,jane@johnsmith.net\n /static.johnsmith.net/db-backup.dat.gz</pre>

请注意对“x-amz-”标头排序、删减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。

此外，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
GET / HTTP/1.1 Host: s3.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000  <b>Authorization: AWS</b> <b>AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh6OqZehG9s=</b>	GET\n\n\nWed, 28 Mar 2007 01:29:59 +0000\n/

## Unicode 密钥

请求	StringToSign
GET /dictionary/fran%C3%A7ais/pr%c3%a9re HTTP/1.1 Host: s3.amazonaws.com Date: Wed, 28 Mar 2007 01:49:49 +0000  <b>Authorization: AWS</b> <b>AKIAIOSFODNN7EXAMPLE:DNEZGsoieTZ92F3bUFSPQcbGm%2c3%a8re</b>	GET\n\n\nWed, 28 Mar 2007 01:49:49 +0000\n/dictionary/fran%C3%A7ais/pr%c3%a9re

### 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，SignatureDoesNotMatch 错误文档的 StringToSign 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。

### Note

有关使用 AWS 开发工具包生成预签名 URL 的示例，请参阅[与其他用户共享数据元 \(p. 147\)](#)。

## 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: johnsmith.s3.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
AWSAccessKeyId	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 %2ByT272YEAiv4%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( YourSecretAccessKeyID, UTF-8-Encoding-
Of( StringToSign ) ) ) );

StringToSign = HTTP-VERB + "\n" +
Content-MD5 + "\n" +
Content-Type + "\n" +
Expires + "\n" +
CanonicalizedAmzHeaders +
CanonicalizedResource;
```

YourSecretAccessKeyID 是在您注册成为 Amazon Web Service 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
GET /photos/puppy.jpg? AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE& Signature=NpgCjnDzrM %2BWFWzoENXmpNDUsSn8%3D& Expires=1175139620 HTTP/1.1	GET\n \n \n 1175139620\n

请求	StringToSign
Host: johnsmith.s3.amazonaws.com	/johnsmith/photos/puppy.jpg

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 (p. 536)。

## 使用 POST ( AWS 签名版本 2 ) 的基于浏览器的上传

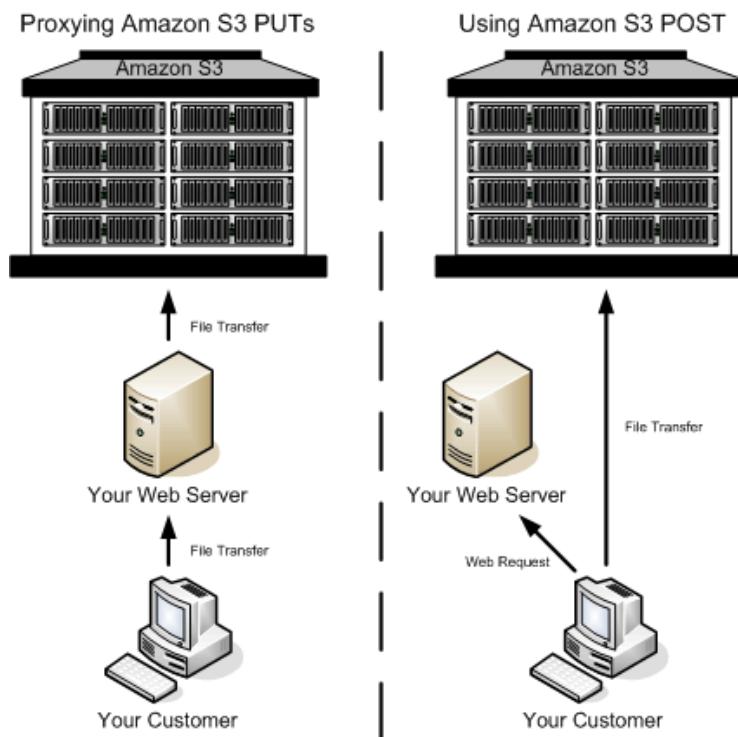
Amazon S3 支持 POST，使您的用户可以直接将内容上传到 Amazon S3。POST 旨在简化上传过程和缩短上传延迟，而且可以节省用于上传数据以存储于 Amazon S3 中所用应用程序的开支。

### Note

本节中讨论的请求身份验证基于 AWS 签名版本 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 区域支持签名版本 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 - 签名版本 2。于 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。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的[对使用 POST \(AWS 签名版本 4\) 的基于浏览器的上传中的请求进行身份验证](#)。

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 签名版本 2)。

### 主题

- [HTML 表单编码 \(p. 542\)](#)
- [HTML 表单声明 \(p. 543\)](#)
- [HTML 表单字段 \(p. 543\)](#)
- [策略构建 \(p. 545\)](#)
- [构建签名 \(p. 548\)](#)
- [重定向 \(p. 549\)](#)

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

HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。对于 SOAP 将不支持新 Amazon S3 功能。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，如果存储桶的名称是“johnsmith”，则 URL 为“<http://johnsmith.s3.amazonaws.com>”。

### Note

在表单字段中指定键名称。

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](#)。

### Example

以下示例是用于存储桶“johnsmith”的表单声明。

```
<form action="http://johnsmith.s3.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，该拥有者将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。有关 ACL 的更多信息，请参阅 <a href="#">访问控制列表 (p. 6)</a> 。  类型：字符串  默认值：private  Valid Values: private   public-read   public-read-write   aws-exec-read   authenticated-read   bucket-owner-read   bucket-owner-full-control	否

字段名称	说明	必需
<code>Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires</code>	特定于 REST 的标头。有关更多信息，请参阅 <a href="#">PUT Object</a> 。	否
<code>key</code>	<p>已上传的密钥的名称。</p> <p>要使用由用户提供文件名，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>。</p> <p>有关更多信息，请参阅 <a href="#">对象键和元数据 (p. 86)</a>。</p>	是
<code>policy</code>	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
<code>success_action_redirect, redirect</code>	<p>上传成功后客户端重定向到的 URL。Amazon S3 将存储桶、键和 etag 值作为查询字符串参数附加到 URL。</p> <p>如果未指定 <code>success_action_redirect</code>，Amazon S3 将返回在 <code>success_action_status</code> 字段中指定的空文档类型。</p> <p>如果 Amazon S3 无法解释 URL，它将忽略该字段。</p> <p>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。</p> <p>有关更多信息，请参阅 <a href="#">重定向 (p. 549)</a>。</p> <p><b>Note</b></p> <p>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。</p>	否

字段名称	说明	必需
<code>success_action_status</code>	<p>如果没有指定 <code>success_action_redirect</code>，上传成功后状态代码将返回到客户端。</p> <p>有效值为 200、201 或 204 (默认)。</p> <p>如果值被设置为 200 或 204，Amazon S3 将返回一个空文档和一个 200 或 204 状态代码。</p> <p>如果值被设置为 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 <a href="#">POST 对象</a>。</p> <p>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。</p> <p><b>Note</b></p> <p>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 <code>success_action_status</code> 设置为 201。</p>	否
<code>signature</code>	HMAC 签名，使用与提供的 AWSAccessKeyId 对应的秘密访问密钥构建。如果策略文档没有随请求一起提供，则此字段为必填字段。	条件
<code>x-amz-security-token</code>	<p>会话凭证使用的安全令牌。</p> <p>如果请求使用 Amazon DevPay，则它需要两个 <code>x-amz-security-token</code> 表单字段：一个用于产品标识，另一个用于用户令牌。</p> <p>如果请求使用会话凭证，则它需要一个 <code>x-amz-security-token</code> 表单。有关更多信息，请参阅 <a href="#">IAM 用户指南</a> 中的临时安全凭证。</p>	否
其他以 <code>x-amz-meta-</code> 为前缀的字段名称	<p>特定于用户的元数据。</p> <p>Amazon S3 没有验证或使用此数据。</p> <p>有关更多信息，请参阅 <a href="#">PUT Object</a>。</p>	否
文件	<p>文件或文本内容。</p> <p>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。</p> <p>您一次仅能上传一个文件。</p>	是

## 策略构建

### 主题

- [过期 \(p. 546\)](#)

- [条件 \(p. 546\)](#)
- [条件匹配 \(p. 547\)](#)
- [字符转义 \(p. 548\)](#)

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": "johnsmith" },
    ["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 \(p. 547\)](#)。

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 starts-with。

元素名称	说明
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 starts-with。
键	已上传的键的名称。 支持精确匹配和 starts-with。
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 starts-with。
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。 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：一个用于产品标识，另一个用于用户令牌。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	<p>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：</p> <pre>{ "acl": "public-read" }</pre> <p>此示例是指示 ACL 必须设置为公共读取的替代方法：</p> <pre>[ "eq", "\$acl", "public-read" ]</pre>

Condition	说明
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：  [ "starts-with", "\$key", "user/betty/" ]
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：  [ "starts-with", "\$success_action_redirect", "" ]
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：  [ "content-length-range", 1048579, 10485760 ]

## 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	描述
\\	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

## 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。

有关身份验证的常见信息，请参阅[使用身份验证访问](#)。

## 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 签名版本 2)

### 主题

- [文件上传 \(p. 549\)](#)
- [文本区域上传 \(p. 552\)](#)

### Note

本节中讨论的请求身份验证基于 AWS 签名版本 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 区域支持签名版本 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 - 签名版本 2。于 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。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 [示例：使用 HTTP POST 的基于浏览器的上传（使用 AWS 签名版本 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

下面的策略支持为 `johnsmith` 存储桶上传到 Amazon S3。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "johnsmith"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "http://johnsmith.s3.amazonaws.com/
successful_upload.html"},
    ["starts-with", "$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 johnsmith 存储桶。
- 键必须以“user/eric/”开头。
- ACL 必须设置为公共读取。
- 将 success\_action\_redirect 设置为http://johnsmith.s3.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。

```
eyAizXhwaxJhdGlvbiI6ICiyMDA3LTEyLTAXVDEyOjAwOjAwLjAwMfoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiA
```

