ALIBABA CLOUD

Alibaba Cloud

对象存储 OSS
开发指南

文档版本：20220324

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.使用前须知 11
2.基本概念 12
3.访问域名（Endpoint） 14
3.1. 访问域名和数据中心 14
3.2. OSS访问域名使用规则 15
3.3. OSS内网域名与VIP网段对照表 17
3.4. ECS实例通过OSS内网地址访问OSS资源 18
3.5. 如何选择OSS地域 20
4.存储类型 21
4.1. 存储类型介绍 21
4.2. 存储类型转换 22
5.存储空间（Bucket） 25
5.1. 存储空间概述 25
5.2. 存储空间命名 26
5.3. 创建存储空间 26
5.4. 获取存储空间的地域信息 28
5.5. 列举存储空间 29
5.6. 分层命名空间 30
5.7. 生命周期 31
5.7.1. 生命周期规则介绍 31
5.7.2. 基于最后一次修改时间的生命周期规则介绍 32
5.7.3. 基于最后一次访问时间的生命周期规则介绍 36
5.7.4. 生命周期配置元素 38
5.7.5. 生命周期配置示例 40
5.8. 存储空间清单 47
5.9. 请求者付费模式 52

5.10. 绑定自定义域名 53
5.11. 传输加速 55
5.12. 设置跨域资源共享 60
5.13. 存储空间标签 63
5.14. 回源类型 64
5.14.1. 回源类型概述 64
5.14.2. 镜像回源 64
5.14.3. 重定向 67
5.15. 删除存储空间 68
5.16. 常见问题 69
5.16.1. CDN回源到OSS时，如何隐藏OSS返回的报错信息？ 69
5.16.2. 为什么CDN回源私有Bucket时，不支持访问Bucket的默认首页... 70
6.对象/文件（Object） 71
6.1. 对象概述 71
6.2. 对象命名 71
6.3. 上传文件 71
6.3.1. 简单上传 71
6.3.2. 分片上传 74
6.3.3. 断点续传上传 77
6.3.4. 上传回调 78
6.3.5. 授权给第三方上传 80
6.3.6. 表单上传 82
6.3.7. 追加上传 84
6.3.8. RTMP推流上传 86
6.4. 下载文件 88
6.4.1. 简单下载 88
6.4.2. 断点续传下载 89
6.4.3. 授权给第三方下载 90

6.5. 管理文件 92
6.5.1. 列举文件 92
6.5.2. 拷贝文件 93
6.5.3. 解冻文件 95
6.5.4. 删除文件 96
6.5.5. 对象标签 97
6.5.6. 管理文件元信息 99
6.5.7. 通过目录管理文件 101
6.5.8. 单链接限速 104
6.5.9. 使用SelectObject查询文件 106
6.6. 常见问题 110
6.6.1. OSS有带宽和QPS限制吗？ 110
6.6.2. 上传Object后如何获取访问URL？ 110
6.6.3. 如何设置Content-Type（MIME）？ 111
6.6.4. OSS有哪些批量操作？ 115
6.6.5. OSS怎样上传下载文件夹（目录）？ 117
6.6.6. OSS如何限制上传文件类型及大小？ 117
6.6.7. OSS MD5一致性校验说明 117
6.6.8. OSS的gzip压缩如何使用？ 118
6.6.9. 如何配置HTTPS请求和证书？ 118
6.6.10. OSS文件删除或覆盖后能不能恢复？ 119
6.6.11. 匿名用户无法访问公共读的Object 119
6.6.12. 为什么文件签名URL过期后仍可以访问？ 119
6.6.13. 哪些操作会影响OSS文件的LastModified属性？ 120
6.6.14. 签名URL可以修改过期时间吗？ 120
6.6.15. 如何修改、更新、编辑文件？ 120
6.6.16. OSS中可以重命名Bucket吗？是否支持Object迁移？ 120
6.6.17. 通过浏览器访问Bucket或Object时提示“TypeError: Failed t... 120

6.6.18. 访问CDN加速域名报错“You are forbidden to list bucket... 120
6.6.19. 为什么数据丢失了？ 121
7.数据安全 122
7.1. 访问控制 122
7.1.1. 访问控制概述 122
7.1.2. OSS鉴权详解 122
7.1.3. Bucket ACL 123
7.1.4. Object ACL 124
7.1.5. Bucket Policy 126
7.1.5.1. Bucket Policy概述 126
7.1.5.2. Bucket Policy常见示例 130
7.1.5.3. 教程示例：通过Bucket Policy限制公网访问OSS 135
7.1.5.4. 教程示例：基于Bucket Policy实现跨部门数据共享 136
7.1.5.5. 教程示例：基于Bucket Policy实现跨账号访问OSS 138
7.1.6. RAM Policy 139
7.1.6.1. RAM Policy概述 139
7.1.6.2. RAM Policy常见示例 142
7.1.6.3. 教程示例：使用RAM Policy控制OSS的访问权限 149
7.1.6.4. 教程示例：基于RAM角色实现跨账号访问OSS 154
7.2. 数据容灾 155
7.2.1. 同城冗余存储 155
7.2.2. 跨区域复制 157
7.2.3. 同区域复制 160
7.2.4. 特殊场景下的复制行为 162
7.2.5. 数据复制问题排查 163
7.3. 数据加密 164
7.3.1. 服务器端加密 164
7.3.2. 客户端加密 167

7.4. 版本控制 169
7.4.1. 版本控制介绍 169
7.4.2. 开启版本控制下Object的操作 172
7.4.3. 暂停版本控制下Object的操作 174
7.4.4. 删除标记 175
7.4.5. 常见问题 176
7.5. 签名 176
7.5.1. OSS请求流程 176
7.5.2. 在Header中包含签名 178
7.5.3. 在URL中包含签名 181
7.5.4. 签名常见问题 183
7.6. 使用STS临时访问凭证访问OSS 186
7.7. 防盗链 190
7.8. 合规保留策略 192
7.9. OSS沙箱 194
7.10. OSS高防 194
8.日志管理 196
8.1. 日志转存 196
8.2. 实时日志查询 198
9.静态网站托管 201
9.1. 静态网站托管介绍 201
9.2. 教程示例：使用自定义域名设置静态网站托管 205
9.3. 教程示例：通过静态网站托管部署单页应用 207
10.数据处理与索引 210
10.1. 数据处理与索引介绍 210
10.2. 新版图片处理指南 210
10.2.1. 简介 210
10.2.2. 图片处理操作方式 211

10.2.3. 图片处理参数 213
10.2.3.1. 图片缩放 213
10.2.3.2. 图片水印 216
10.2.3.3. 自定义裁剪 221
10.2.3.4. 质量变换 222
10.2.3.5. 格式转换 223
10.2.3.6. 获取信息 225
10.2.3.7. 自适应方向 225
10.2.3.8. 内切圆 226
10.2.3.9. 索引切割 227
10.2.3.10. 圆角矩形 228
10.2.3.11. 模糊效果 229
10.2.3.12. 旋转 229
10.2.3.13. 渐进显示 230
10.2.3.14. 获取图片主色调 231
10.2.3.15. 亮度 232
10.2.3.16. 锐化 232
10.2.3.17. 对比度 233
10.2.4. 图片样式 234
10.2.5. 图片处理持久化 235
10.2.6. 错误响应 237
10.2.7. 图片处理常见问题 237
10.2.8. 新旧版本图片处理服务及使用说明 241
10.3. 视频截帧 242
10.4. 数据索引（Data Indexing） 244
11.监控服务 246
11.1. 监控服务概览 246
11.2. 使用监控服务 246

11.3. 使用报警服务 253
11.4. 访问监控数据 255
11.5. 监控指标参考 258
11.6. 监控、诊断和故障排除 261
11.7. 常见问题 265
12.事件通知 268
12.1. 事件通知概述 268
12.2. 教程示例：结合消息服务实现OSS事件通知 271
13.数据湖接入 274
13.1. OSS-HDFS服务概述 274
13.2. OSS-HDFS服务快速入门 275
13.3. OSS-HDFS服务快照功能 279

1.使用前须知

本文列出您在使用阿里云对象存储OSS前需要了解的内容。

如果您初次使用阿里云OSS，请参见阿里云OSS快速入门系列文档，帮助您了解OSS并快速使用OSS。

如果您已经充分了解OSS，您也可以通过下列资源快速使用OSS的其他各项功能：

资源	描述
阿里云OSS基本概念	介绍阿里云OSS的核心概念。
阿里云OSS最佳实践	介绍阿里云OSS的各种使用场景与配置实践。
阿里云OSS SDK参考	介绍主流语言的SDK开发操作和参数。
阿里云OSS API参考	详细探讨了阿里云OSS支持的RESTful API操作和相关的示例。
阿里云OSS官方工具	介绍阿里云官方提供的各种便捷工具，帮助您更高效的管理OSS资源。
阿里云OSS控制台用户指南	阿里云OSS控制台可让您通过界面执行OSS的部分功能。本文档为您介绍基于阿里云OSS控制台的所有操作。
阿里云OSS图片处理指南	详细探讨了阿里云OSS提供的图片处理服务的详细内容与操作方式。

2.基本概念

本文将向您介绍对象存储OSS产品中涉及的几个基本概念，以便于您更好地理解OSS产品。

存储空间（Bucket）

存储空间是用户用于存储对象（Object）的容器，所有的对象都必须隶属于某个存储空间。存储空间具有各种配置属性，包括地域、访问权限、存储类型等。用户可以根据实际需求，创建不同类型的存储空间来存储不同的数据。

- 同一个存储空间的内部是扁平的，没有文件系统的目录等概念，所有的对象都直接隶属于其对应的存储空间。
- 每个用户可以拥有多个存储空间。
- 存储空间的名称在OSS范围内必须是全局唯一的，一旦创建之后无法修改名称。
- 存储空间内部的对象数目没有限制。

存储空间的命名规范如下：

- 只能包括小写字母、数字和短划线（-）。
- 必须以小写字母或者数字开头和结尾。
- 长度必须在3~63字符之间。

对象（Object）

对象是OSS存储数据的基本单元，也被称为OSS的文件。和传统的文件系统不同，对象没有文件目录层级结构的关系。对象由元信息（Object Meta），用户数据（Data）和文件名（Key）组成，并且由存储空间内部唯一的Key来标识。对象元信息是一组键值对，表示了对象的一些属性，比如最后修改时间、大小等信息，同时用户也可以在元信息中存储一些自定义的信息。

对象的生命周期是从上传成功到被删除为止。在整个生命周期内，除通过追加方式上传的Object可以通过继续追加上传写入数据外，其他方式上传的Object内容无法编辑，您可以通过重复上传同名的对象来覆盖之前的对象。

对象的命名规范如下：

- 使用UTF-8编码。
- 长度必须在1~1023字符之间。
- 不能以正斜线（/）或者反斜线（\）开头。

> **说明** 对象名称需要区分大小写。如无特殊说明，本文档中的对象、文件称谓等同于Object。

ObjectKey

在各语言SDK中，ObjectKey、Key以及ObjectName是同一概念，均表示对Object执行相关操作时需要填写的Object名称。例如向某一存储空间上传Object时，ObjectKey表示上传的Object所在存储空间的完整名称，即包含文件后缀在内的完整路径，如填写为abc/efg/123.jpg。

Region（地域）

Region表示OSS的数据中心所在物理位置。用户可以根据费用、请求来源等选择合适的地域创建Bucket。一般来说，距离用户更近的Region访问速度更快。详情请参见OSS已经开通的Region。

Region是在创建Bucket的时候指定的，一旦指定之后就不允许更改。该Bucket下所有的Object都存储在对应的数据中心，目前不支持Object级别的Region设置。

Endpoint（访问域名）

Endpoint表示OSS对外服务的访问域名。OSS以HTTP RESTful API的形式对外提供服务，当访问不同的Region的时候，需要不同的域名。通过内网和外网访问同一个Region所需要的Endpoint也是不同的。例如杭州Region的外网Endpoint是oss-cn-hangzhou.aliyuncs.com，内网Endpoint是oss-cn-hangzhou-internal.aliyuncs.com。具体的内容请参见各个Region对应的Endpoint。

AccessKey（访问密钥）

AccessKey简称AK，指的是访问身份验证中用到的AccessKeyId和AccessKeySecret。OSS通过使用AccessKeyId和AccessKeySecret对称加密的方法来验证某个请求的发送者身份。AccessKeyId用于标识用户；AccessKeySecret是用户用于加密签名字符串和OSS用来验证签名字符串的密钥，必须保密。对于OSS来说，AccessKey的来源有：

- Bucket的拥有者申请的AccessKey。
- 被Bucket的拥有者通过RAM授权给第三方请求者的AccessKey。
- 被Bucket的拥有者通过STS授权给第三方请求者的AccessKey。

更多AccessKey介绍请参见获取AccessKey。

强一致性

Object操作在OSS上具有原子性，操作要么成功要么失败，不会存在有中间状态的Object。OSS保证用户一旦上传完成之后读到的Object是完整的，OSS不会返回给用户一个部分上传成功的Object。

Object操作在OSS同样具有强一致性，用户一旦收到了一个上传（PUT）成功的响应，该上传的Object就已经立即可读，并且Object的冗余数据已经写成功。不存在一种上传的中间状态，即read-after-write却无法读取到数据。对于删除操作也是一样的，用户删除指定的Object成功之后，该Object立即变为不存在。

数据冗余机制

OSS使用基于纠删码、多副本的数据冗余存储机制，将每个对象的不同冗余存储在同一个区域内多个设施的多个设备上，确保硬件失效时的数据持久性和可用性。

- OSS Object操作具有强一致性，用户一旦收到了上传或复制成功的响应，则该上传的Object就已经立即可读，且数据已经冗余写入到多个设备中。
- OSS会通过计算网络流量包的校验和，验证数据包在客户端和服务端之间传输中是否出错，保证数据完整传输。
- OSS的冗余存储机制，可支持两个存储设施并发损坏时，仍维持数据不丢失。
  - 当数据存入OSS后，OSS会检测和修复丢失的冗余，确保数据持久性和可用性。
  - OSS会周期性地通过校验等方式验证数据的完整性，及时发现因硬件失效等原因造成的数据损坏。当检测到数据有部分损坏或丢失时，OSS会利用冗余的数据，进行重建并修复损坏数据。

OSS与文件系统的对比

对比项	OSS	文件系统
数据模型	OSS是一个分布式的对象存储服务，提供的是一个Key-Value对形式的对象存储服务。	文件系统是一种典型的树状索引结构。
数据获取	根据Object的名称（Key）唯一的获取该Object的内容。 虽然用户可以使用类似test1/test.jpg的名字，但是这并不表示用户的Object是保存在test1目录下面的。对于OSS来说，test1/test.jpg仅仅只是一个字符串，与example.jpg并没有本质的区别。因此不同名称的Object之间的访问消耗的资源是类似的。	一个名为test1/test.jpg的文件，访问过程需要先访问到test1这个目录，然后再在该目录下查找名为test.jpg的文件。
优势	支持海量的用户并发访问。	支持文件的修改，比如修改指定偏移位置的内容、截断文件尾部等。也支持文件夹的操作，比如重命名目录、删除目录、移动目录等非常容易。
劣势	OSS保存的Object不支持修改（追加写Object需要调用特定的接口，生成的Object也和正常上传的Object类型上有差别）。用户哪怕是仅仅需要修改一个字节也需要重新上传整个Object。 OSS可以通过一些操作来模拟类似文件夹的功能，但是代价非常昂贵。比如重命名目录，希望将test1目录重命名成test2，那么OSS的实际操作是将所有以test1/开头的Object都重新复制成以test2/开头的Object，这是一个非常消耗资源的操作。因此在使用OSS的时候要尽量避免类似的操作。	受限于单个设备的性能。访问越深的目录消耗的资源也越大，操作拥有很多文件的目录也会非常慢。

因此，将OSS映射为文件系统是非常低效的，也是不建议的做法。如果一定要挂载成文件系统的话，建议尽量只做写新文件、删除文件、读取文件这几种操作。使用OSS应该充分发挥其优点，即海量数据处理能力，优先用来存储海量的非结构化数据，比如图片、视频、文档等。

以下是OSS与文件系统的概念对比：

对象存储 OSS	文件系统
Object	文件
Bucket	主目录
Region	无
Endpoint	无
AccessKey	无
无	多级目录
GetService	获取主目录列表
GetBucket	获取文件列表
PutObject	写文件
AppendObject	追加写文件
GetObject	读文件
DeleteObject	删除文件
无	修改文件内容
CopyObject （目标文件和源文件相同）	修改文件属性
CopyObject（目标文件和源文件不同）	复制文件
无	重命名文件