使用您的凭证创建签名，例如 ORavWzkygo6QX9caELEqKi9kDbU= 是先前的策略文档的签名。

以下表单支持针对使用此策略的 johnsmith.net 存储桶的 POST 请求。

```
<html>
<head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
</head>
<body>
    ...
<form action="http://johnsmith.s3.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="http://
johnsmith.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" />
    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: johnsmith.s3.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"

http://johnsmith.s3.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"

eyAiZXhwaXJhdGlvbiI6IClEyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMfoILAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiA
--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: cxzFLJRatFH+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: http://johnsmith.s3.amazonaws.com/successful_upload.html?
bucket=johnsmith&key=user/eric/
MyPicture.jpg&etag="39d459dfbc0faabbb5e179358dfb94c3";
Server: AmazonS3
```

## 文本区域上传

### 主题

- [策略和表单构建 \(p. 552\)](#)
- [示例请求 \(p. 553\)](#)
- [示例响应 \(p. 554\)](#)

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 johnsmith 存储桶将文本区域上传到 Amazon S3。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "johnsmith"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "http://johnsmith.s3.amazonaws.com/new_post.html"},
    ["eq", "$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 johnsmith 存储桶。
- 键必须以“user/eric/”开头。
- ACL 必须设置为公共读取。
- 将 success\_action\_redirect 设置为 [http://johnsmith.s3.amazonaws.com/new\\_post.html](http://johnsmith.s3.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。

```
eyAiZXhwaxJhdGlvbiI6ICIyMDA3LTEyLTAXVDEyOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0iInOsCiAgICBbInN0YXJ0cy13aXR0iIwgIiRrZXkiLCAiadXNlcj9lcmljLyIAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3RpB25fcmVkaXJlY3QiOiAiaHR0cDovL2pvaG5zbWLc5zM5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXBiLCAiJENvbnnRlbnQtVHlwZSIiCJ0ZXh0L2h0bWwiXSwsKAQgIHSieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0InOsCiAgICBbInN0YXJ0cy13aXR0iIwgIiR4LWFtei1tZXRhLXRhISICIIxQogIF0KfQo=
```

使用您的凭证创建签名。例如 qA7FWXKq6VvU681I9KdveT1cWgF= 是先前的策略文档的签名。

以下表单支持针对使用此策略的 johnsmith.net 存储桶的 POST 请求。

```
<html>
<head>
  ...
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  ...

```

```
</head>
<body>
...
<form action="http://johnsmith.s3.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="http://
johnsmith.s3.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: johnsmith.s3.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"

user/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

http://johnsmith.s3.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"
eyAizXhwaXJhdGlvbiI6IClEyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQioiA

--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+NGtaDFRR8Yv19BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: http://johnsmith.s3.amazonaws.com/new_post.html?bucket=johnsmith&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

## POST 和 Adobe Flash

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，请在策略中包含下面的条件：

```
[ '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 对象](#)。有关表单字段的信息，请参阅 [HTML 表单字段 \(p. 543\)](#)。

# 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
<a href="#">Amazon Simple Storage Service 入门指南</a>	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。
<a href="#">Amazon Simple Storage Service API Reference</a>	“API 引用”详细介绍了 Amazon S3 操作。
<a href="#">Amazon S3 技术常见问题</a>	“常见问题解答”涵盖了开发人员对此产品提出的一些最热门的问题。
<a href="#">AWS 开发人员资源中心</a>	这是一个帮助您入门的资源整合点，您可以在这里找到相关的文档、代码示例、发布说明和其他信息，帮助您通过 AWS 构建创新的应用程序。
<a href="#">AWS 管理控制台</a>	该控制台可让您执行 Amazon S3 的大多数功能且无需编程。
<a href="https://forums.aws.amazon.com/">https://forums.aws.amazon.com/</a>	由开发人员组成的社区形式的论坛，在这里开发人员可以讨论与 AWS 有关的技术问题。
<a href="#">AWS 支持中心</a>	AWS 技术支持的首页包括对我们的开发人员论坛、技术 FAQ、服务状态和高级支持等页面。
<a href="#">AWS Premium Support</a>	提供有关 AWS Premium Support 信息的主要 Web 页面，它是一种一对一、快速响应的支持渠道，可帮助您在 AWS 基础设施服务上构建和运行应用程序。
<a href="#">Amazon S3 产品信息</a>	提供有关 Amazon S3 信息的主要 Web 页面。
<a href="#">联系我们</a>	这是一个有关查询的中心联络点，可帮助您查询有关 AWS 计费、AWS 账户、事件和滥用等方面的信息。
<a href="#">使用条件</a>	有关在 Amazon.com 和其他主题中的版权和商标使用的详细信息。

# 适用于 Amazon S3 Select 和 Amazon Glacier Select 的 SQL 参考

此参考包含 Amazon S3 Select 和 Amazon Glacier Select 支持的结构化查询语言 (SQL) 元素的描述。

## 主题

- [SELECT 命令 \(p. 557\)](#)
- [数据类型 \(p. 560\)](#)
- [运算符 \(p. 561\)](#)
- [保留关键字 \(p. 562\)](#)
- [SQL 函数 \(p. 566\)](#)

## SELECT 命令

Amazon S3 Select 和 Amazon Glacier Select 仅支持 SELECT SQL 命令。SELECT 支持以下 ANSI 标准子句：

- `SELECT list`
- `FROM 子句`
- `WHERE 子句`
- `LIMIT 子句 (仅限 Amazon S3 Select)`

### Note

Amazon S3 Select 和 Amazon Glacier Select 查询目前不支持子查询或联接。

## SELECT 列表

SELECT 列表指定希望查询返回的列、函数和表达式。列表表示查询的输出。

```
SELECT *
SELECT projection [ AS column_alias | column_alias ] [, ...]
```

第一个带 \* (星号) 的表单按原样返回每个传递 WHERE 子句的行。第二个表单使用用户定义的输出标量表达式投影为每列创建一行。

## FROM 子句

Amazon S3 Select 和 Amazon Glacier Select 支持以下形式的 FROM 子句：

```
FROM table_name
FROM table_name alias
FROM table_name AS alias
```

其中 *table\_name* 是 S3Object (适用于 Amazon S3 Select)、ARCHIVE 或 OBJECT (适用于 Amazon Glacier Select) 之一，指代正被查询的档案。对于来自传统关系数据库的用户，可以将其视为一个数据库架构，其中包含一个表的多个视图。

按照标准 SQL , FROM 子句将会创建在 WHERE 子句中筛选并在 SELECT 列表中投影的行。

## WHERE 子句

WHERE 子句遵循以下语法：

```
WHERE condition
```

WHERE 子句根据 condition 筛选行。condition 是具有布尔结果的表达式。只有 condition 计算结果为 TRUE 的行才会在结果中返回。

## LIMIT 子句 (仅限 Amazon S3 Select)

LIMIT 子句遵循以下语法：

```
LIMIT number
```

LIMIT 子句根据 number 限制您希望查询返回的记录数。

### Note

Amazon Glacier 不支持 LIMIT 子句。

## 属性访问

SELECT 和 WHERE 子句可以使用以下部分的任一方法引用记录数据，具体取决于将查询的文件是否采用 CSV 或 JSON 格式。

### CSV

- 列编号 – 您可以引用列名为 \_N 的行的第 N 列，其中 N 是列位置。位置计数从 1 开始。例如，第一列名为 \_1，第二列名为 \_2。

您可以通过 \_N 或 alias.\_N 引用列。例如，\_2 和 myAlias.\_2 都是有效的，都可用于引用 SELECT 列表中的列和 WHERE 子句。

- 列标头 – 对于具有标头行的 CSV 格式的对象，可在 SELECT 列表和 WHERE 子句中使用这些标头。尤其在传统 SQL 的 SELECT 和 WHERE 子句表达式中，可以通过 alias.column\_name 或 column\_name 引用列。

## JSON (仅限 Amazon S3 Select)

- 文档 – 可将 JSON 文档字段作为 alias.name 进行访问。还可以访问嵌套字段；例如，alias.name1.name2.name3。
- 列表 – 可以将零基索引和 [ ] 运算符配合使用以访问 JSON 列表中的元素。例如，可以通过 alias[1] 访问列表中第二个元素。可以结合字段以 alias.name1.name2[1].name3 的形式访问列表元素。
- 示例：将此 JSON 对象视为示例数据集：

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
[
    {"project_name": "project1", "completed": false},
```

```
        {"project_name": "project2", "completed": true}  
    ]  
}
```

示例 #1：以下查询返回这些结果：

```
Select s.name from S3Object s
```

```
{"name": "Susan Smith"}
```

示例 #2：以下查询返回这些结果：

```
Select s.projects[0].project_name from S3Object s
```

```
{"project_name": "project1"}
```

## 标头/属性名称区分大小写

借助 Amazon S3 Select 和 Amazon Glacier Select，您可以使用双引号指示区分大小写的列标头（对于 CSV 对象）和属性（对于 JSON 对象）。如果不是有双引号，对象标头/属性不区分大小写。如果出现歧义，则会抛出错误。

以下示例是：1) 包含指定列标头和 `FileInfo` 设置为“用于”查询请求的 CSV 格式的 Amazon S3 或 Amazon Glacier 对象；或 2) 包含指定属性的 JSON 格式的 Amazon S3 对象。

示例 #1：将查询的对象具有标头/属性“NAME”。

- 以下表达式从对象成功返回值（无引号：不区分大小写）：