3.访问域名（Endpoint）

3.1. 访问域名和数据中心

Region表示OSS的数据中心所在的地域，Endpoint表示OSS对外服务的访问域名。本文主要介绍Region与Endpoint的对应关系。

公共云下OSS Region和Endpoint对照表

公共云下OSS各地域Endpoint如下：

Region	Region ID	外网Endpoint	内网Endpoint①
华东1（杭州）	oss-cn-hangzhou	oss-cn-hangzhou.aliyuncs.com	oss-cn-hangzhou-internal.aliyuncs.com
华东2（上海）	oss-cn-shanghai	oss-cn-shanghai.aliyuncs.com	oss-cn-shanghai-internal.aliyuncs.com
华北1（青岛）	oss-cn-qingdao	oss-cn-qingdao.aliyuncs.com	oss-cn-qingdao-internal.aliyuncs.com
华北2（北京）	oss-cn-beijing	oss-cn-beijing.aliyuncs.com	oss-cn-beijing-internal.aliyuncs.com
华北 3（张家口）	oss-cn-zhangjiakou	oss-cn-zhangjiakou.aliyuncs.com	oss-cn-zhangjiakou-internal.aliyuncs.com
华北5（呼和浩特）	oss-cn-huhehaote	oss-cn-huhehaote.aliyuncs.com	oss-cn-huhehaote-internal.aliyuncs.com
华北6（乌兰察布）	oss-cn-wulanchabu	oss-cn-wulanchabu.aliyuncs.com	oss-cn-wulanchabu-internal.aliyuncs.com
华南1（深圳）	oss-cn-shenzhen	oss-cn-shenzhen.aliyuncs.com	oss-cn-shenzhen-internal.aliyuncs.com
华南2（河源）	oss-cn-heyuan	oss-cn-heyuan.aliyuncs.com	oss-cn-heyuan-internal.aliyuncs.com
华南3（广州）	oss-cn-guangzhou	oss-cn-guangzhou.aliyuncs.com	oss-cn-guangzhou-internal.aliyuncs.com
西南1（成都）	oss-cn-chengdu	oss-cn-chengdu.aliyuncs.com	oss-cn-chengdu-internal.aliyuncs.com
中国（香港）	oss-cn-hongkong	oss-cn-hongkong.aliyuncs.com	oss-cn-hongkong-internal.aliyuncs.com
美国（硅谷）*	oss-us-west-1	oss-us-west-1.aliyuncs.com	oss-us-west-1-internal.aliyuncs.com
美国（弗吉尼亚）*	oss-us-east-1	oss-us-east-1.aliyuncs.com	oss-us-east-1-internal.aliyuncs.com
日本（东京）*	oss-ap-northeast-1	oss-ap-northeast-1.aliyuncs.com	oss-ap-northeast-1-internal.aliyuncs.com
韩国（首尔）	oss-ap-northeast-2	oss-ap-northeast-2.aliyuncs.com	oss-ap-northeast-2-internal.aliyuncs.com
新加坡*	oss-ap-southeast-1	oss-ap-southeast-1.aliyuncs.com	oss-ap-southeast-1-internal.aliyuncs.com
澳大利亚（悉尼）*	oss-ap-southeast-2	oss-ap-southeast-2.aliyuncs.com	oss-ap-southeast-2-internal.aliyuncs.com
马来西亚（吉隆坡）*	oss-ap-southeast-3	oss-ap-southeast-3.aliyuncs.com	oss-ap-southeast-3-internal.aliyuncs.com
印度尼西亚（雅加达）*	oss-ap-southeast-5	oss-ap-southeast-5.aliyuncs.com	oss-ap-southeast-5-internal.aliyuncs.com
菲律宾（马尼拉）	oss-ap-southeast-6	oss-ap-southeast-6.aliyuncs.com	oss-ap-southeast-6-internal.aliyuncs.com
泰国（曼谷）	oss-ap-southeast-7	oss-ap-southeast-7.aliyuncs.com	oss-ap-southeast-7-internal.aliyuncs.com
印度（孟买）*	oss-ap-south-1	oss-ap-south-1.aliyuncs.com	oss-ap-south-1-internal.aliyuncs.com
德国（法兰克福）*	oss-eu-central-1	oss-eu-central-1.aliyuncs.com	oss-eu-central-1-internal.aliyuncs.com
英国（伦敦）	oss-eu-west-1	oss-eu-west-1.aliyuncs.com	oss-eu-west-1-internal.aliyuncs.com
阿联酋（迪拜）*	oss-me-east-1	oss-me-east-1.aliyuncs.com	oss-me-east-1-internal.aliyuncs.com

说明

- 关于OSS域名的构成规则及使用方式，请参见OSS访问域名使用规则。
- `oss.aliyuncs.com` 默认指向华东1（杭州）地域外网地址；`oss-internal.aliyuncs.com` 默认指向华东1（杭州）地域内网地址。
- ①与OSS同地域的阿里云产品可以通过内网Endpoint访问OSS。如果您是ECS用户，建议使用内网地址访问同地域的OSS。访问方式，请参见ECS实例通过OSS内网地址访问OSS资源。
- *标注的海外地域与OSS产品定价页或资源包购买页面中的部分海外地域在表述上略有差异，但代表的都是同一个地域。例如，美国（硅谷）地域也称为美西1或者美西。更多信息，请参见OSS产品定价或者购买资源包。
- IPv6、IPv4客户端均可以使用OSS提供的统一双栈域名访问OSS。更多信息，请参见通过IPv6地址访问OSS。

传输加速Endpoint

Bucket开启传输加速功能后，会增加如下传输加速Endpoint：

- 全球加速Endpoint：地址为*oss-accelerate.aliyuncs.com*。传输加速接入点分布在全球各地，全球各地的Bucket均可以使用该域名进行传输加速。
- 非中国内地加速Endpoint：地址为*oss-accelerate-overseas.aliyuncs.com*。传输加速接入点分布在除中国内地以外的各地域，仅在中国香港及海外各地域Bucket绑定未备案的域名做CNAME指向时使用。

更多信息，请参见传输加速。

3.2. OSS访问域名使用规则

OSS会为每一个存储空间（Bucket）分配默认的访问域名，本文介绍OSS访问域名的构成规则及使用方式。

OSS域名构成规则

针对OSS的网络请求，除了GetService (ListBuckets)以及DescribeRegions API以外，其他所有请求的域名都是由带有指定Bucket信息的三级域名组成的。

访问域名结构为 `BucketName.Endpoint` 。BucketName为您的存储空间名称，Endpoint为存储空间对应的地域域名。

Endpoint分为外网访问、内网访问以及传输加速Endpoint。传输加速Endpoint又分为全球加速Endpoint以及非中国内地加速Endpoint。例如华东1（杭州）地域的访问域名如下：

- 外网Endpoint：oss-cn-hangzhou.aliyuncs.com
- 内网Endpoint：oss-cn-hangzhou-internal.aliyuncs.com
- 传输加速全球加速Endpoint：oss-accelerate.aliyuncs.com
- 传输加速非中国内地加速Endpoint：oss-accelerate-overseas.aliyuncs.com

内网、外网访问Endpoint可直接使用，无需额外配置，而传输加速Endpoint使用前需先开启Bucket的传输加速功能。详情请参见开启传输加速。

> **说明**
> - OSS以HTTP RESTful API的形式对外提供服务，当访问不同的地域（Region）时，需要不同的访问域名。
> - Region和Endpoint对照表请参见访问域名和数据中心。
> - 您也可以通过绑定自定义域名或绑定传输加速域名，将OSS的外网访问域名替换为您的自有域名。

通过外网访问OSS

外网指的是互联网。通过外网访问产生的流入流量（写）是免费的，流出流量（读）是收费的。

> **说明** OSS费用详情请参见OSS产品定价和计量项和计费项。

外网访问OSS有如下两种方式：

- 方式一：以URL的形式访问OSS Object。

  以URL形式访问OSS Object时，与Object的读写权限ACL有关。

Object ACL	公共读或公共读写	私有
URL格式	`<Schema>://<Bucket>.<外网Endpoint>/<Object>`	`<Schema>://<Bucket>.<外网Endpoint>/<Object>?签名信息`
参数说明	○ Schema：HTTP或者为HTTPS。 ○ Bucket：OSS存储空间名称。 ○ 外网Endpoint：Bucket所在数据中心供外网访问的Endpoint，各地域Endpoint详情请参见访问域名和数据中心。 ○ Object：上传到OSS上的文件的访问路径。	除签名信息以外，其他参数的用法与公共读或公共读写Object相同。签名信息包含标识URL超时时间的Expires、密钥中的AccessKey ID以及Signature三种元素。 有关在文件URL中添加签名的具体步骤，请参见在URL中包含签名。
使用示例	您在华东1（杭州）地域创建了名为examplebucket的存储空间，存储空间下有名为example.txt的Object，该Object保存在exampledir目录下，且允许匿名访问。此时，文件URL为 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/exampledir/example.txt` 。	您在华东1（杭州）地域创建了名为examplebucket的存储空间，存储空间下有名为example.txt的私有Object，该Object保存在exampledir目录下。此时，文件URL为 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/exampledir/example.txt?OSSAccessKeyId=nz2pc56s936****&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv****` 。

> **注意** OSS访问域名需携带Object访问路径才可以被访问。如果仅使用访问域名，例如examplebucket.oss-cn-hangzhou.aliyuncs.com，会有报错提示。如果您希望通过OSS访问域名直接访问Object，可以通过配置静态网站托管来实现。详情请参见静态网站托管介绍。

- 方式二： 通过OSS SDK配置外网访问域名。

  OSS SDK会对您的每一个操作拼接访问域名。但您在对不同地域的Bucket进行操作的时候需要设置不同的Endpoint。

  以Java SDK为例，对华东1（杭州）的Bucket进行操作时，需要在对类实例化时设置Endpoint：

```
String accessKeyId = "<yourAccessKeyId>";
  String accessKeySecret = "<yourAccessKeySecret>";
  String endpoint = "oss-cn-hangzhou.aliyuncs.com";
  OSSClient client = new OSSClient(endpoint, accessKeyId, accessKeySecret);
```

通过内网访问OSS

内网指的是阿里云同地域产品之间的内部通信网络，例如您通过ECS云服务器访问同地域的OSS服务。内网产生的流入和流出流量均免费，但是请求次数仍会计费。

> **说明** OSS费用详情请参见OSS定价页和计费概述。

内网访问OSS有如下两种方式：

- 方式一：以URL的形式访问OSS Object。

  以URL形式访问OSS Object时，与Object的读写权限ACL有关。

Object ACL	公共读或公共读写	私有
URL格式	`<Schema>://<Bucket>.<内网Endpoint>/<Object>`	`<Schema>://<Bucket>.<内网Endpoint>/<Object>?签名信息`
参数说明	○ Schema：HTTP或者为HTTPS。 ○ Bucket：OSS存储空间名称。 ○ 内网Endpoint：Bucket所在数据中心供同地域ECS访问的内网Endpoint。有关各地域的Endpoint详情，请参见访问域名和数据中心。 ○ Object：上传到OSS上的文件的访问路径。	除签名信息以外，其他参数的用法与公共读或公共读写Object相同。签名信息包含标识URL超时时间的Expires、密钥中的AccessKey ID以及Signature三种元素。 有关在文件URL中添加签名的具体步骤，请参见在URL中包含签名。
使用示例	您在华东1（杭州）地域创建了名为examplebucket的存储空间，存储空间下有名为example.txt的Object，该Object保存在exampledir目录下，且允许匿名访问。此时，文件夹URL为 `https://examplebucket.oss-cn-hangzhou-internal.aliyuncs.com/exampledir/example.txt` 。	您在华东1（杭州）地域创建了名为examplebucket的存储空间，存储空间下有名为example.txt的私有Object，该Object保存在exampledir目录下。此时，文件URL为 `https://examplebucket.oss-cn-hangzhou-internal.aliyuncs.com/exampledir/example.txt?OSSAccessKeyId=nz2pc56s936****&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv****` 。

- 方式二：通过ECS使用OSS SDK配置内网Endpoint。

  以Java SDK为例，对华东1（杭州）地域的Bucket进行操作时，需要将Endpoint设置为华东1（杭州）地域的内网Endpoint。

```
String accessKeyId = "<yourAccessKeyId>";
  String accessKeySecret = "<yourAccessKeySecret>";
  String endpoint = "oss-cn-hangzhou-internal.aliyuncs.com";
  OSSClient client = new OSSClient(endpoint, accessKeyId, accessKeySecret);
```

  同一个Region的ECS和OSS之间内网互通，不同Region的ECS和OSS之间内网不互通。例如您的OSS有两个Bucket，并且购买了华北2（北京）的ECS：

  - 其中一个Bucket名称为srcbucket，所在Region为华北2（北京），您可以在华北2（北京）的ECS中使用 `https://srcbucket.oss-cn-beijing-internal.aliyuncs.com` 来访问srcbucket的资源。
  - 另外一个Bucket名称为destbucket，所在Region为华北1（青岛），您无法在华北2（北京）的ECS中使用 `https://destbucket.oss-cn-qingdao-internal.aliyuncs.com` 来访问destbucket的资源，必须使用外网地址 `https://destbucket.oss-cn-qingdao.aliyuncs.com` 。

通过传输加速域名访问OSS

OSS传输加速支持数据上传、下载加速，可优化跨国、跨洋数据上传、下载体验。使用传输加速域名前，需先开启传输加速功能。开启后，您只需将外网Endpoint替换为传输加速Endpoint，即可实现数据传输加速。

以全球加速Endpoint为例，通过浏览器访问位于华东1（杭州）的存储空间examplebucket根目录下的文件*myphoto.jpg*，且文件读写权限ACL为公共读或者公共读写，此时文件URL为 `https://examplebucket.oss-accelerate.aliyuncs.com/myphoto.jpg` 。

如果*myphoto.jpg*的读写权限ACL为私有，则还要在文件URL中添加签名信息。例如 `https://examplebucket.oss-accelerate.aliyuncs.com/myphoto.jpg?OSSAccessKeyId=nz2pc56s936****&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv****` 。有关在文件URL中添加签名的具体步骤，请参见在URL中包含签名。

更多关于传输加速功能的介绍，请参见传输加速。

通过IPv6地址访问OSS

IPv6是互联网工程任务组IETF（Internet Engineering Task Force）设计用于替代现行版本IP协议（IPv4）的下一代IP协议，它可以让地球上的每一粒沙子都拥有地址。目前OSS已支持通过IPv6、IPv4双栈域名访问。

您的IPv6、IPv4客户端均可以使用OSS提供的统一双栈域名访问您的Bucket，DNS服务器将按照您使用的协议版本解析对应的OSS服务器地址。例如，杭州地域的Endpoint为 `cn-hangzhou.oss.aliyuncs.com` ，Bucket名称为examplebucket，则IPv6、IPv4客户端都可以通过 `https://examplebucket.cn-hangzhou.oss.aliyuncs.com` 访问该Bucket。

> **注意** 当前不支持通过专有网络VPC或经典网络ECS解析IPv6或IPv4地址。

目前可以通过IPv6协议访问的Endpoint如下：

Region	Endpoint
华东1（杭州）	cn-hangzhou.oss.aliyuncs.com
华东2（上海）	cn-shanghai.oss.aliyuncs.com
华北1（青岛）	cn-qingdao.oss.aliyuncs.com
华北2（北京）	cn-beijing.oss.aliyuncs.com
华北 3（张家口）	cn-zhangjiakou.oss.aliyuncs.com
华北5（呼和浩特）	cn-huhehaote.oss.aliyuncs.com
华北6（乌兰察布）	cn-wulanchabu.oss.aliyuncs.com
华南1（深圳）	cn-shenzhen.oss.aliyuncs.com
华南2（河源）	cn-heyuan.oss.aliyuncs.com
华南3（广州）	cn-guangzhou.oss.aliyuncs.com
西南1（成都）	cn-chengdu.oss.aliyuncs.com
中国（香港）	cn-hongkong.oss.aliyuncs.com
杭州金融云	cn-hangzhou-finance.oss.aliyuncs.com

Region	Endpoint
上海金融云	cn-shanghai-finance.oss.aliyuncs.com
深圳金融云	cn-shenzhen-finance.oss.aliyuncs.com

3.3. OSS内网域名与VIP网段对照表

云下机房设备或异地ECS实例希望通过内网访问OSS时，可通过CEN、高速通道、专线、VPN等连接OSS所在地域内网网络，并配置指向OSS内网网段的路由。本文列出OSS各地域的内网网段。

警告

- OSS为每个Region内网VIP网段划分了固定地址段，您按Region配置路由时必须按照下表地址配置完整的路由，否则可能会造成网络不通。
- 使用ECS实例通过内网访问OSS时，安全组中不能禁止以下任一VIP网段。

Region	Region ID	VPC网络Endpoint	VIP网段
华东1（杭州）	oss-cn-hangzhou	oss-cn-hangzhou-internal.aliyuncs.com	• 100.118.28.0/24 • 100.114.102.0/24 • 100.98.170.0/24 • 100.118.31.0/24
华东2（上海）	oss-cn-shanghai	oss-cn-shanghai-internal.aliyuncs.com	• 100.98.35.0/24 • 100.98.110.0/24 • 100.98.169.0/24 • 100.118.102.0/24
华北1（青岛）	oss-cn-qingdao	oss-cn-qingdao-internal.aliyuncs.com	• 100.115.173.0/24 • 100.99.113.0/24 • 100.99.114.0/24 • 100.99.115.0/24
华北2（北京）	oss-cn-beijing	oss-cn-beijing-internal.aliyuncs.com	• 100.118.58.0/24 • 100.118.167.0/24 • 100.118.170.0/24 • 100.118.171.0/24 • 100.118.172.0/24 • 100.118.173.0/24
华北 3（张家口）	oss-cn-zhangjiakou	oss-cn-zhangjiakou-internal.aliyuncs.com	• 100.118.90.0/24 • 100.98.159.0/24 • 100.114.0.0/24 • 100.114.1.0/24
华北5（呼和浩特）	oss-cn-huhehaote	oss-cn-huhehaote-internal.aliyuncs.com	• 100.118.195.0/24 • 100.99.110.0/24 • 100.99.111.0/24 • 100.99.112.0/24
华北6（乌兰察布）	oss-cn-wulanchabu	oss-cn-wulanchabu-internal.aliyuncs.com	• 100.114.11.0/24 • 100.114.12.0/24 • 100.114.100.0/24 • 100.118.214.0/24
华南1（深圳）	oss-cn-shenzhen	oss-cn-shenzhen-internal.aliyuncs.com	• 100.118.78.0/24 • 100.118.203.0/24 • 100.118.204.0/24 • 100.118.217.0/24
华南2（河源）	oss-cn-heyuan	oss-cn-heyuan-internal.aliyuncs.com	• 100.98.83.0/24 • 100.118.174.0/24
华南3（广州）	oss-cn-guangzhou	oss-cn-guangzhou-internal.aliyuncs.com	• 100.115.33.0/24 • 100.114.101.0/24
西南1（成都）	oss-cn-chengdu	oss-cn-chengdu-internal.aliyuncs.com	• 100.115.155.0/24 • 100.99.107.0/24 • 100.99.108.0/24 • 100.99.109.0/24

Region	Region ID	VPC网络Endpoint	VIP网段
中国（香港）	oss-cn-hongkong	oss-cn-hongkong-internal.aliyuncs.com	• 100.115.61.0/24 • 100.99.103.0/24 • 100.99.104.0/24 • 100.99.106.0/24
美国（硅谷）*	oss-us-west-1	oss-us-west-1-internal.aliyuncs.com	提交工单咨询
美国（弗吉尼亚）*	oss-us-east-1	oss-us-east-1-internal.aliyuncs.com	• 100.115.60.0/24 • 100.99.100.0/24 • 100.99.101.0/24 • 100.99.102.0/24
日本（东京）*	oss-ap-northeast-1	oss-ap-northeast-1-internal.aliyuncs.com	提交工单咨询
韩国（首尔）	oss-ap-northeast-2	oss-ap-northeast-2-internal.aliyuncs.com	提交工单咨询
新加坡*	oss-ap-southeast-1	oss-ap-southeast-1-internal.aliyuncs.com	• 100.118.219.0/24 • 100.99.213.0/24 • 100.99.116.0/24 • 100.99.117.0/24
澳大利亚（悉尼）*	oss-ap-southeast-2	oss-ap-southeast-2-internal.aliyuncs.com	提交工单咨询
马来西亚（吉隆坡）*	oss-ap-southeast-3	oss-ap-southeast-3-internal.aliyuncs.com	• 100.118.165.0/24 • 100.99.125.0/24 • 100.99.130.0/24 • 100.99.131.0/24
印度尼西亚（雅加达）*	oss-ap-southeast-5	oss-ap-southeast-5-internal.aliyuncs.com	提交工单咨询
菲律宾（马尼拉）	oss-ap-southeast-6	oss-ap-southeast-6-internal.aliyuncs.com	100.115.16.0/24
泰国（曼谷）	oss-ap-southeast-7	oss-ap-southeast-7-internal.aliyuncs.com	提交工单咨询
印度（孟买）*	oss-ap-south-1	oss-ap-south-1-internal.aliyuncs.com	• 100.118.211.0/24 • 100.99.122.0/24 • 100.99.123.0/24 • 100.99.124.0/24
德国（法兰克福）*	oss-eu-central-1	oss-eu-central-1-internal.aliyuncs.com	100.115.154.0/24
英国（伦敦）	oss-eu-west-1	oss-eu-west-1-internal.aliyuncs.com	提交工单咨询
阿联酋（迪拜）*	oss-me-east-1	oss-me-east-1-internal.aliyuncs.com	提交工单咨询