```
SELECT s.name from S3Object s
```

- 以下表达式导致出现 400 错误 `MissingHeaderName`（包含引号：区分大小写）：

```
SELECT s."name" from S3Object s
```

示例 #2：将查询的 Amazon S3 对象具有一个包含“NAME”的标头/属性和另一个包含“name”的标头/属性。

- 以下表达式导致出现 400 错误 `AmbiguousFieldName`（无引号：不区分大小写，但有两个匹配项）：

```
SELECT s.name from S3Object s
```

- 以下表达式从对象成功返回值（引号：区分大小写，因此可以解析歧义）。

```
SELECT s."NAME" from S3Object s
```

## 将保留关键字作为用户定义的术语

Amazon S3 Select 和 Amazon Glacier Select 具有一组保留关键字，执行用于查询对象内容的 SQL 表达式时需要使用这组关键字。保留关键字包含函数名称、数据类型、运算符等。在某些情况下，诸如列标头（用于 CSV 文件）或属性（用于 JSON 对象）之类的用户定义的术语可能与保留关键字发生冲突。发生这种情况

时，必须使用双引号指示您特意使用与保留关键字冲突的用户定义的术语。否则将会导致出现 400 解析错误。

有关保留关键字的完整列表，请参阅[保留关键字 \(p. 562\)](#)。

以下示例是：1) 包含指定列标头和 `FileHeaderInfo` 设置为“用于”查询请求的 CSV 格式的 Amazon S3 或 Amazon Glacier 对象；或 2) 包含指定属性的 JSON 格式的 Amazon S3 对象。

示例：将查询的对象具有名为“CAST”的标头/属性，这是保留关键字。

- 以下表达式从对象成功返回值（引号：利用用户定义的标头/属性）：

```
SELECT s."CAST" from S3Object s
```

- 以下表达式导致出现 400 解析错误（无引号：与保留关键字发生冲突）：

```
SELECT s.CAST from S3Object s
```

## 标量表达式

在 `WHERE` 子句和 `SELECT` 列表中，您可以使用 SQL 标量表达式（返回标量值的表达式）。它们具有以下形式：

- literal

SQL 文本。

- column\_reference

以 `column_name` 或 `alias.column_name` 形式引用列。

- unary\_op expression

其中 `unary_op` 一元是一个 SQL 一元运算符。

- expression binary\_op expression

其中 `binary_op` 是一个 SQL 二元运算符。

- func\_name

其中 `func_name` 是要调用的标量函数的名称。

- expression [ NOT ] BETWEEN expression AND expression

- expression LIKE expression [ ESCAPE expression ]

## 数据类型

Amazon S3 Select 和 Amazon Glacier Select 支持多个基元数据类型。

## 数据类型转换

一般规则是遵循 `CAST` 函数（如果已定义）。如果未定义 `CAST`，则将所有输入数据视为字符串。必要时，必须将其转换为相关的数据类型。

有关 `CAST` 函数的更多信息，请参阅[CAST \(p. 568\)](#)。

## 受支持数据类型

Amazon S3 Select 和 Amazon Glacier Select 支持下面这组基元数据类型。

名称	描述	示例
bool	TRUE 或 FALSE	FALSE
INT、INTEGER	范围 -9,223,372,036,854,775,808 到 9,223,372,036,854,775,807 中的 8 字节有符号整数。	100000
字符串	UTF8 编码的长度可变的字符串。默认限制为一个字符。最大字符限制为 2,147,483,647。	'xyz'
float	8 字节浮点数。	CAST(0.456 AS FLOAT)
DECIMAL、NUMBER	base-10 数字，最大精度为 38 (即最大有效位数)，取值范围在 $-2^{31}$ 到 $2^{31}-1$ 之间 (即以 10 为底的指数)。	123.456
timestamp	时间戳表示特定的时刻，始终包括本地偏移，并且能够支持任意精度。  在文本格式中，时间戳遵循 <a href="#">日期和时间格式的 W3C 注释</a> ，但它们必须以文本“T”结尾 (如果不是最低限度全天精度)。允许使用小数秒，具有至少一位精度，以及无限的最大值。本地时间偏移可以表示为与 UTC 相比的小时:分钟偏移量，或表示为文本“Z”以指示 UTC 的本地时间。它们在具有时间的时间戳上是必需的，在日期值上是不允许的。	CAST('2007-04-05T14:30Z' AS TIMESTAMP)

## 运算符

Amazon S3 Select 和 Amazon Glacier Select 支持以下运算符。

### 逻辑运算符

- AND
- NOT
- OR

### 比较运算符

- <
- >
- <=
- >=
- =
- <>
- !=
- BETWEEN
- IN – 例如: IN ('a', 'b', 'c')

## 模式匹配运算符

- LIKE

## 数学运算符

支持加法、减法、乘法、除法和取模。

- +
- -
- \*
- %

## 运算符优先顺序

下表按降序显示运算符的优先顺序。

运算符/元素	关联性	必需
-	右	一元减法
*、/、%	左	乘法、除法和取模
+、-	左	加法、减法
IN		设置成员资格
BETWEEN		范围包含
LIKE		字符串模式匹配
<>		小于、大于
=	右	等于、分配
NOT	右	逻辑非
AND	左	逻辑和
OR	左	逻辑或

## 保留关键字

以下是适用于 Amazon S3 Select 和 Amazon Glacier Select 的保留关键字列表。这些包括功能名称、数据类型、运算符等，执行用于查询对象内容的 SQL 表达式时需要使用这些内容。

```
absolute
action
add
all
allocate
alter
```

```
and
any
are
as
asc
assertion
at
authorization
avg
bag
begin
between
bit
bit_length
blob
bool
boolean
both
by
cascade
cascaded
case
cast
catalog
char
char_length
character
character_length
check
clob
close
coalesce
collate
collation
column
commit
connect
connection
constraint
constraints
continue
convert
corresponding
count
create
cross
current
current_date
current_time
current_timestamp
current_user
cursor
date
day
deallocate
dec
decimal
declare
default
deferrable
deferred
delete
desc
describe
descriptor
diagnostics
```

```
disconnect
distinct
domain
double
drop
else
end
end-exec
escape
except
exception
exec
execute
exists
external
extract
false
fetch
first
float
for
foreign
found
from
full
get
global
go
goto
grant
group
having
hour
identity
immediate
in
indicator
initially
inner
input
insensitive
insert
int
integer
intersect
interval
into
is
isolation
join
key
language
last
leading
left
level
like
limit
list
local
lower
match
max
min
minute
missing
```

```
module
month
names
national
natural
nchar
next
no
not
null
nullif
numeric
octet_length
of
on
only
open
option
or
order
outer
output
overlaps
pad
partial
pivot
position
precision
prepare
preserve
primary
prior
privileges
procedure
public
read
real
references
relative
restrict
revoke
right
rollback
rows
schema
scroll
second
section
select
session
session_user
set
sexp
size
smallint
some
space
sql
sqlcode
sqlerror
sqlstate
string
struct
substring
sum
symbol
```

```
system_user
table
temporary
then
time
timestamp
timezone_hour
timezone_minute
to
trailing
transaction
translate
translation
trim
true
tuple
union
unique
unknown
unpivot
update
upper
usage
user
using
value
values
varchar
varying
view
when
whenever
where
with
work
write
year
zone
```

## SQL 函数

Amazon S3 Select 和 Amazon Glacier Select 支持多个 SQL 函数。

### 主题

- [聚合函数 \(仅限 Amazon S3 Select\) \(p. 566\)](#)
- [条件函数 \(p. 567\)](#)
- [转换函数 \(p. 568\)](#)
- [日期函数 \(p. 569\)](#)
- [字符串函数 \(p. 574\)](#)

## 聚合函数 (仅限 Amazon S3 Select)

Amazon S3 Select 支持以下聚合函数。

### Note

Amazon Glacier Select 不支持聚合函数。

函数	参数类型	返回类型
AVG(expression)	INT , FLOAT , DECIMAL	DECIMAL 适用于 INT 参数 , FLOAT 适用于浮点型参数 ; 否则与参数数据类型相同。
COUNT	-	INT
MAX(expression)	INT , DECIMAL	和参数类型相同。
MIN(expression)	INT , DECIMAL	和参数类型相同。
SUM(expression)	INT , FLOAT , DOUBLE , DECIMAL	INT 适用于 INT 参数 , FLOAT 适用于浮点型参数 ; 否则与参数数据类型相同。

## 条件函数

Amazon S3 Select 和 Amazon Glacier Select 支持以下条件函数。

### 主题

- [COALESCE \(p. 567\)](#)
- [NULLIF \(p. 568\)](#)

## COALESCE

按顺序评估参数并返回第一个非未知值，即第一个非空或非缺失值。此函数无法传播空值和缺失值。

### 语法

```
COALESCE ( expression, expression, ... )
```

### 参数

#### expression

对其执行函数的目标表达式。

### 示例

```
COALESCE(1)          -- 1
COALESCE(null)       -- null
COALESCE(null, null) -- null
COALESCE(missing)    -- null
COALESCE(missing, missing) -- null
COALESCE(1, null)    -- 1
```

```
| COALESCE(null, null, 1)      -- 1
| COALESCE(null, 'string')    -- 'string'
| COALESCE(missing, 1)        -- 1
```

## NULLIF

指定两个表达式，如果两个表达式的计算结果为相同值，则返回空；否则返回第一个表达式的计算结果。

### 语法

```
NULLIF ( expression1, expression2 )
```

### 参数

expression1, expression2

对其执行函数的目标表达式。

### 示例

```
NULLIF(1, 1)          -- null
NULLIF(1, 2)          -- 1
NULLIF(1.0, 1)        -- null
NULLIF(1, '1')        -- 1
NULLIF([1], [1])      -- null
NULLIF(1, NULL)       -- 1
NULLIF(NULL, 1)        -- null
NULLIF(null, null)     -- null
NULLIF(missing, null)   -- null
NULLIF(missing, missing) -- null
```

## 转换函数

Amazon S3 Select 和 Amazon Glacier Select 支持以下转换函数。

### 主题

- [CAST \(p. 568\)](#)

## CAST

CAST 函数将一种类型的实体（如计算结果为单个值的表达式）转换为另一种类型。

### 语法

```
CAST ( expression AS data_type )
```

### 参数

expression

一个或多个值、运算符和计算结果为值的 SQL 函数的组合。

data\_type

要将表达式转换到的目标数据类型，如 INT。有关支持的数据类型的列表，请参阅[数据类型 \(p. 560\)](#)。

## 示例

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

# 日期函数

Amazon S3 Select 和 Amazon Glacier Select 支持以下日期函数。

### 主题

- [DATE\\_ADD \(p. 569\)](#)
- [DATE\\_DIFF \(p. 570\)](#)
- [EXTRACT \(p. 570\)](#)
- [TO\\_STRING \(p. 571\)](#)
- [TO\\_TIMESTAMP \(p. 573\)](#)
- [UTCNOW \(p. 574\)](#)

## DATE\_ADD

指定日期部分、数量和时间戳，返回通过修改由数量的日期部分而更新的时间戳。

### 语法

```
DATE_ADD( date_part, quantity, timestamp )
```

### 参数

#### date\_part

指定要修改的日期部分。这可能是以下值之一：

- 年
- 月
- day
- 小时
- minute
- 秒

#### 数量

应用到已更新的时间戳的值。将数量的正值添加到时间戳的 date\_part，并减去负值。

#### timestamp

对其执行函数的目标时间戳。

## 示例

```
DATE_ADD(year, 5, `2010-01-01T`)
DATE_ADD(month, 1, `2010T`)
    necessary)
DATE_ADD(month, 13, `2010T`)
DATE_ADD(day, -1, `2017-01-10T`)
DATE_ADD(hour, 1, `2017T`)
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)
-- 2015-01-01 (equivalent to 2015-01-01T)
-- 2010-02T (result will add precision as
-- necessary)
-- 2011-02T
-- 2017-01-09 (equivalent to 2017-01-09T)
-- 2017-01-01T01:00-00:00
-- 2017-01-02T04:04Z
```

```
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

## DATE\_DIFF

指定日期部分和两个有效的时间戳，返回日期部分的不同之处。当 timestamp1 的 date\_part 值大于 timestamp2 的 date\_part 值时，返回值为负整数。当 timestamp1 的 date\_part 值小于 timestamp2 的 date\_part 值时，返回值为正整数。

### 语法

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

### 参数

date\_part

指定要比较的时间戳部分。有关 date\_part 的定义，请参阅[DATE\\_ADD \(p. 569\)](#)。

timestamp1

要比较的第一个时间戳。

timestamp2

要比较的第二个时间戳。

### 示例

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)          -- 1
DATE_DIFF(year, `2010T`, `2010-05T`)                   -- 4 (2010T is equivalent to
2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                    -- 12
DATE_DIFF(month, `2011T`, `2010T`)                    -- -12
DATE_DIFF(day, `2010-01-01T23:00T`, `2010-01-02T01:00T`) -- 0 (need to be at least 24h
apart to be 1 day apart)
```

## EXTRACT

指定部分日期和时间戳，返回时间戳日期部分的值。

### 语法

```
EXTRACT( date_part FROM timestamp )
```

### 参数

date\_part

指定要提取的时间戳部分。这可能是以下值之一：

- 年
- 月
- day
- 小时
- minute
- 秒

- `timezone_hour`
  - `timezone_minute`
- `timestamp`

对其执行函数的目标时间戳。

## 示例

```
EXTRACT(YEAR FROM `2010-01-01T`)  
EXTRACT(MONTH FROM `2010T`)  
2010-01-01T00:00:00.000Z) -- 2010  
-- 1 (equivalent to  
EXTRACT(MONTH FROM `2010-10T`)  
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)  
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)  
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)  
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)
```

## TO\_STRING

指定时间戳和格式模式，以指定格式返回时间戳的字符串表示形式。

### 语法

```
TO_STRING ( timestamp time_format_pattern )
```

### 参数

`timestamp`

对其执行函数的目标时间戳。

`time_format_pattern`

具有以下特殊字符解释的字符串。

格式	示例	描述
<code>yy</code>	69	2 位年份
<code>y</code>	1969	4 位年份
<code>yyyy</code>	1969	以零填充的 4 位年份
<code>M</code>	1	一年中的月
<code>MM</code>	01	以零填充的一年中的月份
<code>MMM</code>	Jan	一年中的月的缩写名称
<code>MMMM</code>	January	一年中的月的完整名称
<code>MMMMM</code>	J	一年中的月的首字母 (注释：与 to_timestamp 函数相同)

格式	示例	描述
		数结合使用时无效)
d	2	一个月中的日(1-31)
dd	02	以零填充的一个月中的日(01-31)
a	AM	一天的 AM 或 PM
h	3	一天中的小时(1-12)
hh	03	以零填充的一天中的小时(01-12)
H	3	一天中的小时(0-23)
HH	03	以零填充的一天中的小时(00-23)
m	4	一小时中的分钟(0-59)
mm	04	以零填充的一小时中的分钟(00-59)
s	5	一分钟中的秒(0-59)
ss	05	以零填充的一分钟中的秒(00-59)
S	0	一秒钟的若干分之几(精度：0.1，范围：0.0-0.9)
SS	6	一秒钟的若干分之几(精度：0.01，范围：0.0-0.99)
SSS	60	一秒钟的若干分之几(精度：0.001，范围：0.0-0.999)
...	...	...

格式	示例	描述
SSSSSSSS	60000000	一秒钟的若干分之几(最大精度：1 纳秒，范围：0.0-0.999999999)
n	60000000	纳秒级
X	+07 or Z	小时数偏移，如果偏移为 0，则为“Z”
XX or XXXX	+0700 or Z	小时和分钟数偏移，如果偏移为 0，则为“Z”
XXX or XXXXX	+07:00 or Z	小时和分钟数偏移，如果偏移为 0，则为“Z”
x	7	小时数偏移
xx or xxxx	700	小时和分钟数偏移
xxx or xxxxx	+07:00	小时和分钟数偏移

## 示例

```

TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y')
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')
TO_STRING(`1969-07-20T20:18Z`, 'MMMN d, y h:m a')
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')
"1969-07-20T20:18:00+0800"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX') --
"1969-07-20T20:18:00+08:00"

```

## TO\_TIMESTAMP

指定字符串，将其转换为时间戳。这是 TO\_STRING 的逆运算。

### 语法

```
TO_TIMESTAMP ( string )
```

### 参数

string

对其执行函数的目标字符串。

## 示例

```
TO_TIMESTAMP('2007T') -- `2007T`  
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

## UTCNOW

以时间戳的形式返回当前时间 (UTC)。

### 语法

```
UTCNOW()
```

### 参数

无

### 示例

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

## 字符串函数

Amazon S3 Select 和 Amazon Glacier Select 支持以下字符串函数。

### 主题

- [CHAR\\_LENGTH , CHARACTER\\_LENGTH \(p. 574\)](#)
- [LOWER \(p. 575\)](#)
- [SUBSTRING \(p. 575\)](#)
- [TRIM \(p. 576\)](#)
- [UPPER \(p. 576\)](#)

## CHAR\_LENGTH , CHARACTER\_LENGTH

计算指定字符串中的字符数。

### Note

CHAR\_LENGTH 和 CHARACTER\_LENGTH 是同义词。

### 语法

```
CHAR_LENGTH ( string )
```

### 参数

string

对其执行函数的目标字符串。

## 示例

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

## LOWER

指定字符串，将所有大写字符转换为小写字符。所有非大写字符保持不变。

### 语法

```
LOWER ( string )
```

### 参数

string

对其执行函数的目标字符串。

## 示例

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

## SUBSTRING

指定字符串、起始索引和长度（可选），返回从起始索引到字符串结尾处的子字符串，或最大长度为输入的字符串长度的子字符串。

### Note

输入字符串的首字符索引为 1。如果 start 为 < 1，则将其设置为 1。

### 语法

```
SUBSTRING( string FROM start [ FOR length ] )
```

### 参数

string

对其执行函数的目标字符串。

start

字符串的开始位置。

length

要返回的子字符串的长度。如果不存在，则执行到字符串的结尾。

## 示例

```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
```

```
SUBSTRING("123456789", 2)      -- "23456789"  
SUBSTRING("123456789", -4)     -- "123456789"  
SUBSTRING("123456789", 0, 999)  -- "123456789"  
SUBSTRING("123456789", 1, 5)    -- "12345"
```

## TRIM

从字符串中剪裁前导或尾随字符。要删除的默认字符为 ''。

### 语法

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

### 参数

string

对其执行函数的目标字符串。

LEADING | TRAILING | BOTH

是否剪裁前导和/或尾随字符。

remove\_chars

要删除的一组字符。请注意，remove\_chars 可能是长度 > 1 的字符串。此函数返回包含在已删除字符串开头或结尾发现的 remove\_chars 中任何字符的字符串。

### 示例

```
TRIM('      foobar      ')          -- 'foobar'  
TRIM('      \tfoobar\t      ')       -- '\tfoobar\t'  
TRIM(LEADING FROM '      foobar      ') -- 'foobar      '  
TRIM(TRAILING FROM '      foobar      ') -- '      foobar'  
TRIM(BOTH FROM '      foobar      ')  -- 'foobar'  
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

## UPPER

指定字符串，将所有小写字符转换为大写字符。所有非小写字符保持不变。

### 语法