说明 *标注的海外地域与OSS产品定价页或资源包购买页面中的部分海外地域在表述上略有差异，但代表的都是同一个地域。例如，美国（硅谷）地域也称为美西1或者美西。更多信息请参见OSS产品定价或者购买资源包。

3.4. ECS实例通过OSS内网地址访问OSS资源

当您通过OSS内网地址访问OSS资源时，不收取流量费用。本文介绍ECS实例如何通过OSS内网地址访问OSS资源。

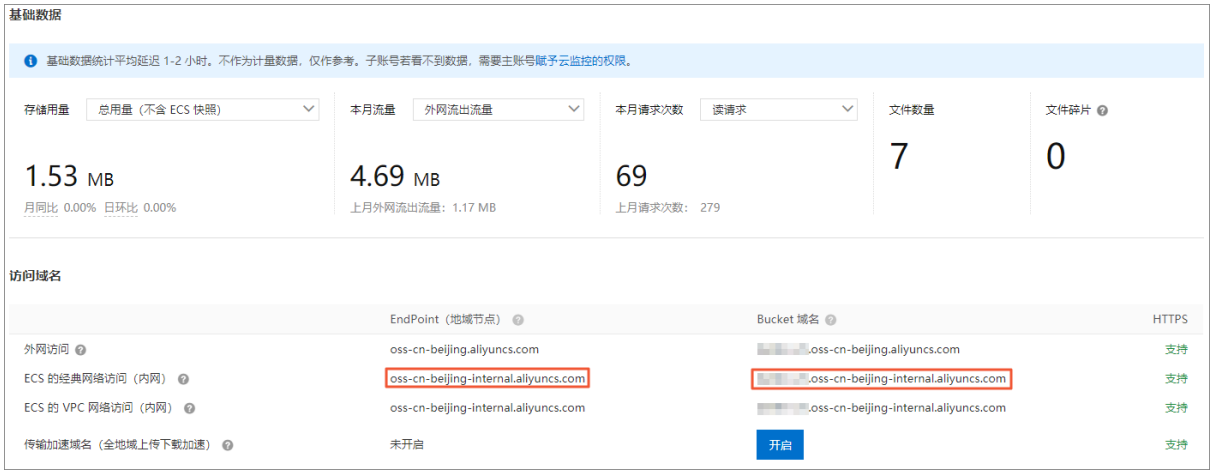
通过OSS内网地址访问OSS资源有以下两种方式：

- 与OSS同地域ECS实例可以直接通过内网访问有权限的OSS资源。
- 与OSS不同地域的ECS实例或公网用户可通过配置ECS反向代理，间接实现通过OSS内网地址访问OSS资源。

获取OSS内网地址

- 通过OSS控制台获取

登录OSS管理控制台，打开指定Bucket的概览页面，在**访问域名**区域查看Bucket的Endpoint和Bucket域名，如下图所示。



- 通过固定格式获取

  OSS的访问地址为固定格式：`BucketName.Endpoint`。其中，`BucketName`为您的存储空间名称，`Endpoint`为存储空间所在的地域对应的访问域名。详情请参见OSS访问域名使用规则。

同地域ECS实例访问OSS资源

与OSS同地域的ECS实例可以通过以下方式使用内网访问OSS资源：

- 通过URL直接访问OSS资源

  您可以直接使用OSS资源的内网地址访问有权限的OSS资源。例如，杭州地域某Bucket名为test，根目录下有个Object名为1.jpg，处于公共读状态。此时，杭州地域的ECS实例均可以使用`http://test.oss-cn-hangzhou-internal.aliyuncs.com/1.jpg`访问此Object。因此，您可以将OSS资源的访问URL嵌入到您的网站中，提供给同地域的ECS用户或已通过专线接入到与OSS同地域内网的用户访问。

  > **警告** 为了您的数据安全，不建议您将OSS资源设置为公共读或公共读写，您可以通过Bucket Policy授权给指定用户访问您的资源。

- 通过ossbrowser访问OSS资源

  您可以在配置ossbrowser访问参数的时候，将Endpoint设置为自定义，并填写OSS的内网Endpoint地址。详情请参见ossbrowser。

- 通过ossutil访问OSS资源

  您可以在配置ossutil访问参数的时候，将Endpoint设置为OSS的内网Endpoint地址。详情请参见ossutil。

- 通过SDK访问OSS资源

  SDK初始化client的时候，Endpoint配置OSS内网对应的Endpoint即可。

  - Java SDK

    ```
    String endpoint = "http://oss-cn-hangzhou-internal.aliyuncs.com";//以华东 1为例
    String accessKeyId = "<yourAccessKeyId>";
    String accessKeySecret = "<yourAccessKeySecret>";
    OSSClient client = new OSSClient(endpoint, accessKeyId, accessKeySecret);
    ```

    更多详情请参见Java SDK初始化。

  - PHP SDK

    ```
    $accessKeyId = "<yourAccessKeyId>";
    $accessKeySecret = "<yourAccessKeySecret>";
    $endpoint = "<您选定的OSS数据中心访问域名，例如http://oss-cn-hangzhou-internal.aliyuncs.com>";
    ```

    更多详情请参见PHP SDK初始化。

  - Python SDK

    ```
    auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
    endpoint = 'http://oss-cn-hangzhou-internal.aliyuncs.com' # 您选定的OSS数据中心访问域名，假设Bucket处于杭州地域
    bucket = oss2.Bucket(auth, endpoint, 'BucketName')
    ```

    更多详情请参见Python SDK初始化。

  - .NET SDK

    ```
    const string accessKeyId = "<yourAccessKeyId>";
    const string accessKeySecret = "<yourAccessKeySecret>";
    const string endpoint = "http://oss-cn-hangzhou-internal.aliyuncs.com";
    var ossClient = new OssClient(endpoint, accessKeyId, accessKeySecret);
    ```

    更多详情请参见.NET SDK初始化。

- C SDK

```
ptions->config = oss_config_create(options->pool);
aos_str_set(&options->config->endpoint, "http://oss-cn-hangzhou-internal.aliyuncs.com");
aos_str_set(&options->config->access_key_id, "<yourAccessKeyId>");
aos_str_set(&options->config->access_key_secret, "<yourAccessKeySecret>");
options->config->is_cname = 0;
options->ctl = aos_http_controller_create(options->pool, 0);
```

更多详情请参见C SDK初始化。

通过ECS反向代理访问OSS资源

不同地域的ECS实例或外网用户是无法直接通过OSS内网地址访问OSS资源的，但是您可以通过配置ECS反向代理来间接实现：

1. 在OSS同地域创建一个有公网地址的ECS实例。详情请参见创建ECS实例。
2. 在ECS实例上配置反向代理。详情请参见如何在ECS实例中反向代理OSS域名并实现HTTPS访问。
3. OSS配置Bucket Policy，允许该ECS实例的内网地址访问OSS资源。详情请参见通过Bucket Policy授权用户访问指定资源。

以上步骤配置完成后，您的用户将通过您的ECS公网地址访问您的OSS资源。当用户访问时，ECS实例通过内网向OSS请求资源，之后再返回给用户。

3.5. 如何选择OSS地域

本文介绍在创建OSS的存储空间（Bucket）时如何选择合适的地域。

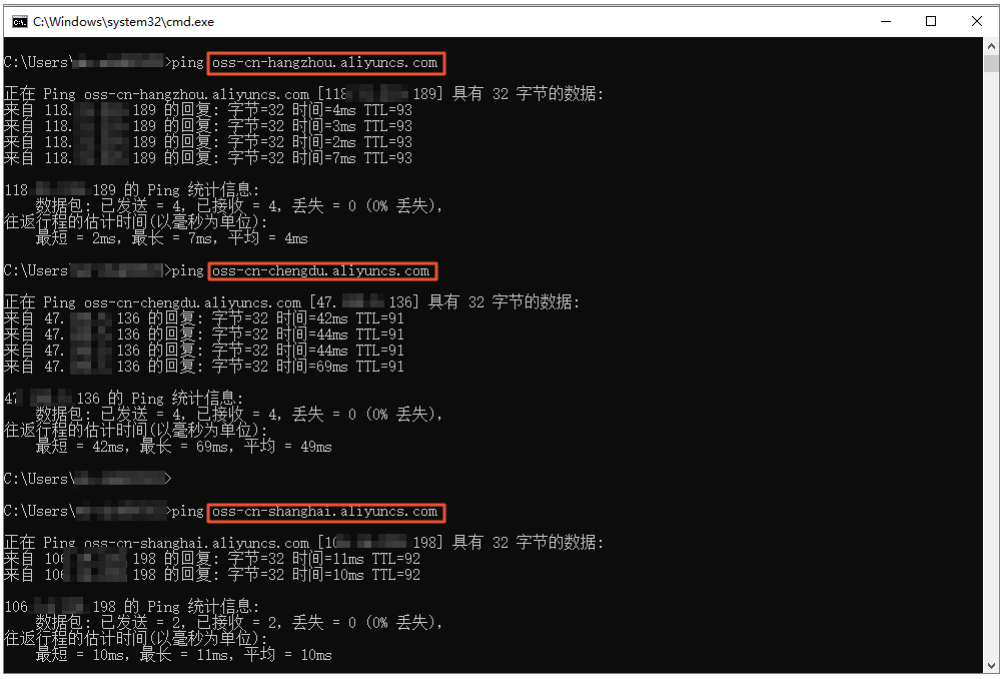
选择OSS地域时，通常需要考虑以下几个方面：

- 用户所在地
- 云产品之间的关系
- 资源价格
- 产品功能

用户所在地

如果您的OSS资源需要开放给其他用户访问，且希望用户有良好的访问体验，就必须考虑用户访问时的网络延迟。除了通信线路的质量外，距离是影响网络延迟的一个重要因素。

例如以杭州的用户为例，使用**ping**命令可以测试出其访问不同地域OSS数据中心的延迟情况。



可以看出，距离访问的数据中心越远，数据返回所需时间越长。所以，在选择地域的时候，尽量考虑离用户更近的地域。

云产品之间的关系

如果您需要将OSS作为其他阿里云产品的数据源，则需要根据其他云产品的地域去选择OSS的地域。当其他云产品和OSS在同一地域时，可以通过VPC地址访问OSS。访问时不会产生流量费用，且访问速度较外网会更快。

资源价格

因各地域的优惠政策不同，某些地域的资源包价格会更优惠。选择OSS地域的时候可优先考虑资源包价格更优惠的地域。

产品功能

OSS的新功能在发布初期会选择部分地域进行公测，若您希望试用这些新功能，需在指定地域创建存储空间。OSS产品功能发布记录请参见新功能发布记录。

4.存储类型

4.1. 存储类型介绍

对象存储OSS提供标准、低频访问、归档、冷归档四种存储类型，全面覆盖从热到冷的各种数据存储场景。

> **说明** 各存储类型的定价，请参见OSS产品定价。各存储类型的计费方式，请参见存储费用。

标准存储（Standard）

提供高可靠、高可用、高性能的对象存储服务，能够支持频繁的数据访问。适用于各种社交、分享类的图片、音视频应用、大型网站、大数据分析等业务场景。提供标准存储-本地冗余（LRS）和标准存储-同城冗余（ZRS）两种数据冗余存储方式。

- 标准存储-本地冗余（LRS）

  采用数据冗余存储机制，将每个对象的不同冗余存储在同一个可用区内多个设施的多个设备上，确保硬件失效时的数据持久性和可用性。

- 标准存储-同城冗余（ZRS）

  采用多可用区（AZ）机制，将用户的数据分散存放在同一地域（Region）的3个可用区。当某个可用区不可用时，仍然能够保障数据的正常访问。

低频访问（Infrequent Access）

提供高持久性、较低存储成本的对象存储服务。有最低存储时间（30天）和最小计量单位（64 KB）要求。支持数据实时访问，访问数据时会产生数据取回费用，适用于较低访问频率（平均每月访问频率1到2次）的业务场景。提供低频访问-本地冗余（LRS）和低频访问-同城冗余（ZRS）两种数据冗余存储方式。

- 低频访问-本地冗余（LRS）

  采用数据冗余存储机制，将每个对象的不同冗余存储在同一个可用区内多个设施的多个设备上，确保硬件失效时的数据持久性和可用性。

- 低频访问-同城冗余（ZRS）

  采用多可用区（AZ）机制，将用户的数据分散存放在同一地域（Region）的3个可用区。当某个可用区不可用时，仍然能够保障数据的正常访问。

归档存储（Archive）

提供了高持久性、极低存储成本的对象存储服务。有最低存储时间（60天）和最小计量单位（64 KB）要求。数据需解冻（约1分钟）后访问，解冻会产生数据取回费用。适用于数据长期保存的业务场景，例如档案数据、医疗影像、科学资料、影视素材等。

冷归档存储（Cold Archive）

提供了高持久性的对象存储服务，存储费用在四种存储类型中最低。有最低存储时间（180天）和最小计量单位（64 KB）要求。数据需解冻后访问，解冻时间根据数据大小和选择的解冻模式决定，解冻会产生数据取回费用。适用于需要超长时间存放的极冷数据，例如因合规要求需要长期留存的数据、大数据及人工智能领域长期积累的原始数据、影视行业长期留存的媒体资源、在线教育行业的归档视频等业务场景。

> **说明** 冷归档存储类型支持以下地域：华北1（青岛）、华北2（北京）、华北3（张家口）、华东1（杭州）、华东2（上海）、华南1（深圳）、西南1（成都）、华北6（乌兰察布）、中国香港、澳大利亚（悉尼）、新加坡、美国（硅谷）、德国（法兰克福）、马来西亚（吉隆坡）、印度尼西亚（雅加达）、印度（孟买）、阿联酋（迪拜）。请联系技术支持申请使用。

存储类型对比

对比指标	标准存储-本地冗余（LRS）	标准存储-同城冗余（ZRS）	低频访问-本地冗余（LRS）	低频访问-同城冗余（ZRS）	归档存储类型	冷归档存储类型
数据设计持久性	99.999999999%（11个9）	99.9999999999%（12个9）	99.999999999%（11个9）	99.9999999999%（12个9）	99.999999999%（11个9）	99.999999999%（11个9）
服务可用性	99.99%	99.995%	99.00%	99.50%	99.00%（数据解冻之后）	99.00%（数据解冻之后）
最小计量单位	无	无	64 KB	64 KB	64 KB	64 KB
最低存储时间	无	无	30天	30天	60天	180天
数据取回费用	无	无	按实际获取的数据量收取，单位GB。	按实际获取的数据量收取，单位GB。	按实际解冻的数据量收取，单位GB。	按实际解冻时选择的数据取回能力及数据大小收取，单位GB。
数据访问特点	实时访问，毫秒延迟。	实时访问，毫秒延迟。	实时访问，毫秒延迟。	实时访问，毫秒延迟。	数据需要先解冻，解冻完成后才能读取。解冻时间需要1分钟。	数据需要先解冻，解冻完成后才能读取。不同优先级的首字节取回能力如下： • 高优先级：1小时以内 • 标准：2~5小时 • 批量：5~12小时
图片处理	支持	支持	支持	支持	支持，但需要先解冻。	支持，但需要先解冻。
适用场景	各种社交、分享类的图片、音视频应用、大型网站、大数据分析等业务场景。例如程序下载、移动应用等。	各种社交、分享类的图片、音视频应用、大型网站、大数据分析等，且对持久性和可用性有更高要求的业务场景。例如企业重要文件、敏感信息等。	较低访问频率（平均每月访问频率1到2次）的业务场景。例如热备数据、监控视频数据等。	较低访问频率（平均每月访问频率1到2次），且对持久性和可用性有更高要求的业务场景。例如企业业务数据、近期的医疗档案等。	数据长期保存的业务场景。例如档案数据、医疗影像、科学资料、影视素材等	需要超长时间存放的极冷数据。例如因合规要求需要长期留存的数据、大数据及人工智能领域长期积累的原始数据、影视行业长期留存的媒体资源、在线教育行业的归档视频等。

说明 数据取回费用中的数据是从底层分布式存储系统读取的数据量，在公网传输的数据量会计入到流出流量的计费项中。

4.2. 存储类型转换

OSS支持标准存储、低频访问、归档存储、冷归档存储四种存储类型，您可以通过生命周期规则或者CopyObject的方式随时转换文件（Object）的存储类型。

说明 有关OSS支持的四种存储类型的更多信息，请参见存储类型介绍。

通过生命周期规则自动转换Object的存储类型

OSS生命周期管理（Lifecycle）提供Object Transition机制，支持自动转换文件存储类型。

- 基于最后一次修改时间的存储类型转换规则
  - 本地冗余（LRS）

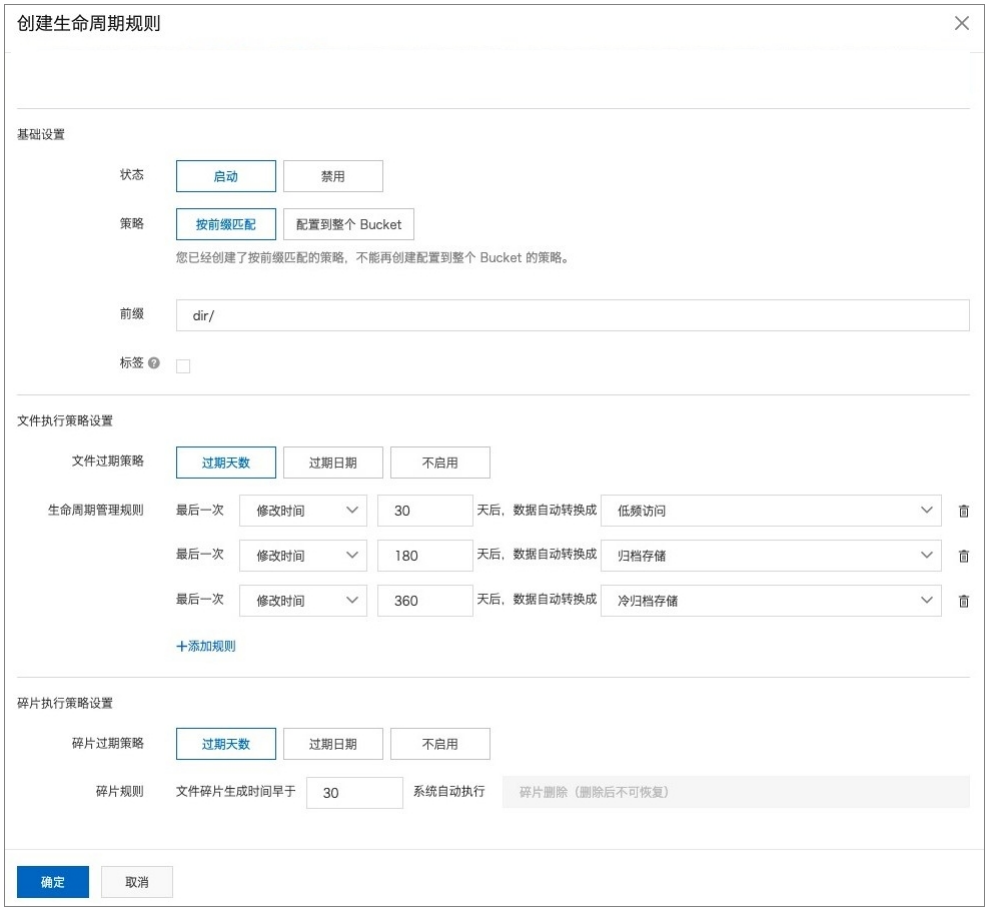
    本地冗余类型文件转换规则如下：
    - 标准存储（LRS）类型可转换为低频访问（LRS）、归档存储（LRS）和冷归档存储（LRS）类型。
    - 低频访问（LRS）类型可转换为归档存储（LRS）和冷归档存储（LRS）类型。
    - 归档存储（LRS）类型可转换为冷归档存储（LRS）类型。
  - 同城冗余（ZRS）

    同城冗余类型文件转换规则：仅支持标准存储（ZRS）类型转换为低频访问（ZRS）类型。

  更多信息，请参见基于最后一次修改时间的生命周期规则介绍。
- 基于最后一次访问时间的存储类型转换规则

  您可以通过生命周期规则指定距离标准存储类型Object的最后一次访问时间达到指定天数后，数据自动转换成低频存储类型，且在数据被访问后，依旧保留为低频访问类型或者转为标准存储类型。更多信息，请参见基于最后一次访问时间的生命周期规则介绍。
- 存储类型转换示例

  例如Bucket的冗余类型为本地冗余，您希望结合基于最后一次修改时间的生命周期规则实现Bucket内指定前缀的Object在达到指定天数后转换为目标存储类型，策略说明如下：
  - Object存储30天后，自动转换为低频访问类型。
  - Object存储180天后，自动转换为归档存储类型。
  - Object存储360天后，自动转换为冷归档存储类型。



当Bucket同时配置了Object保留指定周期后转换为低频访问、Object保留指定周期后转换为归档存储、Object保留指定周期后转换为冷归档存储的策略，其转换周期必须满足以下条件：

转换为低频访问的周期<转换为归档的周期<转换为冷归档的周期

- 生命周期规则配置方式

操作方式	说明
控制台	Web应用程序，直观易用。
命令行工具	命令行工具，性能好。
Java SDK	丰富、完整的各类语言SDK demo。
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	
Node.js SDK	
Ruby SDK	

通过CopyObject接口手动转换Object的存储类型

您可以通过CopyObject接口，将Object覆写为指定的存储类型。如果转换的Object是低频访问、归档存储或冷归档存储类型，且存储未满指定周期的，会产生存储不足规定时长容量费用。更多信息，请参见存储费用。

- 基于CopyObject的存储类型转换规则
  - 本地冗余（LRS）

    各存储类型之间可任意转换。

    > **说明** 归档或冷归档类型的文件需要解冻（Restore）之后才能修改存储类型。关于如何解冻文件，请参见解冻文件。

  - 同城冗余（ZRS）

    仅支持标准存储（ZRS）和低频访问（ZRS）之间互相转换。

- 基于CopyObject的存储类型转换操作方式

操作方式	说明
控制台	Web应用程序，直观易用。
命令行工具ossutil	命令行工具，性能好。
Java SDK	丰富、完整的各类语言SDK demo。
Python SDK	
Go SDK	
C++ SDK	

注意事项

当Object被转换为低频访问、归档存储和冷归档存储类型后，需注意如下事项：

- 最小计量空间

  对于小于64 KB的Object，会按照64 KB计算空间大小。

- 最短存储周期

  低频访问类型的Object需要至少保存30天；归档存储类型的Object需至少保存60天；冷归档存储类型的Object需至少保存180天。如果存储未满指定周期，会收取存储不足规定时长容量费用。

  - 通过生命周期自动转换Object存储类型

    通过生命周期转换Object存储类型时，不会重新计算Object的存储时间。例如*a.txt*作为标准存储类型已经在OSS中存储了10天，通过生命周期转换为低频访问类型，继续存储20天即满足最少存储30天的要求。详情请参见常见问题。

  - 手动转换Object存储类型

    手动转换Object存储类型时，会重新计算Object的存储时间。例如*a.txt*作为标准存储类型已经在OSS中存储了10天，手动将Object转换为低频访问类型，则需继续存储30天才满足最少存储30天的要求。

- 解冻时间

  归档存储和冷归档存储类型Object恢复为可读取状态需要一定的解冻时间，如果业务场景需要实时读取Object时，不建议将Object转换成归档存储或冷归档存储类型。

- 数据取回费用

  访问低频访问类型的Object时，会根据实际访问量额外收取数据取回费用；解冻归档存储和冷归档存储类型的Object会额外收取数据解冻费用，此费用与流出流量费用是两个独立计费项。如果Object每月平均访问频率高于1次，Object转换成低频访问、归档存储或冷归档存储类型后的使用成本可能高于标准存储类型。

- 临时存储费用

  冷归档存储类型Object在数据解冻时会生成一份标准存储类型的Object副本用于访问。该Object在解冻时间结束前会以标准存储的存储费率计算临时存储费用。

5.存储空间（Bucket）

5.1. 存储空间概述

在上传数据（例如文档、图片、音视频等）到OSS之前，您需要在OSS所支持的地域中创建一个存储空间（Bucket），然后将无限数量的对象（Object）上传到该Bucket中。

背景信息

Bucket和Object都是OSS资源，OSS提供了相关的API接口来管理这些资源。例如您可以通过API接口来创建Bucket并上传Object，您也可以通过控制台来完成这些操作。控制台使用OSS API接口发送请求到OSS。

同一个Bucket的内部是扁平的，没有文件系统的目录等概念，所有的Object都直接隶属于其对应的Bucket。Bucket的名称在OSS范围内全局唯一，且创建之后无法修改，详情请参见存储空间（Bucket）命名规范。

相关操作

下表汇总了OSS支持的Bucket相关操作。关于Object的操作，请参见对象概述。

操作	说明
创建存储空间	在上传Object到OSS之前，您需要创建一个用于存储文件的Bucket。Bucket具有各种配置属性，包括地域、访问权限以及其他元数据。 创建Bucket时，您需要综合考虑时延、成本及合规等要求选择一个合适的地域。有关OSS支持的地域列表，请参见访问域名和数据中心。
设置存储空间读写权限（ACL）	您可以在创建存储空间（Bucket）时设置存储空间的访问权限（ACL），也可以在创建Bucket后根据自己的业务需求修改存储空间的ACL，该操作只有存储空间的拥有者可以执行。
获取存储空间地域信息	您可以通过OSS API的GetBucketLocation接口获取存储空间（Bucket）所属的地域，即数据中心的物理位置信息。
列举存储空间	您可以通过设置不同的列举条件，列举某个地域下的全部或部分Bucket。
存储空间清单	您可以通过清单功能获取Bucket中指定Object的数量、大小、存储类型、加密状态等信息。相对于GetBucket (ListObjects)接口，在海量Object的列举场景中，建议您优先使用清单功能。
请求者付费模式	请求者付费模式是指由请求者支付访问Bucket内数据时产生的费用，而Bucket拥有者仅支付存储费用。当您希望共享数据，但又不希望支付因共享数据产生的额外费用时，您可以开启此功能。
绑定自定义域名	文件上传至Bucket后，OSS会自动生成文件URL，您可以直接通过文件URL访问该文件。若您希望通过自定义域名访问这些文件，需要将自定义域名绑定至文件所在的Bucket，并添加CNAME记录。
传输加速	OSS传输加速利用全球分布的云机房，将全球各地用户对您Bucket的访问，经过智能路由解析至就近的接入点，使用优化后的网络及协议，为云存储互联网的上传、下载提供端到端的加速方案。
跨域资源共享	跨域资源共享CORS（Cross-Origin Resource Sharing）简称跨域访问，是HTML5提供的标准跨域解决方案，允许Web应用服务器进行跨域访问控制，使得跨域数据传输得以安全进行。
存储空间标签	您可以通过Bucket的标签功能， 对Bucket进行分类管理，如列举带有指定标签的Bucket、对拥有指定标签的Bucket设置访问权限等。
事件通知	为Bucket配置事件时可自定义Bucket内您关注的Object，当这些Object发生指定事件时，您可以第一时间收到通知。
生命周期	生命周期规则允许您定期将Bucket内的Object转储为低频访问、归档存储或冷归档存储类型，或将过期的Object和碎片删除，从而节省存储费用。
实时日志查询	开启实时日志查询后，您可以追踪Bucket的访问请求。帮助您完成操作审计、访问统计、异常事件回溯和问题定位等工作，提升您的工作效率并更好地帮助您基于数据进行决策。
合规保留策略	OSS支持针对Bucket设置基于时间的合规保留策略。当策略锁定后，用户可以在Bucket中上传和读取Object。但是在Object的保留时间到期之前，任何用户都无法删除Object和策略。Object的保留时间到期后，才可以删除Object。
Bucket Policy	Bucket Policy是阿里云OSS推出的针对Bucket的授权策略，您可以通过Bucket Policy授权其他用户访问您指定的OSS资源。
数据复制	您可以将源Bucket的数据复制到目标Bucket，目标Bucket可以与源Bucket处于同一地域，也可以是不同地域。 • 同区域复制是指将源Bucket中的Object的创建、更新和删除等操作自动、异步（近实时）地复制到相同地域的目标Bucket。 • 跨区域复制是指将源Bucket中的Object的创建、更新和删除等操作自动、异步（近实时）地复制到不同地域的目标Bucket。
版本控制	版本控制是针对Bucket级别的数据保护功能。开启版本控制后，针对数据的覆盖和删除操作将会以历史版本的形式保存下来。您在错误覆盖或者删除Object后，能够将Bucket中存储的Object恢复至任意时刻的历史版本。

操作	说明
静态网站托管	静态网站是指所有的网页都由静态内容构成，包括客户端执行的脚本（例如JavaScript）。您可以通过静态网站托管功能将您的静态网站托管到OSS的Bucket，并使用Bucket的访问域名访问这个网站。
删除存储空间	如果您不再需要保留某个Bucket时，可将其删除。

5.2. 存储空间命名

存储空间（Bucket）是用于存储对象（Object）的容器，所有的Object都必须隶属于某个Bucket。Bucket具有各种配置属性，包括地域、访问权限、存储类型等。您可以根据实际需求，创建不同类型的Bucket来存储不同的数据。

命名规则

同一阿里云账号在同一地域内创建的Bucket总数不能超过100个。Bucket创建后，其名称无法修改。Bucket命名规则如下：

- Bucket名称在OSS范围内必须全局唯一。
- 只能包括小写字母、数字和短划线（-）。
- 必须以小写字母或者数字开头和结尾。
- 长度为3~63个字符。

命名示例

Bucket名称的正确示例如下：

- examplebucket1
- test-bucket-2021
- aliyun-oss-bucket

Bucket名称的错误示例以及错误的原因如下：

- Examplebucket1（包含了大写字母）
- test_bucket_2021（包含了下划线）
- aliyun-oss-bucket-（结尾包含了短划线）

5.3. 创建存储空间

在上传文件（Object）到OSS之前，您需要创建一个用于存储文件的存储空间（Bucket）。存储空间具有各种配置属性，包括地域、访问权限、存储类型等。您可以根据实际需求，创建不同类型的存储空间来存储不同的数据。

注意事项

- 创建存储空间本身不收取任何费用，仅收取上传至存储空间中Object的存储费用或者访问Object产生的流量费用等。更多信息，请参见计量项和计费项。
- 存储空间的容量弹性扩展，您无需提前购买容量。

使用限制

- 同一阿里云账号在同一地域内创建的存储空间总数不能超过100个。
- 存储空间名称在OSS范围内必须全局唯一。有关存储空间的命名规范，请参见存储空间命名。
- 存储空间创建后，其名称、所处地域、存储类型、冗余类型不支持修改。
- 单个存储空间的容量不限制。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击**创建Bucket**。
3. 在**创建Bucket**面板，按如下说明配置各项参数。

参数	是否必选	描述
Bucket名称	是	填写Bucket名称。命名规则如下： ○ 所选定的存储空间名称在阿里云OSS的所有现有存储空间名称中必须具有唯一性。 ○ 只能包括小写字母、数字和短划线（-）。 ○ 必须以小写字母或者数字开头和结尾。 ○ 长度必须在3~63字符之间。
地域	是	Bucket的数据中心。 如需通过ECS内网访问OSS，请选择与您ECS相同的地域。更多信息，请参见 OSS访问域名使用规则。 **说明** 如果您创建的存储空间位于中国内地，您需要在实名登记页面提交您的相关信息完成实名登记。

参数	是否必选	描述
存储类型	是	Bucket的存储类型。 ○ **标准存储**：提供高可靠、高可用、高性能的对象存储服务，能够支持频繁的数据访问。适用于各种社交、分享类的图片、音视频应用、大型网站、大数据分析等业务场景。 ○ **低频访问存储**：提供高持久性、较低存储成本的对象存储服务。有最低存储时间（30天）和最小计量单位（64 KB）要求。支持数据实时访问，访问数据时会产生数据取回费用，适用于较低访问频率（平均每月访问频率1到2次）的业务场景。 ○ **归档存储**：提供高持久性、极低存储成本的对象存储服务。有最低存储时间（60天）和最小计量单位（64 KB）要求。数据需解冻（约1分钟）后访问，解冻会产生数据取回费用。适用于数据长期保存的业务场景，例如档案数据、医疗影像、科学资料、影视素材等。 ○ **冷归档存储**：提供高持久性的对象存储服务，费用在四种存储类型最低。有最低存储时间（180天）和最小计量单位（64 KB）要求。数据需解冻后访问，解冻时间根据数据大小和选择的解冻模式决定，解冻会产生数据取回费用。适用于需要超长时间存放的极冷数据，例如因合规要求需要长期留存的数据、大数据及人工智能领域长期积累的原始数据、影视行业长期留存的媒体资源、在线教育行业的归档视频等业务场景。 **说明** 冷归档存储类型支持以下地域：华北1（青岛）、华北2（北京）、华北3（张家口）、华东1（杭州）、华东2（上海）、华南1（深圳）、西南1（成都）、华北6（乌兰察布）、中国香港、澳大利亚（悉尼）、新加坡、美国（硅谷）、德国（法兰克福）、马来西亚（吉隆坡）、印度尼西亚（雅加达）、印度（孟买）、阿联酋（迪拜）。请联系技术支持申请使用。 有关存储类型的更多信息，请参见存储类型介绍。
HDFS服务	否	如果您希望通过JindoSDK访问OSS实现数据湖场景，请先开启HDFS服务。 **注意** ○ 仅华东1（杭州）、华东2（上海）、华南1（深圳）、华北2（北京）和华北 3（张家口）地域支持开启HDFS服务，请联系技术支持申请试用。HDFS服务开启后不支持关闭，请谨慎操作。 ○ 归档以及冷归档存储类型Bucket不支持开启HDFS服务。
同城冗余存储	否	Bucket的数据容灾类型。 ○ 启用：开启后，将以同城冗余ZRS的方式存储您的OSS数据。同城冗余采用多可用区（AZ）机制，将您的数据冗余存储在同一地域（Region）的3个可用区。可支持单个可用区（机房）整体故障时（如断电、火灾等），仍然能够保障数据的正常访问。 **注意** 仅华南1（深圳）、华北2（北京）、华东1（杭州）、华东2（上海）、中国（香港）、新加坡以及印度尼西亚（雅加达）地域支持开启同城冗余存储。此外，同城冗余存储的费用较高，且开启后不支持关闭，请谨慎操作。 有关同城冗余存储的更多信息，请参见同城冗余存储。 ○ 关闭：以本地冗余LRS的方式存储您的OSS数据。本地冗余将您的数据冗余存储在同一个可用区的不同存储设备上，可支持两个存储设备并发损坏时，仍维持数据不丢失，可正常访问。
版本控制	否	选择是否开通版本控制功能。 ○ **开通**：开通Bucket版本控制功能后，针对数据的覆盖和删除操作将会以历史版本的形式保存下来。当您在错误覆盖或者删除Object后，能够将Bucket中存储的Object恢复至任意时刻的历史版本。更多详情请参见版本控制介绍。 ○ **不开通**：不开通版本控制功能，则不保存覆盖或删除的数据。
读写权限	是	选择Bucket的读写权限。 ○ **私有（private）**：只有该存储空间的拥有者可以对该存储空间内的文件进行读写操作，其他人无法访问该存储空间内的文件。 ○ **公共读（public-read）**：只有该存储空间的拥有者可以对该存储空间内的文件进行写操作，任何人（包括匿名访问者）可以对该存储空间中的文件进行读操作。 **警告** 互联网上任何用户都可以对该Bucket内文件进行访问，这有可能造成您数据的外泄以及费用激增，请谨慎操作。 ○ **公共读写（public-read-write）**：任何人（包括匿名访问者）都可以对该存储空间内文件进行读写操作。 **警告** 互联网上任何用户都可以对该Bucket内的文件进行访问，并且向该Bucket写入数据。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益。除特殊场景外，不建议您配置公共读写权限。