```
UPPER ( string )
```

### 参数

string

对其执行函数的目标字符串。

### 示例

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

# 文档历史记录

- 上次文档更新日期：2018 年 19 月 6 日
- 当前 API 版本：2006-03-01

下表介绍了 2018 年 6 月 19 日之后每个 Amazon Simple Storage Service 开发人员指南 发布版中的重要更改。如需对此文档更新的通知，您可以订阅 RSS 源。

update-history-change	update-history-description	update-history-date
<a href="#">现在可通过 RSS 更新 (p. 577)</a>	您现在可以订阅 RSS 源来接收有关 Amazon Simple Storage Service 开发人员指南更新的通知。	June 19, 2018

## 早期更新

下表描述了 2018 年 6 月 19 日之前每个 Amazon Simple Storage Service 开发人员指南 发布版中的重要更改。

更改	描述	日期
代码示例更新	<p>更新了代码示例：</p> <ul style="list-style-type: none"><li>C# — 已将所有示例更新为使用基于任务的异步模式。有关更多信息，请参阅适用于 .NET 的 AWS 开发工具包开发人员指南 中的<a href="#">适用于 .NET 的 Amazon Web Services 异步 API</a>。代码示例现在符合版本 3 的适用于 .NET 的 AWS 开发工具包的要求。</li><li>Java — 已将所有示例更新为使用客户端生成器模型。有关客户端生成器模型的更多信息，请参阅<a href="#">创建服务客户端</a>。</li><li>PHP — 已将所有示例更新为使用适用于 PHP 的 AWS 开发工具包 3.0。有关适用于 PHP 的 AWS 开发工具包 3.0 的更多信息，请参阅<a href="#">适用于 PHP 的 AWS 开发工具包</a>。</li><li>Ruby — 已更新示例代码，以便让示例适用于适用于 Ruby 的 AWS 开发工具包版本 3。</li></ul>	2018 年 30 月 4 日
Amazon S3 现在将 GLACIER 和 ONEZONE_IA 存储类报告到 Amazon CloudWatch Logs 存储指标	<p>除了报告实际字节数量之外，这些存储指标还包含合适的存储类 (ONEZONE_IA、STANDARD_IA 和 GLACIER) 的每对象开销字节数：</p> <ul style="list-style-type: none"><li>对于 ONEZONE_IA 和 STANDARD_IA 存储类对象，Amazon S3 将小于 128 KB 的对象报告为 128 KB。有关更多信息，请参阅<a href="#">存储类别 (p. 90)</a>。</li><li>对于 GLACIER 存储类对象，存储指标将报告以下开销：<ul style="list-style-type: none"><li>一个 32 KB 每对象开销，按 GLACIER 存储类定价收费</li><li>一个 8 KB 每对象开销，按 STANDARD 存储类定价收费</li></ul></li></ul> <p>有关更多信息，请参阅<a href="#">转换对象 (p. 105)</a>。</p>	2018 年 30 月 4 日

更改	描述	日期
	有关存储指标的更多信息，请参阅 <a href="#">使用 Amazon CloudWatch 来监控指标 (p. 482)</a> 。	
新存储类别	Amazon S3 现在提供了用于存储对象的新存储类 ONEZONE_IA (IA，适用于不常访问)。有关更多信息，请参阅 <a href="#">存储类别 (p. 90)</a> 。	2018 年 4 月 4 日
Amazon S3 Select	Amazon S3 现在支持根据 SQL 表达式检索对象内容。有关更多信息，请参阅 <a href="#">从对象中选择内容 (p. 220)</a> 。	2018 年 4 月 4 日
亚太区域 (大阪当地)	Amazon S3 目前在 亚太区域 (大阪当地) 区域中可用。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。  <b>Important</b>  亚太区域 (大阪当地) 区域只能与 亚太区域 (东京) 区域结合使用。要请求访问 亚太区域 (大阪当地) 区域，请联系您的销售代表。	2018 年 2 月 12 日
Amazon S3 清单创建时间戳	Amazon S3 清单现在包括创建 Amazon S3 清单报告的日期和开始时间的时间戳。可以使用此时间戳确定开始生成清单报告的时间之后 Amazon S3 存储中的变化。	2018 年 1 月 16 日
欧洲 (巴黎) 区域	Amazon S3 在 欧洲 (巴黎) 区域中现已可用。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。	2017 年 12 月 18 日
中国 (宁夏) 区域	Amazon S3 目前已在中国 (宁夏) 区域中提供。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。	2017 年 12 月 11 日
使用 SQL 查询档案	Amazon S3 现在支持使用 SQL 查询 Amazon Glacier 数据档案。有关更多信息，请参阅 <a href="#">查询存档对象 (p. 226)</a> 。	2017 年 11 月 29 日
ORC 格式 Amazon S3 清单文件的支持	对于清单输出文件，除了逗号分隔值 (CSV) 文件格式之外，Amazon S3 现在还支持 <a href="#">Apache 优化行列式 (ORC)</a> 格式。此外，您现在可通过 Amazon Athena、Amazon Redshift Spectrum 和其他工具 (例如 <a href="#">Presto</a> 、 <a href="#">Apache Hive</a> 和 <a href="#">Apache Spark</a> ) 使用标准 SQL 查询 Amazon S3 清单。有关更多信息，请参阅 <a href="#">Amazon S3 清单 (p. 235)</a> 。	2017 年 11 月 17 日
S3 存储桶的默认加密	Amazon S3 默认加密提供了一种方法来设置 S3 存储桶的默认加密行为。您可以对存储桶设置默认加密，以便在存储桶中存储所有对象时对这些对象进行加密。这些对象使用 Amazon S3 托管密钥 (SSE-S3) 或 AWS KMS 托管密钥 (SSE-KMS) 通过服务器端加密进行加密。有关更多信息，请参阅 <a href="#">S3 存储桶的 Amazon S3 默认加密 (p. 61)</a> 。	2017 年 11 月 06 日
Amazon S3 清单中的加密状态	Amazon S3 现在的支持包括 Amazon S3 清单中的加密状态，因此您查看对象在静态时如何加密，以用于合规性审计或其他用途。您还可以配置使用服务器端加密 (SSE) 或 SSE-KMS 来加密 S3 清单，从而相应地加密所有清单文件。有关更多信息，请参阅 <a href="#">Amazon S3 清单 (p. 235)</a> 。	2017 年 11 月 06 日

更改	描述	日期
跨区域复制 (CRR) 增强功能	<p>跨区域复制现在支持以下功能：</p> <ul style="list-style-type: none"> <li>在跨账户方案中，您可以添加 CRR 配置，将副本所有权更改为拥有目标存储桶的 AWS 账户。有关更多信息，请参阅 <a href="#">跨区域复制的其他配置：更改副本拥有者 (p. 451)</a>。</li> <li>默认情况下，Amazon S3 不复制您的源存储桶中使用 AWS KMS 托管密钥通过服务器端加密创建的对象。在您的 CRR 配置中，您现在可以指示 Amazon S3 复制这些对象。有关更多信息，请参阅 <a href="#">CRR 其他配置：复制使用 AWS KMS 托管加密密钥通过服务器端加密 (SSE) 创建的对象 (p. 453)</a>。</li> </ul>	2017 年 11 月 06 日
欧洲 (伦敦) 区域	Amazon S3 在欧洲 (伦敦) 区域中现已可用。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。	2016 年 12 月 13 日
加拿大 (中部) 区域	Amazon S3 在加拿大 (中部) 区域现已可用。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。	2016 年 12 月 8 日
对象标签	<p>Amazon S3 现在支持对象标签。利用对象标签，您可以对存储进行分类。对象键名称前缀还可让您对存储进行分类，对象标签将向其添加另一个维度。</p> <p>添加标签还可提供其他好处。包括：</p> <ul style="list-style-type: none"> <li>对象标签支持权限的精细访问控制（例如，您可以向 IAM 用户授予对具有特定标签的只读对象的权限）。</li> <li>指定生命周期配置时的精细控制。您可以指定标签以选择生命周期规则应用于的对象子集。</li> <li>如果您已配置跨区域复制 (CRR)，则 Amazon S3 可以复制标签。您必须向为 Amazon S3 创建的要代入的 IAM 角色授予代表您复制对象的权限。</li> <li>您还可以自定义 CloudWatch 指标和 CloudTrail 事件以按特定标签筛选条件显示信息。</li> </ul> <p>有关更多信息，请参阅 <a href="#">对象标签 (p. 96)</a>。</p>	2016 年 11 月 29 日
S3 生命周期现在支持基于标签的筛选	Amazon S3 现在支持生命周期配置中的基于标签的筛选。您现在可以指定生命周期规则，在该规则下，您可以指定键前缀、一个或多个对象标签或二者的组合以选择生命周期规则将应用于的对象子集。有关更多信息，请参阅 <a href="#">对象生命周期管理 (p. 104)</a> 。	2016 年 11 月 29 日
存储桶的 CloudWatch 请求指标	Amazon S3 现在支持在存储桶上发出的请求的 CloudWatch 指标。当您为存储桶启用这些指标时，指标每分钟报告一次。您还可以配置存储桶中的哪些对象将报告这些请求指标。有关更多信息，请参阅 <a href="#">使用 Amazon CloudWatch 来监控指标 (p. 482)</a> 。	2016 年 11 月 29 日