参数	是否必选	描述
服务端加密	否	选择是否开启服务器端加密。 ○ **服务端加密方式**：选择Object的加密方式。 ■ **无**：不启用服务器端加密。 ■ **OSS完全托管**：使用OSS托管的密钥进行加密。OSS会为每个Object使用不同的密钥进行加密，作为额外的保护，OSS会使用定期轮转的主密钥对加密密钥本身进行加密。 ■ **KMS**：使用KMS默认托管的CMK或指定CMK ID进行加解密操作。 使用KMS加密方式前，需要开通KMS服务。具体步骤，请参见开通KMS服务。 ○ **加密算法**：目前仅支持AES256。 ○ **加密密钥**：**服务端加密方式**选择**KMS**时，可配置此项。参数说明如下： ■ **alias/acs/oss**：使用默认托管的CMK生成不同的密钥来加密不同的Object，并且在Object被下载时自动解密。 ■ CMK ID：使用指定的CMK生成不同的密钥来加密不同的Object，并将加密Object的CMK ID记录到Object的元信息中，具有解密权限的用户下载Object时会自动解密。选择指定的CMK ID前，您需在KMS管理控制台创建一个与Bucket处于相同地域的普通密钥或外部密钥。详情请参见导入密钥材料。
实时日志查询	否	如果您希望在不付费的情况下实时查询最近7天的OSS访问日志，请选择**开通**。 有关实时日志查询的更多信息，请参见实时日志查询。 如果您不需要进行实时日志查询，请保持**不开通**的默认配置。
定时备份	否	如果您希望定时备份您的OSS数据，请选择**开通**。此时，OSS将自动创建备份计划，并由混合云备份HBR执行备份频率为每天备份一次OSS数据，备份文件保存一周的任务。 **注意** 如果HBR未开通或未授权HBR访问OSS，则无法创建定时备份计划。更多信息，请参见设置定时备份。 如果您不需要定时备份您的OSS数据，请保持**不开通**的默认配置。
分层命名空间	否	如果您希望快速地对目录或文件进行重命名，请开启分层命名空间。 **注意** 仅允许在创建Bucket时开启分层命名空间功能。注意开启后不支持关闭，且不支持部分功能。请参见分层命名空间查看分层命名空间不支持的功能列表。

4. 单击**确定**。

使用图形化管理工具ossbrowser

ossbrowser支持的Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成创建Bucket的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的创建存储空间代码示例。关于其他SDK的创建存储空间代码示例，请参见SDK简介。

Java

使用命令行工具ossutil

关于使用ossutil创建存储空间的具体步骤， 请参见mb（创建存储空间）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucket。

更多参考

- 文件管理
  - 上传文件

    存储空间创建完成后，您可以向该存储空间上传文件。关于上传文件的具体操作，请参见简单上传。
  - 下载文件

    文件上传完成后，您可以将文件下载至浏览器默认路径或本地指定路径。关于下载文件的具体操作，请参见简单下载。
  - 分享文件

    您还可以将文件URL分享给第三方，供其下载或预览。关于分享文件的具体操作，请参见分享文件。
- 权限控制

  默认情况下，为保证存储在OSS中数据的安全性，OSS资源默认为私有权限，只有资源拥有者或者被授权的用户允许访问。如果要授权第三方用户访问或使用自己的OSS资源，可以通过多种权限控制策略向他人授予资源的特定权限。关于访问控制的更多信息，请参见访问控制概述。

5.4. 获取存储空间的地域信息

当您在多个地域创建了大量存储空间（Bucket）时，您可以通过指定存储空间名称的方式，快速获取该存储空间所属的地域信息。

注意事项

- 关于OSS支持的地域信息，请参见访问域名和数据中心。
- 您可以通过GetBucketLocation接口返回的Location字段查看存储空间所属的地域信息。例如，华东1（杭州）的Location字段信息显示为 `oss-cn-hangzhou` 。

使用OSS控制台

1. 登录OSS管理控制台。
2. 在左侧导航栏，单击**Bucket列表**，然后单击目标Bucket名称。
   页面左上角将显示目标Bucket所属的地域信息，例如华东1（杭州）。

使用图形化管理工具ossbrowser

ossbrowser支持的Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成获取存储空间所属地域信息的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的获取存储空间所属地域信息的代码示例。关于其他SDK的获取存储空间所属地域信息的代码示例，请参见SDK简介。

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            String location = ossClient.getBucketLocation(bucketName);
            System.out.println(location);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil获取存储空间所属地域信息的具体步骤，请参见stat。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见GetBucketLocation。

5.5. 列举存储空间

存储空间（Bucket）按字母序排列。您可以结合实际场景列举当前账号下的所有存储空间、指定前缀的存储空间、指定个数的存储空间等。

列举条件

您可以通过设置prefix、marker或者max-keys参数列举满足指定条件的存储空间。

参数名称	描述
prefix	限制返回的存储空间名称必须以prefix作为前缀。如果不指定该参数，则返回所有存储空间。
marker	限制结果从marker之后按字母排序的第一个开始返回。如果不指定该参数，则从头开始返回数据。
max-keys	限定此次返回存储空间的最大个数。 取值范围：1~1000 默认值：100

使用限制

不支持使用传输加速Endpoint列举Bucket。原因是传输加速服务仅针对携带Bucket名称信息的三级域名（格式为*https://BucketName.oss-accelerate.aliyuncs.com*）提供解析服务。而列举Bucket的请求域名中不携带Bucket名称信息（格式为*https://oss-cn-hangzhou.aliyuncs.com*）。

使用OSS控制台

1. 登录OSS管理控制台。
2. 在左侧导航栏，单击**Bucket列表**。

   Bucket列表页面默认显示当前账号下的所有存储空间。如果您需要快速获取存储空间的个数以及存储空间的相关属性信息，请单击右上角的导出CSV图标 。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成列举存储空间的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的列举存储空间代码示例。关于其他SDK的列举存储空间代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.Bucket;
import java.util.List;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 列举当前账号下的所有存储空间。
            List<Bucket> buckets = ossClient.listBuckets();
            for (Bucket bucket : buckets) {
                System.out.println(" - " + bucket.getName());
            }
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil列举存储空间的具体操作， 请参见列举Bucket。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见GetService (ListBuckets)。

5.6. 分层命名空间

OSS分层命名空间功能，将对象（Object）组织到多级目录层级结构中，使得重命名目录或文件成为单个原子操作，无需列举和处理共享目录名称前缀的所有对象。与原Bucket内部扁平命名空间中使用正斜线+文件名的方式模拟文件夹相比，分层命名空间功能极大地提高了目录管理操作的性能。

功能优势

- 原子目录操作

  在OSS扁平命名空间没有真实目录的情况下，应用程序可能需要处理数百万个单独的对象才能完成目录级任务。 相比之下，分层命名空间可通过更新单个父目录批量处理多个任务。

- 性能优化

  与OSS扁平命名空间相比，分层命名空间不需要在分析数据前，对数据进行复制或转换，因此极大地提高了目录管理操作的性能。分层命名空间对于大数据分析框架尤为重要，例如Hive、Spark等工具在作业过程中通常将输出结果写入临时路径，然后在作业结束时需要对该位置进行重命名。 如果没有分层命名空间，该场景下重命名操作通常比分析过程本身更耗时。

操作方式

仅允许在创建Bucket时开启分层命名空间功能，且开启后不支持关闭。开启分层命名空间的具体操作，请参见创建存储空间。

您可以通过以下两种方式在开启了分层命名空间的Bucket中创建目录、并进行目录的重命名或删除操作。

操作方式	说明
控制台 • 创建目录 • 重命名目录或文件 • 删除目录	Web应用程序，直观易用
Java SDK	简单易用的SDK demo

注意事项

- 支持地域

  目前仅支持在澳大利亚（悉尼）、美国（硅谷）、日本（东京）、印度（孟买）、英国（伦敦）、马来西亚（吉隆坡）地域开启分层命名空间。
- 创建目录

  设置的目录绝对路径（DirectoryName）中不能出现连续的正斜线（/）。

  目录不支持传入数据，目录的Content-Length固定为0。

  目录的Content-Type固定为 `application/x-directory` ，无法修改。
- 重命名目录或文件

  重命名后的目录或文件不能与同一个Bucket下已存在的目录或者文件重名。

  重命名后的目录或文件，其父级目录必须存在。例如目录重命名为 `destfolder/examplefolder/test` ，则Bucket中必须存在 `destfolder/examplefolder` 。
- 删除目录
  - 递归删除

    递归删除用于删除目录下的文件和子目录。使用递归删除方式删除目录时，您必须拥有目录以及该目录下所有文件和子目录的DeleteObject权限。

    例如要递归删除dest目录下的子目录testfolder及其包含的所有文件，您必须拥有 `dest/testfolder` 和 `dest/testfolder/*` 的DeleteObject权限。

    > **注意** 使用递归删除方式删除目录时，如果同时存在向目录的并发写请求，则可能导致目录删除失败。
  - 非递归删除

    非递归删除用于删除空目录。使用非递归方式删除目录时，您必须拥有目录的DeleteObject权限。

不支持的功能

在开启了分层命名空间的Bucket中，不支持使用以下功能：

分类	说明
Bucket级别的功能	• 跨区域复制 • 版本控制 • 存储空间清单 • 跨域资源共享 • 静态网站托管 • 生命周期 • 合规保留策略 • 传输加速
Object级别的功能	• 软链接 • 追加上传 • 目录不支持读写权限（ACL）和标签（Tagging） • 图片处理 • 存储类型为归档或冷归档 • 回调（Callback） • 解冻归档或冷归档文件（RestoreObject） • 禁止覆盖同名文件参数（ `x-oss-forbid-overwrite` ） • 通过GetObject获取目录时不支持设置 `response-content-*` • LiveChannel相关操作 • 批量删除（DeleteMultipleObjects） • 选取文件（SelectObject）

5.7. 生命周期

5.7.1. 生命周期规则介绍

您可以基于最后一次修改时间（Last Modified Time）以及最后一次访问时间（Last Access Time）的策略创建生命周期规则，定期将存储空间（Bucket）内的多个文件（Object）转储为指定存储类型，或者将过期的Object和碎片删除，从而节省存储费用。

基于最后一次修改时间以及最后一次访问时间策略的区别说明如下：

策略	基于最后一次修改时间	基于最后一次访问时间
适用场景	适用于访问模式较固定或者可以准确预估访问模式的数据场景。	适用于访问模式不固定或者无法预估访问模式的数据场景。
是否支持删除Object	支持	不支持
Object删除后是否可以恢复	**未开启版本控制则Object删除后无法恢复** 如果未开启版本控制的Bucket配置了基于最后一次修改时间的生命周期规则，且规则指向Object的删除行为，则删除后的Object无法恢复。 如果您希望删除后的Object可以恢复，请为Object所在的Bucket开启版本控制。有关开启版本控制的具体操作，请参见版本控制介绍。Bucket开启版本控制后，基于最后一次修改时间的生命周期规则指向不同版本Object的删除行为说明如下： • 如果规则指向当前版本Object的删除行为，则OSS不会直接删除当前版本Object，而是将当前版本Object转为历史版本Object，并添加删除标记。 • 如果规则指向历史版本Object的删除行为，则直接删除历史版本Object。此外，删除历史版本Object的同时也会对标记为删除标记的Object进行删除。	不涉及
Object存储类型转换后是否可逆	不可逆。例如，将Object从标准存储类型自动转换为低频访问类型之后，不能从低频访问类型再自动转为标准类型。有关生命周期规则支持的各存储类型转换规则的说明，请参见通过生命周期规则自动转换Object的存储类型。 当Object被转换为低频访问、归档存储和冷归档存储类型后，涉及最小计量空间、最短存储时长、数据取回费用等问题，更多信息请参见注意事项。	可逆。将Object从标准存储类型自动转换为低频访问类型时，您还可以在Object被访问时选择重新返回标准存储类型。 该场景同样涉及最小计量空间、最短存储时长、数据取回费用等问题，更多信息请参见基于最后一次访问时间的生命周期规则介绍。

有关基于最后一次修改时间的更多信息，请参见基于最后一次修改时间的生命周期规则介绍。

有关基于最后一次访问时间的更多信息，请参见基于最后一次访问时间的生命周期规则介绍。

5.7.2. 基于最后一次修改时间的生命周期规则介绍

您可以通过OSS的PutBucketLifecycle接口创建基于最后一次修改时间（Last Modified Time）的生命周期规则（Lifecycle），定期将对象（Object）转储为低频访问、归档存储或冷归档存储类型，或者将过期的Object和碎片删除，从而节省存储费用。

使用场景

生命周期规则可以定期将非热门数据转换为低频访问、归档存储或冷归档存储，将不再需要访问的数据删除，让您更高效地管理您存储的数据，节省大量人力及存储成本。例如：

- 某医疗机构的医疗档案，上传至OSS后半年内需要偶尔访问，半年后基本不再访问。可以通过设置生命周期规则，将已上传180天的医疗档案转为归档存储。
- 某公司服务热线的录音文件，上传至OSS后2个月内，需要作为数据统计及核查的依据，2个月后偶尔访问，半年后基本不再访问，2年后数据不再需要存储。可以通过设置生命周期规则，设置录音文件上传60天后转为低频访问存储，180天后转为归档存储，730天后删除。
- 某存储空间内有大量文件需要全部删除，但是手动删除每次仅可以删除最多1000个文件，比较麻烦。此时可以配置一条匹配整个Bucket的生命周期规则，设置一天后删除所有文件。此Bucket内的数据会在第二天被全部删除。

使用限制

不支持对重叠前缀的Object设置两条或两条以上包含碎片过期策略的生命周期规则。示例如下：

- 示例一

  您对整个Bucket设置了一条包含碎片过期策略的生命周期规则，则不支持对Bucket中任意前缀的Object再设置一条包含碎片过期策略的生命周期规则。
- 示例二

  您对某个Bucket中前缀为*dir1*设置了一条包含碎片过期策略的生命周期规则，则不支持对该Bucket中包含重叠前缀（例如*dir1/dir2*）的Object再设置一条包含碎片过期策略的生命周期规则。

注意事项

- 费用说明
  - 请求费用

    成功的生命周期异步请求操作会记录在访问日志中并产生相关的请求次数费用，失败的操作不会被记录和收费。
  - 存储费用

    当您在OSS内存储文件时，OSS会根据您存储的文件类型、大小和时长收取一定的存储费用。当低频、归档、冷归档存储类型文件在不足规定时长时通过生命周期策略提前删除，还会产生不足规定时长容量费用。
    - 通过生命周期将文件存储类型转换为低频或归档，且在不足规定时长前删除文件

      低频访问类型最低存储时间（30天）和归档类型最低存储时间（60天）均以文件存储在OSS的Last Modified时间开始计算。例如标准类型文件在其创建10天后，通过生命周期将其转换为低频访问类型，过了20天后将其转换为归档类型，再过5天将其删除。此时会产生25天的归档存储不足规定时长容量费用。
    - 通过生命周期将文件存储类型转换为冷归档，且在不足规定时长前删除文件

      冷归档的最低存储时间（180天）以文件转为冷归档类型的时间开始计算。例如标准类型文件在其创建10天后，通过生命周期规则转换为冷归档存储类型，再过1天将其删除。此时会产生179天的冷归档存储不足规定时长容量费用。

    关于各存储类型的费用详情，请参见存储费用。
- 数量

  通过控制台最多可配置100条生命周期规则，通过SDK或者命令行工具ossutil最多可配置1000条生命周期规则。
- 生效时间

生命周期规则创建后的24小时内，OSS会加载规则。规则加载完成后，OSS会在每天的北京时间8:00开始执行规则，并在随后的24小时内执行完毕。Object的最后修改时间与生命周期规则开始执行时间（8:00）必须间隔24小时以上。例如生命周期规则为Object上传1天后删除，则2020年7月20日上传的文件删除时间如下：

- 北京时间8:00前上传的文件会在2020年7月21日8:00开始删除，并在7月22日8:00前删除完毕。
- 北京时间8:00后上传的文件会在2020年7月22日8:00开始删除，并在7月23日8:00前删除完毕。

> **注意** 更新生命周期规则会中止当天的生命周期任务，请不要频繁更新生命周期规则。

组成元素

生命周期规则由以下元素组成：

- 策略：生命周期规则匹配的Object和碎片。
  - 按前缀匹配：按指定前缀匹配Object和碎片。可创建多条规则匹配不同的前缀，前缀不能重复。前缀的命名规范与Object命名规范相同，详情请参见对象（Object）。
  - 按标签匹配：按指定标签的Key和Value匹配Object。单条规则可配置多个标签，OSS对所有拥有这些标签的对象执行生命周期规则。标签匹配规则不作用于碎片。

    > **说明** 对象标签功能详情请参见对象标签。

  - 按前缀+标签匹配：按指定前缀和标签的筛选条件匹配对象。
  - 配置到整个Bucket：匹配整个Bucket内的所有Object和碎片。配置了覆盖整个Bucket的生命周期规则后，不支持再创建其他生命周期规则。
- 文件过期策略：设置Object的过期时间及操作。
  - 过期天数：指定一个过期天数N，并指定非版本状态下的所有Object、以及版本控制状态下的当前版本Object过期后执行什么操作。Object会在其最后修改时间的N天后过期，并执行指定的操作。
  - 过期日期：指定一个过期日期，并指定非版本状态下的所有Object、以及版本控制状态下的当前版本Object过期后执行什么操作。最后修改时间在该日期之前的Object全部过期，并执行指定的操作。
  - Object成为非当前版本天数：指定一个过期天数N，并指定非当前版本Object过期后执行什么操作。Object会在其成为非当前版本的N天后过期，并执行指定的操作。

  您可以转换过期Object的存储类型或将其删除。详情请参见生命周期配置元素。
- 碎片过期策略：设置碎片的过期时间及操作。
  - 过期天数：可指定一个过期天数N，文件碎片会在其最后修改时间的N天后被删除。
  - 过期日期：指定一个过期日期，最后修改时间在该日期之前的文件碎片会被全部删除。

规则说明

- 规则生效

  例如，某个Bucket有如下几个Object：

  ```
  logs/program.log.1
  logs/program.log.2
  logs/program.log.3
  doc/readme.txt
  ```

  如果生命周期规则指定的前缀是*logs/*，那么此规则仅作用于前三个以*logs/*开头的Object；如果指定的前缀是*doc/readme.txt*，则此规则则只对*doc/readme.txt*起作用。

  对过期策略匹配的Object执行GET或HEAD操作时，OSS会在响应Header中加入 `x-oss-expiration` 头。其中 `expiry-date` 的值表示Object的过期日期； `rule-id` 的值表示相匹配的规则ID。
- 规则冲突
  - 相同前缀和标签

    当不同生命周期规则作用于相同前缀和标签的Object时，删除操作优先于存储类型转换操作。rule1用于指定所有前缀为abc，标签为a=1的Object 20天后删除，rule2规则不生效。

rule	prefix	tag	action
rule1	abc	a=1	20天后删除
rule2	abc	a=1	20天后转为Archive

  - 前缀重叠+标签相同

    rule1用于指定所有标签为a=1的Object 10天后转为IA。rule2用于指定前缀为abc且标签为a=1的Object 120天后删除。

rule	prefix	tag	action
rule1	-	a=1	10天后转为IA
rule2	abc	a=1	120天后被删除

    rule3用于指定所有标签为a=1的Object 20天后转为Archive。由于Archive类型文件无法转换为IA类型，因此rule4指定的前缀为abc且标签为a=1的Object 30天后转为IA的规则不生效。

rule	prefix	tag	action
rule3	-	a=1	20天后转为Archive
rule4	abc	a=1	30天后转为IA

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。

3. 选择**基础设置 > 生命周期**，在**生命周期**区域单击**设置**。
4. 如果您需要创建基于最后一次访问时间策略的生命周期规则，请在**生命周期**页面打开**启用访问追踪**开关。
5. 单击**创建规则**，在**创建生命周期规则**面板，按如下说明配置生命周期规则。
   - 存储空间未开启版本控制

区域	配置项	说明
基本设置	**状态**	设置生命周期规则的状态，可选择**启动**或**禁用**。
	策略	选择生命周期规则作用的Object。您可以选择**按前缀匹配**或**配置到整个Bucket**。 注意 ■ 如果先配置一条针对整个Bucket的生命周期规则，则不支持再配置按前缀匹配的生命周期规则。 ■ 如果先配置一条或多条按前缀匹配的生命周期规则，则允许再配置一条针对整个Bucket的生命周期规则。
	前缀	输入规则要匹配的Object名称的前缀。例如，您需要匹配名称以img开头的Object，则输入*img*。
	标签	生命周期规则仅针对拥有指定标签Object生效。例如选择了**按前缀匹配**，设置前缀为img，并设置标签的key为a，value为1。则该规则将匹配所有名称以img开头，标签为a=1的Object。关于对象标签的更多信息，请参见对象标签。
文件执行策略设置	**文件过期策略**	选择Object过期策略，可选择**过期天数**、**过期日期**和**不启用**。选择**不启用**时，文件过期策略不生效。
	生命周期管理规则	配置转换Object存储类型或者删除过期Object的规则。 配置示例一：当您选择了最后一次**访问时间**策略，然后将**过期天数**设置为30，并指定数据在超出指定过期天数后将自动转换为**低频存储类型（数据被访问后，依旧保留在低频档）**，则最后访问日期为2021年9月1日的Object会在2021年10月1日被转换为指定的存储类型。 配置示例二：当您选择了最后一次**修改时间**策略，然后将**过期日期**设置为2021年9月24日，并指定在指定日期前的数据自动删除，则最后修改日期在2021年9月24日之前的Object会被自动删除，且删除后不可恢复。
碎片执行策略设置	**碎片过期策略**	设置对过期碎片执行的操作。如果选中了**标签**，则无法配置该选项。您可以选择碎片过期策略的**过期天数**或**过期日期**，也可以选择**不启用**碎片过期策略。当选择**不启用**时，碎片过期策略不生效。 注意 生命周期规则至少包含文件过期策略或碎片过期策略。
	碎片规则	根据碎片过期策略选择的过期天数或过期日期设定碎片何时过期，碎片过期后会被自动删除，且删除后不可恢复。

   - 存储空间已开启版本控制

     开启版本控制后，**基本设置**与**碎片执行策略设置**区域涉及的配置项，与未开启版本控制的配置方法相同。以下表格仅介绍与未开启版本控制相比，开启版本控制后配置项存在的差异。

区域	配置项	说明
当前版本文件执行策略设置	**清理对象删除标记**	开启版本控制后，清除策略中增加了**清理对象删除标记**选项，其他选项与未开启版本控制时相同。 选择此选项后，如果当前Object仅有一个版本且为删除标记时，则OSS将删除过期Object的删除标记。如果当前Object有多个版本，且Object的最新版本为删除标记时，则OSS将保留该删除标记。关于删除标记的更多信息，请参见删除标记。
历史版本文件执行策略设置	**文件过期策略**	设置历史版本文件的过期策略，可选择**过期天数**和**不启用**。当选择**不启用**时，文件过期策略不生效。
	生命周期管理规则	设定一个过期天数N，历史版本的Object会在其被转换为历史版本的N天后过期，并在过期的第二天执行指定操作。例如设置为30，则在2021年9月1日被转为历史版本的Object会在2021年10月1日被转换为指定存储类型或被删除。 注意 您可以通过Object下一个版本的最后一次修改时间确定Object被转为历史版本的时间。

6. 单击**确定**。

   生命周期规则保存成功后，您可以在策略列表中查看已设置的生命周期规则。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成Bucket生命周期规则配置操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的配置生命周期规则的代码示例。关于其他SDK的配置生命周期规则的代码示例，请参见SDK简介。

Java

```
OSS ossClient = null;
try {
    // yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
    String endpoint = "yourEndpoint";
    // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
    String accessKeyId = "yourAccessKeyId";
    String accessKeySecret = "yourAccessKeySecret";
```

```
String accessKeySecret = "yourAccessKeySecret";
// 填写Bucket名称，例如examplebucket。
String bucketName = "examplebucket";
// 创建OSSClient实例。
ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 创建SetBucketLifecycleRequest。
SetBucketLifecycleRequest request = new SetBucketLifecycleRequest(bucketName);
// 设置规则ID。
String ruleId0 = "rule0";
// 设置文件匹配前缀。
String matchPrefix0 = "A0/";
// 设置要匹配的标签。
Map<String, String> matchTags0 = new HashMap<String, String>();
// 依次填写要匹配标签的键（例如owner）和值（例如John）。
matchTags0.put("owner", "John");
String ruleId1 = "rule1";
String matchPrefix1 = "A1/";
Map<String, String> matchTags1 = new HashMap<String, String>();
matchTags1.put("type", "document");
String ruleId2 = "rule2";
String matchPrefix2 = "A2/";
String ruleId3 = "rule3";
String matchPrefix3 = "A3/";
String ruleId4 = "rule4";
String matchPrefix4 = "A4/";
String ruleId5 = "rule5";
String matchPrefix5 = "A5/";
String ruleId6 = "rule6";
String matchPrefix6 = "A6/";
// 距最后修改时间3天后过期。
LifecycleRule rule = new LifecycleRule(ruleId0, matchPrefix0, LifecycleRule.RuleStatus.Enabled, 3);
rule.setTags(matchTags0);
request.AddLifecycleRule(rule);
// 指定日期之前创建的文件过期。
rule = new LifecycleRule(ruleId1, matchPrefix1, LifecycleRule.RuleStatus.Enabled);
rule.setCreatedBeforeDate(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
rule.setTags(matchTags1);
request.AddLifecycleRule(rule);
// 分片3天后过期。
rule = new LifecycleRule(ruleId2, matchPrefix2, LifecycleRule.RuleStatus.Enabled);
LifecycleRule.AbortMultipartUpload abortMultipartUpload = new LifecycleRule.AbortMultipartUpload();
abortMultipartUpload.setExpirationDays(3);
rule.setAbortMultipartUpload(abortMultipartUpload);
request.AddLifecycleRule(rule);
// 指定日期之前的分片过期。
rule = new LifecycleRule(ruleId3, matchPrefix3, LifecycleRule.RuleStatus.Enabled);
abortMultipartUpload = new LifecycleRule.AbortMultipartUpload();
abortMultipartUpload.setCreatedBeforeDate(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
rule.setAbortMultipartUpload(abortMultipartUpload);
request.AddLifecycleRule(rule);
// 距最后修改时间10天后转低频访问存储类型，距最后修改时间30天后转归档存储类型。
rule = new LifecycleRule(ruleId4, matchPrefix4, LifecycleRule.RuleStatus.Enabled);
List<LifecycleRule.StorageTransition> storageTransitions = new ArrayList<LifecycleRule.StorageTransition>();
LifecycleRule.StorageTransition storageTransition = new LifecycleRule.StorageTransition();
storageTransition.setStorageClass(StorageClass.IA);
storageTransition.setExpirationDays(10);
storageTransitions.add(storageTransition);
storageTransition = new LifecycleRule.StorageTransition();
storageTransition.setStorageClass(StorageClass.Archive);
storageTransition.setExpirationDays(30);
storageTransitions.add(storageTransition);
rule.setStorageTransition(storageTransitions);
request.AddLifecycleRule(rule);
// 指定最后修改日期在2022-10-12之前的文件转为归档存储。
rule = new LifecycleRule(ruleId5, matchPrefix5, LifecycleRule.RuleStatus.Enabled);
storageTransitions = new ArrayList<LifecycleRule.StorageTransition>();
storageTransition = new LifecycleRule.StorageTransition();
storageTransition.setCreatedBeforeDate(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
storageTransition.setStorageClass(StorageClass.Archive);
storageTransitions.add(storageTransition);
rule.setStorageTransition(storageTransitions);
request.AddLifecycleRule(rule);
// rule6针对版本控制状态下的Bucket。
rule = new LifecycleRule(ruleId6, matchPrefix6, LifecycleRule.RuleStatus.Enabled);
// 设置Object相对最后修改时间365天之后自动转为归档文件。
storageTransitions = new ArrayList<LifecycleRule.StorageTransition>();
storageTransition = new LifecycleRule.StorageTransition();
storageTransition.setStorageClass(StorageClass.Archive);
storageTransition.setExpirationDays(365);
storageTransitions.add(storageTransition);
rule.setStorageTransition(storageTransitions);
// 设置自动移除过期删除标记。
rule.setExpiredDeleteMarker(true);
// 设置非当前版本的object距最后修改时间10天之后转为低频访问类型。
LifecycleRule.NoncurrentVersionStorageTransition noncurrentVersionStorageTransition =
        new LifecycleRule.NoncurrentVersionStorageTransition().withNoncurrentDays(10).withStrorageClass(StorageClass.IA);
// 设置非当前版本的Object距最后修改时间20天之后转为归档类型。
```

```
    LifecycleRule.NoncurrentVersionStorageTransition noncurrentVersionStorageTransition2 =
            new LifecycleRule.NoncurrentVersionStorageTransition().withNoncurrentDays(20).withStrorageClass(StorageClass.Archive);
    // 设置非当前版本Object 30天后删除。
    LifecycleRule.NoncurrentVersionExpiration noncurrentVersionExpiration = new LifecycleRule.NoncurrentVersionExpiration().withNoncurrentDays(30);
    List<LifecycleRule.NoncurrentVersionStorageTransition> noncurrentVersionStorageTransitions = new ArrayList<LifecycleRule.NoncurrentVersionStorage
Transition>();
    noncurrentVersionStorageTransitions.add(noncurrentVersionStorageTransition2);
    rule.setStorageTransition(storageTransitions);
    rule.setNoncurrentVersionExpiration(noncurrentVersionExpiration);
    rule.setNoncurrentVersionStorageTransitions(noncurrentVersionStorageTransitions);
    request.AddLifecycleRule(rule);
    // 发起设置生命周期规则请求。
    ossClient.setBucketLifecycle(request);
    // 查看生命周期规则。
    List<LifecycleRule> listRules = ossClient.getBucketLifecycle(bucketName);
    for(LifecycleRule rules : listRules){
        System.out.println("ruleId="+rules.getId()+", matchPrefix="+rules.getPrefix());
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if(ossClient != null){
        // 关闭OSSClient。
        ossClient.shutdown();
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置生命周期规则的具体操作， 请参见添加或修改生命周期规则。

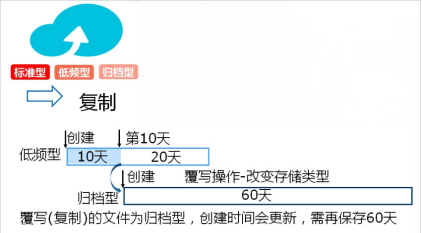
使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketLifecycle。

常见问题

- 通过CopyObject覆写文件存储类型时，是否有最小存储天数限制？

  有最小存储天数限制。例如低频访问类型文件在其创建10天后，通过CopyObject覆写操作将存储类型转换为归档存储，此操作会产生低频访问存储不足规定时长费用。同时文件的创建时间发生变化，且转换后的归档存储类型文件因最小存储天数为60天，因此需再保存至少60天。



- 通过生命周期规则进行的类型转换、过期删除操作，是否有日志记录？

  所有成功通过生命周期规则进行的类型转换、过期删除操作都会有日志记录，日志记录字段如下：

  - Operation
    - CommitTransition：通过生命周期规则转换存储类型，例如转换为低频访问、归档或冷归档存储类型。
    - ExpireObject：通过生命周期规则删除过期Object。
  - Sync Request

    lifecycle：生命周期规则触发的转换存储类型和删除过期Object的操作。

5.7.3. 基于最后一次访问时间的生命周期规则介绍

您可以通过基于最后一次访问时间（Last Access Time）策略的生命周期规则来自动监测数据的访问模式并识别冷数据，然后将识别出来的冷数据进行存储类型的转换，从而达到数据的冷热分层存储，最终降低存储成本。

使用场景

- 多媒体场景

  某网站的视频、图片存储在OSS上，历史数据会逐渐从热转冷。因此，您可能需要将网站内长时间不被访问的数据保存为低频访问类型。此外，部分数据距离上传时间已久，但仍然是热门访问数据，这部分数据需要继续保存为标准存储类型。在该场景下应选用基于最后一次访问时间的生命周期规则，用于自动识别冷热数据并进行分层存储，从而降低存储成本。

- 相册或网盘场景

  对于长时间没有访问的冷数据，希望设置自定义转储天数，自动将冷数据转为低频访问类型，并确保数据的实时访问。

- 生命科学场景

  基因测序生成的大量业务数据，往往需要根据数据的最后访问时间而非最后修改时间来判断数据的冷热。按以往，客户只能手动通过日志分析或其他方式进行数据冷热的分层管理。但如果选用基于最后一次访问时间的生命周期规则，则可实现由服务端根据最后访问时间来自动识别冷热数据并实现数据分层存储。不仅如此，您还可以在同一条生命周期规则中同时结合最后访问时间与最后修改时间的策略，从而更灵活地进行数据管理。

使用限制

不支持对重叠前缀的Object设置两条或两条以上包含碎片过期策略的生命周期规则。示例如下：

- 示例一

  您对整个Bucket设置了一条包含碎片过期策略的生命周期规则，则不支持对Bucket中任意前缀的Object再设置一条包含碎片过期策略的生命周期规则。

- 示例二

  您对某个Bucket中前缀为*dir1*设置了一条包含碎片过期策略的生命周期规则，则不支持对该Bucket中包含重叠前缀（例如*dir1/dir2*）的Object再设置一条包含碎片过期策略的生命周期规则。

注意事项

- 支持地域

  仅华北1（青岛）、华北5（呼和浩特）、德国（法兰克福）以及澳大利亚（悉尼）地域支持创建基于最后一次访问时间的生命周期规则。

- 规则数量

  通过控制台最多可配置100条生命周期规则，单条生命周期规则中可同时包含最后一次修改时间以及最后一次访问时间的策略。如果您需要配置更多数量的生命周期规则，请使用SDK或者命令行工具ossutil。

- 最后一次访问时间更新策略

  开启访问追踪后，OSS默认以访问跟踪开启时间作为Bucket中所有Object的最后一次访问时间，后续会根据Object的访问情况自动更新Object的最后一次访问时间。如果24小时内，同一个Object有多次Get Object请求，则OSS会将首次Get Object的请求时间记录为Object最后一次访问时间。

  此外，访问目标Object对应的软链接时不会更新目标Object的最后一次访问时间。

- 转储的Object类型

  基于最后一次访问时间的生命周期规则支持将Object从标准存储类型转为低频访问类型，还可以选择当Object被访问后是否自动转回标准存储类型。不支持转储为归档或冷归档存储类型。

- 费用说明
  - Object监控管理费用

    开启访问追踪后会产生Object监控管理费，但OSS暂不收取该费用。

  - 存储费用

    您可以针对任意大小的Object设置基于最后一次访问时间的生命周期规则，OSS会根据Object所处的存储类型收取不同的存储费用。当Object处于标准存储类型时，按照标准存储容量收费。当Object处于低频访问类型时，按照低频访问容量收费。但是，当Object小于64 KB时，按照64 KB计算。当Object大于或等于64 KB时，按照实际大小计算。

  - 低频访问不足规定时长容量费用

    低频访问类型Object有最低30天的存储时长要求。如果存储时长未达到最低天数要求，还会产生不足规定时长容量费用。该计费项结合生命周期规则的示例说明如下：

    示例一：标准类型Object在其创建10天后，通过生命周期将其转换为低频访问类型，过了5天后又将其转回标准存储。此时会产生15天的低频访问不足规定时长容量费用。

    示例二：标准类型Object在其创建10天后，通过生命周期将其转换为低频访问类型，过了15天后将其删除。此时会产生5天的低频访问不足规定时长容量费用。

    通过CopyObject将Object覆写为低频访问类型时，Object也有最小存储天数限制。例如标准存储类型文件在其创建10天后，通过CopyObject覆写操作将存储类型转换为低频存储，再过10天后将其删除。此时会产生20天的低频访问存储不足规定时长费用。

  - 低频访问数据取回费用

    访问低频访问类型文件产生的费用，按数据取回量计费。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 选择**基础设置** > **生命周期**，在**生命周期**区域单击**设置**。
4. 如果您需要创建基于最后一次访问时间策略的生命周期规则，请在**生命周期**页面打开**启用访问追踪**开关。
5. 单击**创建规则**，在**创建生命周期规则**面板，按如下说明配置生命周期规则。
   - 存储空间未开启版本控制

区域	配置项	说明
基本设置	**状态**	设置生命周期规则的状态，可选择**启动**或**禁用**。
	策略	选择生命周期规则作用的Object。您可以选择**按前缀匹配**或**配置到整个Bucket**。 **注意** ■ 如果先配置一条针对整个Bucket的生命周期规则，则不支持再配置按前缀匹配的生命周期规则。 ■ 如果先配置一条或多条按前缀匹配的生命周期规则，则允许再配置一条针对整个Bucket的生命周期规则。
	前缀	输入规则要匹配的Object名称的前缀。例如，您需要匹配名称以img开头的Object，则输入*img*。
	标签	生命周期规则仅针对拥有指定标签Object生效。例如选择了**按前缀匹配**，设置前缀为img，并设置标签的key为a，value为1。则该规则将匹配所有名称以img开头，标签为a=1的Object。关于对象标签的更多信息，请参见对象标签。
	文件过期策略	选择Object过期策略，可选择**过期天数**、**过期日期**和**不启用**。选择**不启用**时，文件过期策略不生效。

区域	配置项	说明
文件执行策略设置		
	生命周期管理规则	配置转换Object存储类型或者删除过期Object的规则。 配置示例一：当您选择了最后一次**访问时间**策略，然后将**过期天数**设置为30，并指定数据在超出指定过期天数后将自动转换为**低频存储类型（数据被访问后，依旧保留在低频档）**，则最后访问日期为2021年9月1日的Object会在2021年10月1日被转换为指定的存储类型。 配置示例二：当您选择了最后一次**修改时间**策略，然后将**过期日期**设置为2021年9月24日，并指定在指定日期前的数据自动删除，则最后修改日期在2021年9月24日之前的Object会被自动删除，且删除后不可恢复。
碎片执行策略设置	**碎片过期策略**	设置对过期碎片执行的操作。如果选中了**标签**，则无法配置该选项。您可以选择碎片过期策略的**过期天数**或**过期日期**，也可以选择**不启用**碎片过期策略。当选择**不启用**时，碎片过期策略不生效。 **注意** 生命周期规则至少包含文件过期策略或碎片过期策略。
	碎片规则	根据碎片过期策略选择的过期天数或过期日期设定碎片何时过期，碎片过期后会被自动删除，且删除后不可恢复。

- 存储空间已开启版本控制

  开启版本控制后，**基本设置**与**碎片执行策略设置**区域涉及的配置项，与未开启版本控制的配置方法相同。以下表格仅介绍与未开启版本控制相比，开启版本控制后配置项存在的差异。

区域	配置项	说明
当前版本文件执行策略设置	**清理对象删除标记**	开启版本控制后，清除策略中增加了**清理对象删除标记**选项，其他选项与未开启版本控制时相同。 选择此选项后，如果当前Object仅有一个版本且为删除标记时，则OSS将删除过期Object的删除标记。如果当前Object有多个版本，且Object的最新版本为删除标记时，则OSS将保留该删除标记。关于删除标记的更多信息，请参见删除标记。
历史版本文件执行策略设置	**文件过期策略**	设置历史版本文件的过期策略，可选择**过期天数**和**不启用**。当选择**不启用**时，文件过期策略不生效。
	生命周期管理规则	设定一个过期天数N，历史版本的Object会在其被转换为历史版本的N天后过期，并在过期的第二天执行指定操作。例如设置为30，则在2021年9月1日被转为历史版本的Object会在2021年10月1日被转换为指定存储类型或被删除。 **注意** 您可以通过Object下一个版本的最后一次修改时间确定Object被转为历史版本的时间。

6. 单击**确定**。

   生命周期规则保存成功后，您可以在策略列表中查看已设置的生命周期规则。

常见问题

- 创建的生命周期规则为什么没有即刻生效？

  生命周期规则创建后的24小时内，OSS会加载规则。规则加载完成后，OSS会在每天的北京时间8:00开始执行规则，并在随后的24小时内执行完毕。因此，生命周期规则创建后最多可能需要48小时才会生效。

- 如果针对Bucket内相同前缀的Object创建了两条生命周期规则，其中一条规则基于最后一次修改时间，另外一条规则基于最后一次访问时间，最终执行效果会怎么样？

  例如，针对目标存储空间examplebucket创建了两条生命周期规则，规则一指定该Bucket内所有前缀为doc的Object在距离最后一次修改时间30天后删除，规则二指定该Bucket内所有前缀为doc的Object在距离最后一次访问时间30天后转低频访问类型。

  由于OSS执行生命周期规则时遵循以用户最低开销为原则，因此仅规则一生效。原因是规则一中指定30天后直接删除与前缀匹配的Object，此后将不再产生任意费用。而规则二转为低频访问类型仍会收取相关存储费用或者数据取回费用等。

- 已配置的生命周期规则变更后何时生效，原有规则命中的数据如何处理？

  例如，您已经针对前缀为 `er` 的Object配置了距离最后一次访问时间30天后转低频、又过了30天后当Object被访问时选择将其转为标准存储类型的生命周期规则。但是在距离最后一次访问时间的35天后，您将生命周期指定的前缀 `er` 变更为 `re` ，此时原有Object仅转存为低频访问类型，转存为标准存储类型的行为不生效。变更规则命中Object的最后一次访问时间也是从Bucket开启访问追踪时开始统计。

- 在已开启版本控制的Bucket内开启智能分层，Bucket内不同版本的Object存储层级如何分布？

  在已开启版本控制Bucket内的每一个Object都有唯一的版本ID，且不同版本ID的Object相互独立。因此可能会出现历史版本Object为低频访问类型，但是最新版本Object为标准存储类型的情况。

- 是否支持关闭访问追踪？

  支持，前提是当前Bucket不存在基于最后一次访问时间的生命周期规则。关闭访问追踪后，系统将停止追踪Object的最后一次访问时间信息。待下一次开启访问追踪后，将重新刷新所有Object的最后一次访问时间。

5.7.4. 生命周期配置元素

本文介绍对象（Object）生命周期基本示例中涉及的各个配置元素。

生命周期配置为XML格式，举例如下：