Amazon S3 清单	<p>Amazon S3 现在支持存储清单。Amazon S3 清单每天或每周为 S3 存储桶或共享前缀（即，其名称以通用字符串开头的对象）提供对象及其对应元数据的平面文件输出。</p> <p>有关更多信息，请参阅 <a href="#">Amazon S3 清单 (p. 235)</a>。</p>	2016 年 11 月 29 日

更改	描述	日期
Amazon S3 分析 - 存储类分析	<p>新的 Amazon S3 分析 - 存储类分析功能可观察数据访问模式以帮助您确定何时将不常访问的 STANDARD 存储转换为 STANDARD_IA (IA 表示不常访问) 存储类。在存储类分析观察一组筛选出的数据的不常访问模式一段时间后，您可以使用分析结果来帮助改进您的生命周期策略。此功能还包括在指定的存储桶、前缀或标记级别对存储使用情况的详细日常分析，并且您可以将分析结果导出到 S3 存储桶。</p> <p>有关更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的 <a href="#">Amazon S3 分析 - 存储类分析 (p. 230)</a>。</p>	2016 年 11 月 29 日
从 Amazon Glacier 还原存档对象时新建加速和批量数据检索	在将存档对象还原至 Amazon Glacier 时，Amazon S3 现在支持加速和批量数据检索以及标准检索。有关更多信息，请参阅 <a href="#">恢复存档对象 (p. 223)</a> 。	2016 年 11 月 21 日
CloudTrail 对象日志记录	CloudTrail 支持日志记录 Amazon S3 对象级别 API 操作，例如 GetObject、PutObject 和 DeleteObject。您可以将事件选择器配置为记录对象级别 API 操作。有关更多信息，请参阅 <a href="#">使用 AWS CloudTrail 记录 Amazon S3 API 调用 (p. 488)</a> 。	2016 年 11 月 21 日
美国东部 (俄亥俄) 区域	Amazon S3 现已在 美国东部 (俄亥俄州) 区域可用。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。	2016 年 10 月 17 日
面向 Amazon S3 Transfer Acceleration 的 IPv6 支持	Amazon S3 现支持面向 Amazon S3 Transfer Acceleration 的 Internet 协议版本 6 (IPv6)。您可以使用全新的 Transfer Acceleration 双堆栈终端节点通过 IPv6 连接至 Amazon S3。有关更多信息，请参阅 <a href="#">Amazon S3 Transfer Acceleration 入门 (p. 68)</a> 。	2016 年 10 月 6 日
IPv6 支持	Amazon S3 现在支持 Internet 协议版本 6 (IPv6)。您可以使用双堆栈终端节点通过 IPv6 访问 Amazon S3。有关更多信息，请参阅 <a href="#">通过 IPv6 向 Amazon S3 发出请求 (p. 10)</a> 。	2016 年 8 月 11 日
亚太地区 (孟买) 区域	Amazon S3 现已在 亚太地区 (孟买) 区域可用。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。	2016 年 6 月 27 日
Amazon S3 Transfer Acceleration	<p>Amazon S3 Transfer Acceleration 可在客户与 S3 存储桶之间实现快速、轻松、安全的远距离文件传输。Transfer Acceleration 利用 Amazon CloudFront 的全球分布式边缘站点。</p> <p>有关更多信息，请参阅 <a href="#">Amazon S3 Transfer Acceleration (p. 67)</a>。</p>	2016 年 4 月 19 日
生命周期支持移除过期对象删除标记	生命周期配置 <code>Expiration</code> 操作现在允许您指示 Amazon S3 移除受版本控制的存储桶中的过期对象删除标记。有关更多信息，请参阅 <a href="#">用于描述生命周期操作的元素 (p. 112)</a> 。	2016 年 16 月 3 日

更改	描述	日期
存储桶生命周期配置现在支持中止未完成的分段上传的操作	<p>存储桶生命周期配置现在支持 <code>AbortIncompleteMultipartUpload</code> 操作，您可以使用该操作指示 Amazon S3 中止未在启动后的指定天数内完成的分段上传。当分段上传符合中止操作条件时，Amazon S3 将删除所有已上传的段并中止分段上传。</p> <p>有关概念性信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的以下主题：</p> <ul style="list-style-type: none"> <li>• <a href="#">使用存储桶生命周期策略中止未完成的分段上传 (p. 156)</a></li> <li>• <a href="#">用于描述生命周期操作的元素 (p. 112)</a></li> </ul> <p>以下 API 操作已更新为支持新操作：</p> <ul style="list-style-type: none"> <li>• <b>PUT 存储桶生命周期 - XML 配置</b>现在允许您在生命周期配置规则中指定 <code>AbortIncompleteMultipartUpload</code> 操作。</li> <li>• <b>列出段和启动分段上传 - 如果存储桶包含了指定 <code>AbortIncompleteMultipartUpload</code> 操作的生命周期规则，则这两个 API 操作现在将返回两个额外的响应标头 (<code>x-amz-abort-date</code> 和 <code>x-amz-abort-rule-id</code>)。</b> 响应中的这些标头指示启动的分段上传何时符合中止操作的条件以及适用哪个生命周期规则。</li> </ul>	2016 年 16 月 3 日
亚太区域（首尔）区域	Amazon S3 目前在 亚太区域（首尔）区域中可用。有关 Amazon S3 区域和终端节点的更多信息，请参阅 AWS General Reference 中的 <a href="#">区域和终端节点</a> 。	2016 年 1 月 6 日
新的条件键和分段上传变更	<p>IAM 策略现在支持 Amazon S3 <code>s3:x-amz-storage-class</code> 条件键。有关更多信息，请参阅 <a href="#">在策略中指定条件 (p. 286)</a>。</p> <p>您不再需要作为分段上传的发起者就能上传分段和完成上传。有关更多信息，请参阅 <a href="#">分段上传 API 和权限 (p. 159)</a>。</p>	2015 年 12 月 14 日
重命名美国标准区域	区域名称字符串从“美国标准”变更为“美国东部（弗吉尼亚北部）”。此次只更新了区域名称，功能上没有任何变化。	2015 年 12 月 11 日
新存储类别	<p>Amazon S3 现在为存储对象提供了新存储类别 <code>STANDARD_IA</code> (<code>IA</code>，适用于不常访问)。此存储类别已针对长时间运行且不常访问的数据进行了优化。有关更多信息，请参阅 <a href="#">存储类别 (p. 90)</a>。</p> <p>生命周期配置功能的更新现在可允许您将对象转换为 <code>STANDARD_IA</code> 存储类别。有关更多信息，请参阅 <a href="#">对象生命周期管理 (p. 104)</a>。</p> <p>以前，跨区域复制功能对对象副本使用了源对象的存储类别。现在，当您配置跨区域复制时，您可以在目标存储桶中为创建的对象副本指定存储类别。有关更多信息，请参阅 <a href="#">跨区域复制 (CR) (p. 442)</a>。</p>	2015 年 9 月 16 日
AWS CloudTrail 集成	新的 AWS CloudTrail 集成能够记录您的 S3 存储桶中发生的 Amazon S3 API 活动。您可以使用 CloudTrail 来跟踪 S3 存储桶的创建或删除、访问控制修改、生命周期策略变更等。有关更多信息，请参阅 <a href="#">使用 AWS CloudTrail 记录 Amazon S3 API 调用 (p. 488)</a> 。	2015 年 9 月 1 日

更改	描述	日期
存储桶限额提升	Amazon S3 现在支持存储桶限额提升。默认情况下，客户可在其 AWS 账户中创建多达 100 个存储桶。需要更多存储桶的客户可通过提交服务限额提升来调高限制。有关如何调高存储桶限制的信息，请转到 AWS 一般参考 中的 <a href="#">AWS 服务限制</a> 。有关更多信息，请参阅 <a href="#">创建存储桶 (p. 49)</a> 和 <a href="#">存储桶限制 (p. 53)</a> 。	2015 年 8 月 4 日
一致性模型更新	Amazon S3 现在在美国东部 (弗吉尼亚北部) 区域对添加到 Amazon S3 的新对象支持先写后读一致性。在此更新之前，除美国东部 (弗吉尼亚北部) 区域之外的所有区域都对上传到 Amazon S3 的新对象支持先写后读一致性。利用此改进，Amazon S3 现在在所有区域对添加到 Amazon S3 的新对象都支持先写后读一致性。先写后读一致性允许您在 Amazon S3 中创建对象后立即检索对象。有关更多信息，请参阅 <a href="#">区域 (p. 3)</a> 。	2015 年 8 月 4 日
事件通知	Amazon S3 事件通知进行了更新，以添加在删除对象时发送的通知并添加通过前缀和后缀匹配进行对象名称筛选的功能。有关更多信息，请参阅 <a href="#">配置 Amazon S3 事件通知 (p. 425)</a> 。	2015 年 7 月 28 日
Amazon CloudWatch 集成	利用新的 Amazon CloudWatch 集成，您可以通过 Amazon S3 的 CloudWatch 指标监控 Amazon S3 使用情况并设置警报。支持的指标包含标准存储的总字节数、低冗余存储的总字节数以及给定 S3 存储桶的对象的总数。有关更多信息，请参阅 <a href="#">使用 Amazon CloudWatch 来监控指标 (p. 482)</a> 。	2015 年 7 月 28 日
支持删除和清空非空存储桶	Amazon S3 现在支持删除和清空非空存储桶。有关更多信息，请参阅 <a href="#">删除或清空存储桶 (p. 57)</a> 。	2015 年 7 月 16 日
Amazon VPC 终端节点的存储桶策略	Amazon S3 添加了对 Amazon Virtual Private Cloud (Amazon VPC) 终端节点的存储桶策略的支持。您可以使用 S3 存储桶策略控制从特定 Amazon VPC 终端节点或特定 VPC 对存储桶的访问。VPC 终端节点易于配置，高度可靠，并且提供到 Amazon S3 的安全连接而无需网关或 NAT 实例。有关更多信息，请参阅 <a href="#">Amazon S3 的 VPC 终端节点的示例存储桶策略 (p. 310)</a> 。	2015 年 4 月 29 日
事件通知	已更新 Amazon S3 事件通知，以支持切换到 AWS Lambda 函数的基于资源的权限。有关更多信息，请参阅 <a href="#">配置 Amazon S3 事件通知 (p. 425)</a> 。	2015 年 4 月 9 日
跨区域复制	Amazon S3 现在支持跨区域复制。跨区域复制是跨不同 AWS 区域中的存储桶自动、异步地复制对象。有关更多信息，请参阅 <a href="#">跨区域复制 (CRR) (p. 442)</a> 。	2015 年 3 月 24 日
事件通知	Amazon S3 现在支持存储桶通知配置中的新事件类型和目标。在此版本之前，Amazon S3 仅支持 s3:ReducedRedundancyLostObject 事件类型和 Amazon SNS 主题作为目标。有关新事件类型的更多信息，请参阅 <a href="#">配置 Amazon S3 事件通知 (p. 425)</a> 。	2014 年 11 月 13 日

更改	描述	日期
使用客户提供的加密密钥的服务器端加密	<p>使用 AWS Key Management Service (AWS KMS) 的服务器端加密</p> <p>Amazon S3 现在支持使用 AWS Key Management Service 的服务器端加密。此功能允许您通过 AWS KMS 管理信封密钥，Amazon S3 使用您设置的权限调用 AWS KMS 以访问信封密钥。</p> <p>有关使用 AWS KMS 的服务器端加密的更多信息，请参阅<a href="#">通过 AWS Key Management Service 使用服务器端加密来保护数据</a>。</p>	2014 年 11 月 12 日
欧洲（法兰克福）区域	Amazon S3 目前在欧洲（法兰克福）区域可用。	2014 年 10 月 23 日
使用客户提供的加密密钥的服务器端加密	<p>Amazon S3 现在支持使用客户提供的加密密钥的服务器端加密 (SSE-C)。服务器端加密使您能够请求 Amazon S3 加密您的静态数据。在使用 SSE-C 时，Amazon S3 将使用您提供的自定义加密密钥加密您的对象。由于 Amazon S3 会自动执行加密，因此您可以使用自己的加密密钥，而无需编写或执行自己的加密代码。</p> <p>有关 SSE-C 的更多信息，请参阅<a href="#">服务器端加密 (使用客户提供的加密密钥)</a>。</p>	2014 年 6 月 12 日
版本控制的生命周期支持	在此版本之前，仅对不受版本控制的存储桶支持生命周期配置。现在您对不受版本控制和启用了版本控制的存储桶都可以配置生命周期。有关更多信息，请参阅 <a href="#">对象生命周期管理 (p. 104)</a> 。	2014 年 5 月 20 日
修订了访问控制主题	修订了 Amazon S3 访问控制文档。有关更多信息，请参阅 <a href="#">管理对 Amazon S3 资源的访问权限 (p. 242)</a> 。	2014 年 4 月 15 日
修订了服务器访问日志记录主题	修订了服务器访问日志记录文档。有关更多信息，请参阅 <a href="#">Amazon S3 服务器访问日志记录 (p. 506)</a> 。	2013 年 11 月 26 日
.NET 开发工具包示例已更新为版本 2.0	本指南中的 .NET 开发工具包示例现在符合版本 2.0 标准。	2013 年 11 月 26 日
HTTP 上的 SOAP 支持已弃用	HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。不支持将新 Amazon S3 功能用于 SOAP。我们建议您使用 REST API 或 AWS 开发工具包。	2013 年 9 月 20 日
IAM 策略变量支持	<p>IAM 访问策略语言现在支持变量。在评估策略时，任何策略变量都会替换为来自经身份验证的用户会话的上下文相关信息提供的值。您可以使用策略变量来定义通用策略，无需显式列出策略的所有部分。有关策略变量的更多信息，请参阅 IAM 用户指南 中的<a href="#">IAM 策略变量概述</a>。</p> <p>有关 Amazon S3 中的策略变量的示例，请参阅<a href="#">用户策略示例 (p. 312)</a>。</p>	2013 年 4 月 3 日
对申请方付款的控制台支持	现在，您可以使用 Amazon S3 控制台为申请方付款配置存储桶。有关更多信息，请参阅 <a href="#">使用 Amazon S3 控制台配置申请方付款 (p. 74)</a> 。	2012 年 12 月 31 日

更改	描述	日期
对网站托管的根域支持	Amazon S3 现已支持在根域中托管静态网站。网站访问者在浏览器上访问网站时无需在 Web 地址中指定“www”(如“example.com”)。许多客户已在 Amazon S3 上托管了静态网站，访问者可通过“www”子域(如“www.example.com”)访问这些网站。此前，要支持根域网访问，您必须在运行自己的 Web 服务器时将根域请求从浏览器以代理的方式转至您在 Amazon S3 上的网站。运行 Web 服务器对请求进行代理会导致出现额外成本、运转负载和其他潜在的故障点。现在，对于“www”和根域地址，您都可以充分享受 Amazon S3 带来的高可用性和持久性了。有关更多信息，请参阅 <a href="#">在 Amazon S3 上托管静态网站 (p. 401)</a> 。	2012 年 12 月 27 日
控制台修改	对 Amazon S3 控制台进行了更新。并对与控制台相关的文档主题做出了相应修改。	2012 年 12 月 14 日
支持将对象存档到 Amazon Glacier	Amazon S3 现已支持的存储选项可让您使用 Amazon Glacier 的低成本存储服务来存档数据。要存档对象，您需要定义存档规则来识别对象以及希望 Amazon S3 据此将这些对象存档到 Amazon Glacier 的时间表。通过使用 Amazon S3 控制台或者以编程方式使用 Amazon S3 API 或 AWS SDK，您可以轻松地在存储桶上设置规则。  有关更多信息，请参阅 <a href="#">对象生命周期管理 (p. 104)</a> 。	2012 年 11 月 13 日
对网页重定向的支持	对于配置为网站的存储桶，Amazon S3 现在支持将对象的请求重定向到相同存储桶中的另一个对象或外部 URL。有关更多信息，请参阅 <a href="#">(可选) 配置网页重定向 (p. 408)</a> 。  有关托管网站的信息，请参阅 <a href="#">在 Amazon S3 上托管静态网站 (p. 401)</a> 。	2012 年 10 月 4 日
对跨域资源共享 (CORS) 的支持	Amazon S3 现已支持跨域资源共享 (CORS) 功能。一个域中加载的客户端 Web 应用程序可借助 CORS 定义的方式与不同域中的资源进行交互，或访问这些资源。Amazon S3 中的 CORS 支持让您可以在 Amazon S3 之上构建丰富的客户端 Web 应用程序，同时让您可以选择性地允许跨域访问您的 Amazon S3 资源。有关更多信息，请参阅 <a href="#">跨源资源共享 (CORS) (p. 133)</a> 。	2012 年 8 月 31 日
对成本分配标记的支持	Amazon S3 现在支持成本分配标记，这允许您为 S3 存储桶加标签，从而可以更轻松地根据项目或其他条件跟踪其成本。有关将标记用于存储桶的更多信息，请参阅 <a href="#">使用成本分配 S3 存储桶标签 (p. 84)</a> 。	2012 年 8 月 21 日
对存储桶策略中受 MFA 保护的 API 访问的支持	Amazon S3 现在支持受 MFA 保护的 API 访问，这项功能在访问 Amazon S3 资源时可强制执行 AWS Multi-Factor Authentication 以提供更高级别的安全。它是一项安全功能，要求用户通过提供有效 MFA 代码来证明实际拥有 MFA 设备。有关详细信息，请参阅 <a href="#">AWS Multi-Factor Authentication</a> 。现在，您可以要求对任何访问 Amazon S3 资源的请求进行 MFA 身份验证。  为了强制执行 MFA 身份验证，Amazon S3 现在支持存储桶策略中的 <code>aws:MultiFactorAuthAge</code> 键。有关存储桶策略示例，请参阅 <a href="#">添加存储桶策略以请求 MFA (p. 308)</a> 。	2012 年 7 月 10 日
对象到期支持	借助“对象到期”功能，您可以设定在已配置时间段后自动删除数据。通过向存储桶添加生命周期配置，可以设置对象到期规则。	2011 年 12 月 27 日

更改	描述	日期
增加了新的支持区域	Amazon S3 现在支持 南美洲 ( 圣保罗 ) 区域。有关更多信息 , 请参阅 <a href="#">访问存储桶 (p. 51)</a> 。	2011 年 12 月 14 日
多对象删除	Amazon S3 现在支持多对象删除 API , 使您能够在单个请求中删除多个对象。借助此功能 , 您可以比使用多个单独的 DELETE 请求更快地从 Amazon S3 删除大量对象。有关更多信息 , 请参阅 <a href="#">删除对象 (p. 203)</a> 。	2011 年 12 月 7 日
增加了新的支持区域	Amazon S3 现在支持 美国西部 ( 俄勒冈 ) 区域。有关详细信息 , 请参阅 <a href="#">存储桶和区域 (p. 51)</a> 。	2011 年 11 月 8 日
文档更新	文档错误修复。	2011 年 11 月 8 日