```
<LifecycleConfiguration>
<Rule>
 <ID>rule1</ID>
 <Prefix>logs/</Prefix>
 <Status>Enabled</Status>
 <Expiration>
   <Days>10</Days>
  </Expiration>
</Rule>
<Rule>
 <ID>rule2</ID>
 <Prefix>doc/</Prefix>
 <Status>Disabled</Status>
 <Expiration>
   <CreatedBeforeDate>2017-12-31T00:00:00.000Z</CreatedBeforeDate>
 </Expiration>
</Rule>
<Rule>
 <ID>rule3</ID>
 <Tag><Key>xx</Key><Value>1</Value></Tag>
 <Status>Enabled</Status>
 <Transition>
   <Days>60</Days>
   <StorageClass>Archive</StorageClass>
</Transition>
</Rule>
</LifecycleConfiguration>
```

上述示例中，有三条规则，含义如下：

- 第一条规则会删除前缀为logs/，且最后更新时间是10天前的Object。
- 第二条规则虽然指定了删除2017年12月31日之前被修改的前缀为doc/的Object，但是由于它的Status是Disabled状态，所以该规则并不会生效。
- 第三条规则会将标签为xx=1，且最后更新时间是60天前的Object存储类型修改为Archive（归档存储）。

生命周期规则涉及的各项配置元素如ID元素、操作元素等将在下文提供详细介绍。

ID元素

为存储空间配置的生命周期规则ID。最多由255个字节组成。如没有指定，或者该值为空时，OSS会自动生成一个唯一ID。

Status元素

表示生命周期规则所处的状态。您可以选择启用（Enabled）或禁用（Disabled）生命周期规则。如果规则处于禁用状态，则OSS不会执行规则中定义的任何操作。

Prefix元素

基于您指定的<Prefix>元素，将生命周期规则应用于存储空间中的所有或部分Object。

时间元素

- 按指定日期

  使用子元素<CreatedBeforeDate>指定绝对的日期，按照最后修改时间在该日期之前的Object，执行过期（Expiration）或转换（Transition）存储类型操作。

- 按指定天数

  使用子元素<Days>指定相对天数，并指定Object在其最后修改时间的N天后，执行过期或转换存储类型操作。

操作元素

通过在生命周期规则中指定的一个或多个操作元素，您可以指示OSS在Object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。以下总结了Object执行的生命周期配置规则操作的行为与包含Object的存储空间的版本控制状态的关系。

- 未开启版本控制的Bucket

操作	说明
Transition	达到Object生命周期中指定的日期或时间段时，将Object转换为指定存储类型（StorageClass）。有关生命周期支持转换的存储类型的更多信息，请参见通过生命周期规则自动转换Object的存储类型。
Expiration	达到Object生命周期中指定的日期或时间段时，永久删除符合条件的Object。

- 受版本控制的Bucket

  对受版本控制（即已开启或暂停版本控制）的Bucket中Object生命周期的相关元素说明如下：

- 当前版本Object过期或转换操作

操作	说明
Expiration	根据当前版本Object是否为删除标记，其过期行为说明如下： ■ 当前版本Object不为删除标记： ■ 在开启版本控制状态下，OSS将会插入具有唯一版本ID的删除标记作为当前版本，原当前版本将成为非当前版本。 ■ 在暂停版本控制状态下，OSS将会插入null版本ID的删除标记作为当前版本，而原当前版本中版本ID为null的版本将被覆盖，以保证一个Object只有一个版本ID为null的版本。 ■ 当前版本Object为删除标记： ■ 且在该Object还有一个或多个非当前版本状态下进行Expiration过期操作，OSS将不执行任何操作。 ■ 且在该Object仅有一个版本的状态下，进行Expiration过期操作或者将Expiration规则的子元素ExpiredObjectDeleteMarker的值设置为true，OSS都将自动移除该删除标记。 注意 ■ Object当前版本经过生命周期、或主动发起的非指定版本的删除操作时，当前版本将被置为非当前版本。所有非当前版本Object经过生命周期、或主动发起的指定版本的删除操作后均被永久删除，此时仅剩下唯一的删除标记作为当前版本，即过期删除标记。 ■ 子元素ExpiredObjectDeleteMarker不能与标签规则同时配置。
Transition	达到生命周期中指定的日期或时间段时，将当前版本Object转换为指定存储类型。

- 非当前版本Object过期或转换操作

操作	说明	关联的子元素
NoncurrentVersionExpiration	非当前版本过期操作。	子元素<NoncurrentDays>指相对过期时间，表示该版本从当前版本变成非当前版本开始，到永久删除之间的保留时间段。 说明 例如该版本原本为当前版本，2019年5月1日由于PutObject覆盖操作，该版本变成了非当前版本。NoncurrentVersionExpiration元素设置了<NoncurrentDays>过期时间为3天，那么该版本将在2019年5月4日被彻底删除。由于版本的连续性，每次执行上传操作时，当前版本会被置为第一个非当前版本，新添加的版本成为其后继版本。OSS会通过查看其后继版本的创建时间，来获取一个版本成为非当前版本的开始时间。
NoncurrentVersionTransition	非当前版本Object存储类型转换操作。	■ 子元素< NoncurrentDays>指相对转换时间，表示该版本从当前版本变成非当前版本开始，到进行存储类型转换之间的时间段。 ■ 子元素<StorageClass>用来指定Object转储的存储类型。

5.7.5. 生命周期配置示例

本文档提供多个常见的生命周期配置示例，以便您更好地使用生命周期规则管理您存储空间（Bucket）内的文件（Object）。

指定筛选条件

每个生命周期规则都包含至少一个筛选条件，筛选条件可用于确定Bucket中适用生命周期规则的部分或所有Object。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prefix为 `doc/` ，此规则将应用于prefix为 `doc/` 的Object，例如 `doc/test1.txt` 和 `doc/test2.jpg` 等Object，并指定在Object最后修改时间超过180天后将其转换为IA存储类型（Transition操作）、在Object最后修改时间超过365天后将其删除（Expiration操作）。

  以上生命周期配置规则的XML以及控制台配置示例如下：

  - XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule0</ID>
    <Prefix>doc/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
    <Transition>
      <Days>180</Days>
      <StorageClass>IA</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

示例1

- 指定生命周期规则应用于某个Bucket内的所有Object，并指示该Bucket内的所有Object在距其最后修改时间超过300天后过期。

  以上生命周期配置规则的XML以及控制台配置示例如下：

  - XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule1</ID>
    <Prefix></Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>300</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

  - 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

示例2

- 指定生命周期规则应用于某个Bucket内的所有Object（即Prefix为空），并指示该Bucket内的所有最后修改时间早于2021年12月30日的Object过期。

  以上生命周期配置规则的XML以及控制台配置示例如下：

  - XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule0</ID>
    <Prefix></Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <CreatedBeforeDate>2021-12-30T00:00:00.000Z</CreatedBeforeDate>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

- 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

示例3

重叠的筛选条件

以下说明了筛选条件重叠的情况下，是否造成生命周期操作冲突的情况。

- 假设您指定了两条生命周期规则，规则详情如下：
  - 规则1：指定了基于标签的筛选条件（tag1/value1），并指定Object在其最后修改时间超过180天后转换为IA存储类型。
  - 规则2：指定了基于标签的筛选条件（tag2/value2），并指定Object在其最后修改时间超过10天后过期。

  如果存在带有两组标签的Object，即两个规则都将应用于相同Object。在这种情况下，Object将在其最后修改时间超过10天后过期。Object被删除后，转换存储类型操作将不再有效。因此，仅规则2中指定的过期行为生效。

  以上生命周期配置规则的XML以及控制台配置示例如下：

  - XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule1</ID>
    <Prefix></Prefix>
    <Tag>
      <Key>tag1</Key>
      <Value>value1</Value>
    </Tag>
    <Status>Enabled</Status>
    <Transition>
      <Days>180</Days>
      <StorageClass>IA</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>test-rule2</ID>
    <Prefix></Prefix>
    <Tag>
      <Key>tag2</Key>
      <Value>value2</Value>
    </Tag>
    <Status>Enabled</Status>
    <Expiration>
      <Days>10</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

- 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

  - 规则1

  - 规则2

示例1（冲突）

- 假设您配置了包含两个指定重叠前缀的生命周期规则，规则详情如下：
  - 规则1：指定名为test/的Prefix，并指定Object距其最后修改时间超过30天后转换为Archive存储类型。
  - 规则2：指定针对整个Bucket（即Prefix为空），并指定Object距其最后修改时间超过365天后全部删除。

  由于规则无冲突，因此规则1和规则2指定的行为均生效。

  以上生命周期配置规则的XML以及控制台配置示例如下：

  - XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule1</ID>
    <Prefix>test/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>Archive</StorageClass>
    </Transition>
  </Rule>
 <Rule>
    <ID>test-rule2</ID>
    <Prefix></Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

- 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

  - 规则1

  - 规则2

示例2（无冲突）

禁用生命周期规则

本示例指定了两条生命周期规则。其中规则1指定前缀为 `logs/` 的Object，并在其创建100天后转换为IA存储类型。规则2指定前缀为 `documents/` 的Object，并在其创建50天后转换为Archive存储类型。然后在策略中选择禁用规则1，启动规则2。

应用以上策略后，OSS仅对<Status>处于Enabled（启动）状态的规则生效。即所有前缀为 `documents/` 的Object在其创建50天后转换为Archive存储类型。

以上生命周期配置规则的XML以及控制台配置示例如下：

- XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule1</ID>
    <Prefix>logs/</Prefix>
    <Status>Disabled</Status>
    <Transition>
      <Days>100</Days>
      <StorageClass>IA</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>test-rule2</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>50</Days>
      <StorageClass>Archive</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

- 规则1

- 规则2

结合版本控制的生命周期规则

假设您有一个启用了版本控制的Bucket，即该Bucket内的每个Object都有一个当前版本以及零个或零个以上的非当前版本。有关版本控制的更多信息请参见版本控制介绍。

- 您可以通过配置以下规则，实现当前版本Object距其最后修改时间超过10天后转换为IA存储类型，Object成为非当前版本60天后转换为Archive存储类型，Object成为非当前版本90天后删除。

  以上生命周期配置规则的XML以及控制台配置示例如下：

  - XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule0</ID>
    <Prefix></Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>10</Days>
      <StorageClass>IA</StorageClass>
    </Transition>
    <NoncurrentVersionTransition>
      <NoncurrentDays>60</NoncurrentDays>
      <StorageClass>Archive</StorageClass>
    </NoncurrentVersionTransition>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>90</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

- 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

示例1

- 当Object在仅剩一个删除标记版本，其余版本均已删除的情况下执行Expiration过期操作，则该删除标记即为过期删除标记。移除过期删除标记示例如下：

  以上生命周期配置规则的XML以及控制台配置示例如下：

  - XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>test-rule0</ID>
    <Prefix></Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

  - 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

示例2

清理过期碎片

通过生命周期规则指定在分片上传过程中，前缀为logs的碎片（即未执行CompleteMultipartUpload的Object）5天后过期。

以上生命周期配置规则的XML以及控制台配置示例如下：

- XML

```
<LifecycleConfiguration>
  <Rule>
    <ID>lifecyclerule1</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <AbortMultipartUpload>
      <Days>5</Days>
    </AbortMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

- 控制台

> **说明** 您还可以通过控制台配置符合以上条件的生命周期规则，配置详情如下图所示。具体操作，请参见设置生命周期规则。

5.8. 存储空间清单

您可以使用对象存储OSS的清单功能获取存储空间（Bucket）中指定文件（Object）的数量、大小、存储类型、加密状态等信息。相对于Get Bucket(List Objects)接口，在海量Object的列举场景中，建议您优先使用清单功能。

清单文件

清单任务配置完成后，OSS会按清单规则指定的导出周期生成清单文件。清单文件的目录结构如下：

```
dest_bucket
 └─destination-prefix/
     └─src_bucket/
         └─inventory_id/
             ├─YYYY-MM-DDTHH-MMZ/
             │   ├─manifest.json
             │   └─manifest.checksum
             └─data/
                 └─745a29e3-bfaa-490d-9109-47086afcc8f2.csv.gz
```

目录结构	说明
destination-prefix/	该目录根据设置的清单报告名前缀生成，如果清单报告名前缀设置为空，将省略该目录。
src_bucket/	该目录根据配置清单报告的源Bucket名生成。
inventory_id/	该目录根据清单任务的规则名称生成。
YYY-MM-DDTHH-MMZ/	该目录是标准的格林威治时间戳，表示开始扫描Bucket的时间，例如2020-05-17T16-00Z。该目录下包含了*manifest.json*和*manifest.checksum*文件。
data/	该目录下存放了包含源Bucket中的对象列表以及每个对象的元数据的清单文件，清单文件格式为使用GZIP压缩的CSV文件。 **注意** 当导出的清单数据较大时，会切分成多个CSV压缩文件。CSV压缩文件按照*uuid.csv.gz*、*uuid-1.csv.gz*、*uuid-2.csv.gz*的顺序依次递增。您可以从*manifest.json*文件中获取CSV文件列表，然后按照以上顺序依次解压CSV文件并读取清单数据。

清单功能生成的具体文件如下：

- manifest文件

  manifest文件包含*manifest.json*和*manifest.checksum*，详细说明如下：

- *manifest.json*：提供了有关清单的元数据和其他基本信息。

```
{
    "creationTimestamp": "1642994594",
    "destinationBucket": "destbucket",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker, Size, StorageClass, LastModifiedDate, ETag, IsMultipartUploaded, EncryptionS
tatus",
    "files": [{
            "MD5checksum": "F77449179760C3B13F1E76110F07****",
            "key": "destbucket/inventory0124/data/a1574226-b5e5-40ee-91df-356845777c04.csv.gz",
            "size": 2046}],
    "sourceBucket": "srcbucket",
    "version": "2019-09-01"}
```

各字段详细说明如下：

字段名称	说明
creationTimestamp	以纪元日期格式创建的时间戳，显示开始扫描源Bucket的时间。
destinationBucket	存放清单文件的目标Bucket。
fileFormat	清单文件的格式。
fileSchema	清单文件包含的字段。
files	包含清单文件的MD5值、文件名完整路径及文件大小。
sourceBucket	配置清单规则的源Bucket。
version	清单版本号。

- *manifest.checksum*：包含*manifest.json*文件的MD5值，例如 `8420A430CBD6B659A1C0DFC1C11A****` 。