文档更新	除了文档错误修复 , 本版本还包括以下增强功能 : <ul style="list-style-type: none"><li>• 使用 <a href="#">适用于 PHP 的 AWS 开发工具包</a> (请参阅 <a href="#">使用适用于 PHP 的 AWS 开发工具包指定服务器端加密 (p. 355)</a>) 和适用于 Ruby 的 AWS 开发工具包 (请参阅 <a href="#">使用适用于 Ruby 的 AWS 开发工具包指定服务器端加密 (p. 357)</a>) 的新服务器端加密部分。</li><li>• 有关创建和测试 Ruby 示例 (参阅 <a href="#">使用适用于 Ruby 的 AWS 开发工具包 - 版本 3 (p. 524)</a>) 的新部分。</li></ul>	2011 年 10 月 17 日
服务器端加密支持	Amazon S3 现在支持服务器端加密。它使您能够请求 Amazon S3 加密静态数据 , 即 , 当 Amazon S3 将数据写入其数据中心内的磁盘时 , 加密您的对象数据。除了 REST API 更新 , AWS SDK for Java 和 .NET 可提供必要的功能来请求服务器端加密。当使用 AWS 管理控制台上传对象时 , 还可以请求服务器端加密。要了解有关数据加密的更多信息 , 请转至 <a href="#">使用数据加密</a> 。	2011 年 10 月 4 日
文档更新	除了文档错误修复 , 本版本还包括以下增强功能 : <ul style="list-style-type: none"><li>• 在 <a href="#">创建请求 (p. 9)</a> 部分中添加了 Ruby 和 PHP 示例。</li><li>• 添加了描述如何生成和使用预签名 URL 的部分。有关更多信息 , 请参阅 <a href="#">与其他用户共享数据元 (p. 147)</a> 和 <a href="#">使用预签名 URL 上传对象 (p. 183)</a>。</li><li>• 更新了介绍 AWS Explorer for Eclipse 和 AWS Explorer for Visual Studio 的部分。有关更多信息 , 请参阅 <a href="#">使用 AWS 开发工具包、CLI 和 Explorer (p. 518)</a>。</li></ul>	2011 年 9 月 22 日
对使用临时安全凭证发送请求的支持	除了使用您的 AWS 账户和 IAM 用户安全凭证将经身份验证的请求发送到 Amazon S3 , 您现在还可以使用从 AWS Identity and Access Management (IAM) 获取的临时安全凭证发送请求。您可以使用 AWS Security Token Service API 或 AWS 开发工具包包装程序库从 IAM 请求这些临时凭证。您可以请求这些临时安全凭证以供自己使用 , 也可以提供给联合身份用户和应用程序使用。此功能使您能够管理 AWS 之外的用户 , 并为他们提供临时安全凭证来访问 AWS 资源。  有关更多信息 , 请参阅 <a href="#">创建请求 (p. 9)</a> 。  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 <a href="#">临时安全凭证</a> 。	2011 年 8 月 3 日

更改	描述	日期
扩展了分段上传 API 以支持复制多达 5 TB 的对象	<p>在此版本之前，Amazon S3 API 支持复制最大 5 GB 的对象。为了支持复制大于 5 GB 的对象，Amazon S3 现已通过新的操作 <code>Upload Part (Copy)</code> 扩展了分段上传 API。您可以使用该分段上传操作来复制多达 5 TB 的对象。有关更多信息，请参阅 <a href="#">复制对象 (p. 187)</a>。</p> <p>有关分段上传 API 的概念性信息，请参阅 <a href="#">使用分段上传 API 上传对象 (p. 155)</a>。</p>	2011 年 2 月 6 日
已禁用通过 HTTP 的 SOAP API 调用	为了提高安全性，已禁用通过 HTTP 的 SOAP API 调用。必须使用 SSL 将经身份验证的和匿名的 SOAP 请求发送到 Amazon S3。	2011 年 6 月 6 日
IAM 支持跨账户委托	<p>以前，要访问 Amazon S3 资源，IAM 用户需要从父 AWS 账户和 Amazon S3 资源拥有者获得权限。借助跨账户访问，IAM 用户现在仅需要从所有者账户获得权限。即，如果资源拥有者向 AWS 账户授予访问权限，则 AWS 账户现在可以授予其 IAM 用户对这些资源的访问权限。</p> <p>有关更多信息，请参阅 IAM 用户指南 中的 <a href="#">创建向 IAM 用户委派权限的角色</a>。</p> <p>有关在存储桶策略中指定委托人的更多信息，请参阅 <a href="#">在策略中指定委托人 (p. 281)</a>。</p>	2011 年 6 月 6 日
新链接	此服务的终端节点信息现在位于 AWS 一般参考中。有关更多信息，请参阅 <a href="#">AWS 一般参考</a> 中的“区域和终端节点”。	2011 年 3 月 1 日
支持在 Amazon S3 中托管静态网站	Amazon S3 增强了对托管静态网站的支持。其中包括支持索引文档和自定义错误文档。使用这些功能时，访问存储桶或子文件夹（如 <code>http://mywebsite.com/subfolder</code> ）根主题的请求会返回索引文档，而不是存储桶中对象的列表。遇到错误时，Amazon S3 会返回用户自定义的错误消息，而不是 Amazon S3 错误消息。有关更多信息，请参阅 <a href="#">在 Amazon S3 上托管静态网站 (p. 401)</a> 。	2011 年 2 月 17 日
支持响应标头 API	GET Object REST API 现在允许您为每个请求更改 REST GET Object 请求的响应标头。即，您可以改变响应中的对象元数据，无需改变对象本身。有关更多信息，请参阅 <a href="#">获取对象 (p. 142)</a> 。	2011 年 1 月 14 日
支持大型对象	Amazon S3 已将可以在 S3 存储桶中存储的对象的最大大小从 5 GB 增加到 5 TB。如果您使用 REST API，则可以在单个 PUT 操作中上传最多 5 GB 的对象。对于大型对象，必须使用分段上传 REST API 来分段上传对象。有关更多信息，请参阅 <a href="#">使用分段上传 API 上传对象 (p. 155)</a> 。	2010 年 12 月 9 日
分段上传	分段上传可以更快、更灵活地上传到 Amazon S3。它允许您上传单个对象作为一组分段。有关更多信息，请参阅 <a href="#">使用分段上传 API 上传对象 (p. 155)</a> 。	2010 年 11 月 10 日
支持存储桶策略中的规范 ID	现在您可以在存储桶策略中指定规范 ID。有关更多信息，请参阅 <a href="#">访问策略语言概述 (p. 279)</a> 。	2010 年 9 月 17 日
Amazon S3 与 IAM 结合使用	本服务现已与 AWS Identity and Access Management (IAM) 集成。有关更多信息，请转到 <a href="#">IAM 用户指南</a> 中的与 IAM 配合使用的 AWS 服务。	2010 年 9 月 2 日

更改	描述	日期
通知	Amazon S3 通知功能使您可以配置存储桶，以便当 Amazon S3 在存储桶上检测到关键事件时，Amazon S3 可向 Amazon Simple Notification Service (Amazon SNS) 主题发布消息。有关详细信息，请参阅 <a href="#">设置存储桶事件的通知 (p. 425)</a> 。	2010 年 7 月 14 日
存储桶策略	存储桶策略是用来跨存储桶、对象和对象集设置访问权限的访问管理系统。此功能可补充(在许多情况下，可替代)访问控制列表。有关更多信息，请参阅 <a href="#">使用存储桶策略和用户策略 (p. 279)</a> 。	2010 年 7 月 6 日
在所有区域中可用的路径类型语法	在美国传统区域，或者如果存储桶与请求的终端节点位于相同区域，Amazon S3 现在支持任何存储桶的路径类型语法。有关详细信息，请参阅 <a href="#">虚拟托管 (p. 42)</a> 。	2010 年 6 月 9 日
欧洲(爱尔兰)的新终端节点	Amazon S3 现在为欧洲(爱尔兰)提供终端节点： <a href="http://s3-eu-west-1.amazonaws.com">http://s3-eu-west-1.amazonaws.com</a> 。	2010 年 6 月 9 日
控制台	您现在可以通过 AWS 管理控制台使用 Amazon S3。您可以在 Amazon Simple Storage Service 控制台用户指南中了解控制台中的所有 Amazon S3 功能。	2010 年 6 月 9 日
去冗余	Amazon S3 现在使您能够通过在去冗余的 Amazon S3 中存储对象降低存储成本。有关详细信息，请参阅 <a href="#">低冗余存储 (p. 5)</a> 。	2010 年 5 月 12 日
增加了新的支持区域	Amazon S3 现在支持亚太区域(新加坡)区域。有关详细信息，请参阅 <a href="#">存储桶和区域 (p. 51)</a> 。	2010 年 4 月 28 日
对象版本控制	本版本引入了对象版本控制。所有对象现在都有键和版本。如果您启用对存储桶的版本控制，则 Amazon S3 将为所有添加到存储桶的对象提供唯一的版本 ID。此功能使您能够从意外覆盖和删除中恢复。有关详细信息，请参阅 <a href="#">版本控制 (p. 7)</a> 和 <a href="#">使用版本控制 (p. 380)</a> 。	2010 年 2 月 8 日
增加了新的支持区域	Amazon S3 现在支持美国西部(加利福尼亚北部)区域。针对此区域的请求的新终端节点为 <a href="http://s3-us-west-1.amazonaws.com">s3-us-west-1.amazonaws.com</a> 。有关详细信息，请参阅 <a href="#">存储桶和区域 (p. 51)</a> 。	2009 年 12 月 2 日
适用于 .NET 的 AWS 开发工具包	对于更喜欢使用 .NET 语言特定的 API 操作而不是使用 REST 或 SOAP 来构建应用程序的软件开发人员，AWS 现在为他们提供了库、示例代码、教程和其他资源。这些库提供了一些基本功能(未包括在 REST 或 SOAP API 中)，比如请求身份验证、请求重试和错误处置，以便您轻松地开始工作。有关特定语言的库和资源的更多信息，请参阅 <a href="#">使用 AWS 开发工具包、CLI 和 Explorer (p. 518)</a> 。	2009 年 11 月 11 日

# AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 [AWS 词汇表](#)。