- 清单报告

清单报告存储在*data/*目录中，包含清单功能导出的文件信息。清单样例如下：

[illegible]	1.png	CAEQGhiB	TRUE	FALSE	26010	Standard	2022-01-(	7925D2AA	FALSE	FALSE
[illegible]	1.png	CAEQGhiB	FALSE	FALSE	26010	Standard	2022-01-(	7925D2AA	FALSE	FALSE
[illegible]	Migration.png		TRUE	FALSE	13786	Standard	2021-09-2	E153ECD8	FALSE	FALSE
[illegible]	data_20210928_1003		TRUE	FALSE	3198090	Standard	2021-09-2	BA30D607	FALSE	FALSE
[illegible]	example%2F112%2F		TRUE	FALSE	0	Standard	2021-10-2	D41D8CD9	FALSE	FALSE
[illegible]	example%2Fram_ok.pı		TRUE	FALSE	49144	Standard	2021-10-2	08A9D641	FALSE	FALSE
[illegible]	ram.png		TRUE	FALSE	52437	Standard	2021-09-2	230423B7	FALSE	FALSE
[illegible]	test%2Fpeiyu-demo%2		TRUE	FALSE	32	Standard	2021-10-2	60ACD52D	FALSE	FALSE
[illegible]	test%2Fpeiyu-demo%2		TRUE	FALSE	487	Standard	2021-10-2	C64C1941	FALSE	FALSE
[illegible]	test%2Fpeiyu-demo%2		TRUE	FALSE	32	Standard	2021-10-2	E28592FAE	FALSE	FALSE
[illegible]	test%2Fpeiyu-demo%2		TRUE	FALSE	487	Standard	2021-10-2	C9544105	FALSE	FALSE
[illegible]	test%2Fpeiyu-demo%2		TRUE	FALSE	32	Standard	2021-10-2	9A9F23C1	FALSE	FALSE

清单报告的所有字段从左到右分别为：

字段名称	说明
Bucket	执行清单任务的源Bucket名称。
Key	Bucket中Object的名称。 Object名称使用URL编码，您必须解码后查看。
VersionId	Object的版本ID。 仅当Bucket已开启版本控制功能，且您配置的清单规则为导出所有版本时出现此字段。
IsLatest	Object版本是否为最新版本。当版本为最新版本时取值为*True*，否则取值为*False*。 仅当Bucket已开启版本控制功能，且您配置的清单规则为导出所有版本时出现此字段。
IsDeleteMarker	Object版本是否为删除标记。当版本为删除标记时取值为*True*，否则取值为*False*。 仅当Bucket已开启版本控制功能，且您配置的清单规则为导出所有版本时出现此字段。
Size	Object大小。
StorageClass	Object的存储类型。
LastModifiedDate	Object的最后修改时间。

字段名称	说明
ETag	Object的ETag。 Object生成时会创建相应的ETag，用于标识一个Object的内容。 ○ 通过PutObject接口创建的Object，ETag值是其内容的MD5值。 ○ 通过其他方式创建的Object，ETag值是基于一定计算规则生成的唯一值，但不是其内容的MD5值。
IsMultipartUploaded	Object是否通过分片上传生成。如果是，则该字段值为*True*，否则为*False*。
EncryptionStatus	Object是否已加密。若Object已加密，则该字段值为*True*，否则为*False*。

注意事项

- 权限说明

  RAM用户使用Bucket清单功能所需的权限分以下两种情况：

  - RAM用户无任何权限

    如果RAM用户在没有任何权限的情况下要使用Bucket清单功能，请按如下步骤完成授权：

    a. 通过**脚本配置**方式创建以下自定义策略。具体操作，请参见创建自定义权限策略。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "oss:PutBucketInventory",
                "oss:GetBucketInventory",
                "oss:DeleteBucketInventory",
                "oss:ListBuckets",
                "ram:CreateRole",
                "ram:AttachPolicyToRole",
                "ram:GetRole",
                "ram:ListPoliciesForRole"
            ],
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

    b. 为RAM用户授予已创建的自定义策略。具体操作，请参见为RAM用户授权。

  - RAM用户已拥有 `AliyunOSSFullAccess` 系统权限

    - 授予自定义策略

      在RAM用户已拥有 `AliyunOSSFullAccess` 权限的情况，您可以通过**脚本配置**方式创建如下自定义策略，并为RAM用户授予已创建的自定义策略。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ram:CreateRole",
                "ram:AttachPolicyToRole",
                "ram:GetRole",
                "ram:ListPoliciesForRole"
            ],
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

    - 授予 `AliyunRAMFullAccess` 系统权限

      在RAM用户已拥有 `AliyunOSSFullAccess` 权限的情况下，如果不希望通过授予自定义策略的方式来使用Bucket清单功能，您可以为该RAM用户添加系统权限 `AliyunRAMFullAccess` 。添加系统权限的具体操作，请参见为RAM用户授权。

  完成以上授权操作后，请通过控制台配置Bucket清单规则。规则配置完成后，RAM控制台将自动创建名为 `AliyunOSSRole` 的RAM角色，该角色默认拥有读取源Bucket所有文件和向目标Bucket写入文件的权限。后续如需通过SDK等方式使用Bucket清单功能，建议直接使用控制台创建的RAM角色 `AliyunOSSRole` ，避免额外的角色授权操作。

- 配置建议

  建议您根据源Bucket内的文件数量配置清单任务：

  - 文件数量小于10亿，可以以天为单位生成清单文件。
  - 文件数量为小于100亿，可以以周为单位生成清单文件。
  - 文件数量大于100亿，建议以周为单位，并针对不同的文件前缀设置不同的清单任务，保证每个清单任务涉及的文件不超过100亿个。

- 流量带宽说明

  为保障较快的清单文件列表导出速度，在清单文件列表导出到目标Bucket的过程中，可能会占用一定的Bucket与用户级别带宽。如果存放清单文件列表的目标Bucket与配置清单任务的源Bucket相同，且源Bucket存在流量较大带宽紧张的情况，建议新建一个目标Bucket用于存放清单结果文件。

- 异常说明

  - 若源Bucket没有任何文件，或清单任务设置的前缀没有匹配到任何文件，则不会生成清单文件。

- 导出清单文件的过程中，由于Object的创建、删除或覆盖等操作，可能会导致最终输出的清单列表中不一定包含所有的Object。最后修改时间早于*manifest.json*文件中createTimeStamp字段显示时间的Object会出现在清单文件中；最后修改时间晚于createTimeStamp字段显示时间的Object可能不会出现在清单文件中。建议您对清单列表中的Object进行操作之前，先使用HeadObject接口检查Object的属性。更多信息，请参见HeadObject。

使用限制

- 对于单个Bucket，通过SDK或者命令行工具ossutil最多可配置1000条清单规则，通过OSS管理控制台最多可配置10条清单规则。
- 配置清单的源Bucket与存放清单文件的目标Bucket可以相同也可以不同，但是必须属于同一账号下的相同地域。

计费说明

- 使用Bucket清单功能会产生一定的费用，公测期间仅收取API请求费用和清单文件存储费用，暂不收取功能使用费用。
- 在您删除清单规则前，OSS会按照清单规则一直生成清单文件，会产生一定的存储费用。为避免产生不必要的费用，请及时清理不再需要的清单文件。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 选择**基础设置 > Bucket清单**，之后单击**设置**。
4. 单击**创建清单**，在**设置清单报告规则**对话框设置如下参数：

参数	说明
状态	设置清单任务的状态。可以选择**启动**和**禁用**。
规则名称	设置清单任务的名称。只能包含小写字母、数字、短划线（-），且不能以短划线（-）开头或结尾。
目标Bucket	选择存储清单文件的目标Bucket。 配置清单的源Bucket与存放清单文件的目标Bucket可以相同也可以不同，但是必须属于同一账号下的相同地域。
清单报告目录	设置清单报告存储的目录。 ○ 如果要将清单报告保存在Bucket根目录下，请将此项置空。 ○ 如果要将清单报告保存在Bucket非根目录下，请填写不包含Bucket名称在内的目录完整路径。 例如，当您需要将清单报告保存在目标存储空间examplebucket的exampledir1目录下，则填写为*exampledir1*。当您需要将清单报告保存在exampledir1目录下的子目录exampledir2，则填写为*exampledir1/exampledir2*。
清单报告导出周期	设置清单报告的生成周期。可选择**每周**或**每天**。 建议您根据源Bucket内的文件数量配置清单任务： ○ 文件数量小于10亿，建议以天为单位生成清单文件。 ○ 文件数量为小于100亿，建议以周为单位生成清单文件。 ○ 文件数量大于100亿，建议以周为单位，并针对不同的文件前缀设置不同的清单任务，保证每个清单任务涉及的文件不超过100亿个。
清单报告加密选项	是否加密清单文件。 ○ **无**：不加密。 ○ **AES256**：使用AES256加密算法加密清单文件。 ○ **KMS**：使用KMS密钥加密清单文件。 您可以选择使用OSS托管的KMS密钥或在KMS平台创建一个与目标Bucket相同地域的KMS密钥。KMS密钥配置步骤，请参见创建密钥。 **说明** 使用KMS密钥功能时会产生少量的KMS密钥API调用费用。
对象版本	选择清单扫描的文件版本。 如果Bucket已开启版本控制，可选择导出目标文件的**当前版本**或**所有版本**。更多信息，请参见版本控制介绍。 如果Bucket未开启版本控制，默认导出所有文件。
按前缀匹配	设置清单规则扫描Object的前缀。 ○ 如果要扫描整个Bucket内的所有Object，请将此项置空。 ○ 如果要扫描Bucket某个目录下的所有Object，请填写不包含Bucket名称在内的目录完整路径。 例如，当您需要扫描目标存储空间examplebucket根目录exampledir1下的所有Object，则填写为*exampledir1/*。如果您需要扫描根目录exampledir1下子目录exampledir2的所有Object，则填写为*exampledir1/exampledir2/*。 **说明** 如果设置的前缀没有匹配Bucket内的任意Object，则不生成清单文件。
清单内容可选信息	选择您希望导出的文件信息，包括**Object大小**、**存储类型**、**最后更新时间**、**ETag**、**分片上传状态**、**加密状态**。

5. 选中**我知晓并同意授予阿里云OSS服务访问Bucket资源的权限**后，单击**确定**。
   涉及Object较多时，生成清单文件需要一定的时间。如果您希望第一时间获知清单文件已生成的消息，建议您在生成清单文件的目标Bucket中配置事件通知规则，并将事件类型设置为PutObject。当清单文件生成之后，您会收到Object生成的提醒。配置事件通知方式，请参见设置事件通知规则。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成Bucket清单配置的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK配置Bucket清单的代码示例。关于其他SDK的配置Bucket清单代码示例，请参见SDK简介。

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
import java.util.ArrayList;
import java.util.List;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写存放清单结果的Bucket名称。
        String destBucketName ="yourDestinationBucketName";
        // 填写Bucket所有者授予的账户ID。
        String accountId ="yourDestinationBucketAccountId";
        // 填写具有读取源Bucket所有文件和向目标Bucket写入文件权限的角色名称。
        String roleArn ="yourDestinationBucketRoleArn";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 创建清单配置。
            InventoryConfiguration inventoryConfiguration = new InventoryConfiguration();
            // 设置清单规则名称。
            String inventoryId = "testid";
            inventoryConfiguration.setInventoryId(inventoryId);
            // 设置清单中包含的Object属性。
            List<String> fields = new ArrayList<String>();
            fields.add(InventoryOptionalFields.Size);
            fields.add(InventoryOptionalFields.LastModifiedDate);
            fields.add(InventoryOptionalFields.IsMultipartUploaded);
            fields.add(InventoryOptionalFields.StorageClass);
            fields.add(InventoryOptionalFields.ETag);
            fields.add(InventoryOptionalFields.EncryptionStatus);
            inventoryConfiguration.setOptionalFields(fields);
            // 设置清单的生成计划，以下示例为每周一次。其中，Weekly表示每周一次，Daily表示每天一次。
            inventoryConfiguration.setSchedule(new InventorySchedule().withFrequency(InventoryFrequency.Weekly));
            // 设置清单中包含的Object的版本为当前版本。如果设置为InventoryIncludedObjectVersions.All则表示Object的所有版本在版本控制状态下生效。
            inventoryConfiguration.setIncludedObjectVersions(InventoryIncludedObjectVersions.Current);
            // 清单配置是否启用的标识，取值为true或者false，设置为true表示启用清单配置，设置为false表示关闭清单配置。
            inventoryConfiguration.setEnabled(true);
            // 设置清单筛选规则，指定筛选Object的前缀。
            InventoryFilter inventoryFilter = new InventoryFilter().withPrefix("obj-prefix");
            inventoryConfiguration.setInventoryFilter(inventoryFilter);
            // 创建存放清单结果的目标Bucket配置。
            InventoryOSSBucketDestination ossInvDest = new InventoryOSSBucketDestination();
            // 设置存放清单结果的存储路径前缀。
            ossInvDest.setPrefix("destination-prefix");
            // 设置清单格式。
            ossInvDest.setFormat(InventoryFormat.CSV);
            // 设置目标Bucket的用户accountId。
            ossInvDest.setAccountId(accountId);
            // 设置目标Bucket的roleArn。
            ossInvDest.setRoleArn(roleArn);
            // 设置目标Bucket的名称。
            ossInvDest.setBucket(destBucketName);
            // 如果需要使用KMS加密清单，请参考如下设置。
            // InventoryEncryption inventoryEncryption = new InventoryEncryption();
            // InventoryServerSideEncryptionKMS serverSideKmsEncryption = new InventoryServerSideEncryptionKMS().withKeyId("test-kms-id");
            // inventoryEncryption.setServerSideKmsEncryption(serverSideKmsEncryption);
            // ossInvDest.setEncryption(inventoryEncryption);
            // 如果需要使用OSS服务端加密清单，请参考如下设置。
            // InventoryEncryption inventoryEncryption = new InventoryEncryption();
            // inventoryEncryption.setServerSideOssEncryption(new InventoryServerSideEncryptionOSS());
            // ossInvDest.setEncryption(inventoryEncryption);
            // 设置清单的目的地。
            InventoryDestination destination = new InventoryDestination();
            destination.setOssBucketDestination(ossInvDest);
            inventoryConfiguration.setDestination(destination);
            // 上传清单配置。
            ossClient.setBucketInventoryConfiguration(bucketName, inventoryConfiguration);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
```

```
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil配置Bucket清单的具体操作，请参见inventory（清单）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketInventory。

常见问题

如何快速得知清单文件已生成？

涉及Object较多时，清单文件生成需要一定的时间。如果您希望第一时间获知清单文件已生成，建议您在生成清单的目标Bucket中配置事件通知规则，将事件类型设置为Put Object。当清单生成之后，您会收到Object生成的提醒。关于配置事件通知的具体操作，请参见设置事件通知规则。

5.9. 请求者付费模式

对象存储OSS的请求者付费模式是指由请求者支付访问存储空间（Bucket）内数据时产生的费用，而Bucket拥有者仅支付存储费用。当您希望共享数据，但又不希望支付因共享数据产生的额外费用时，您可以开启此功能。

使用场景

- 共享大型数据集。例如某研究机构希望所有客户都能访问包含邮政编码目录、参考数据、地理空间信息或网络爬取等数据的共享数据集，同时希望下载数据产生的流量费用和请求次数费需由请求者支付。

  配置步骤如下：

  i. 确保存放共享数据集所属Bucket的读写权限ACL为公共读。具体步骤，请参见Bucket ACL。

  ii. 为该Bucket开启请求者付费模式。

- 将生产数据交付给您的客户或合作伙伴。例如，某公司需要将生产数据交付给他的合作伙伴，下载数据产生的流量费用和请求次数费用需要由合作伙伴支付。

  配置步骤如下：

  i. 确保存放生产数据所属Bucket的读写权限ACL设置为私有。具体步骤，请参见Bucket ACL。

  ii. 为该Bucket开启请求者付费模式。

  iii. 通过Bucket Policy授权您的合作伙伴访问该Bucket内指定的生产数据。具体步骤，请参见基于Bucket Policy实现跨账号访问OSS。

> **注意** 您需要将Bucket授权给合作伙伴的RAM用户，而不是将您账号下RAM用户的AccessKey提供给合作伙伴进行访问。原因是当合作伙伴通过您账号下的RAM用户访问时，请求者是您自身，请求费用仍需要您（请求者）来支付。

请求方式

- 不允许匿名访问

  如果您在Bucket上启用了请求者付费模式，则不允许匿名访问该Bucket。请求方必须提供身份验证信息，以便OSS能够识别请求方，从而对请求方而非Bucket拥有者收取请求所产生的费用。

  当请求者是通过扮演阿里云RAM角色来请求数据时，该角色所属的账户将为此请求付费。

- 请求中需携带请求头 `x-oss-request-payer`

  如果您在Bucket上启用了请求者付费模式，请求中必须携带 `x-oss-request-payer` 请求头，取值为*requester*，以表明请求方已了解需要支付请求和数据下载费用。否则，请求方无法通过验证。

  - 对于POST、GET和HEAD请求，需在请求中包含 `x-oss-request-payer:requester` 信息。
  - 对于签名URL，需在请求中包含 `x-oss-request-payer=requester` 信息。

  数据拥有者访问该Bucket时，可以不携带 `x-oss-request-payer` 请求头。数据拥有者作为请求者访问该Bucket时，请求产生的费用由数据拥有者（也是请求者）来支付。

费用说明

请求者付费模式下，请求者会根据请求的内容支付请求次数、外网流出流量、CDN回源流量、图片处理、视频截帧、低频或归档存储的数据取回等费用中的一项或多项，Bucket拥有者仅支付如存储费用、对象标签费用、传输加速费用等。如果出现以下情况，请求会失败，并返回HTTP 403错误，且由Bucket拥有者支付请求费用：

- 请求者未在请求中（GET、HEAD或POST）包含请求头 `x-oss-request-payer` 。
- 请求身份验证失败。
- 该请求是匿名请求 。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**基础设置 > 请求者付费**。在**请求者付费**区域单击**设置**，选择开启或关闭请求者付费模式。

4. 单击**保存**。

使用阿里云SDK

以下仅列举常见SDK的设置请求者付费模式的代码示例。关于其他SDK的设置请求者付费模式的代码示例，请参见SDK简介。

Nodejs

```
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
public class Demo {
    public static void main(String[] args) {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。关于其他Region对应的Endpoint信息，请参见访问域名和数据中心。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 设置请求者付费模式。
            Payer payer = Payer.Requester;
            ossClient.setBucketRequestPayment(bucketName, payer);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (Throwable ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            // 关闭OSSClient。
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置请求者付费模式的具体操作， 请参见设置请求者付费模式。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketRequestPayment。

5.10. 绑定自定义域名

文件（Object）上传至存储空间（Bucket）后，OSS会自动生成文件URL，您可以直接通过文件URL访问该文件。如果您希望通过自定义域名访问这些文件，需要将自定义域名绑定至文件所在的Bucket，并添加CNAME记录。

前提条件

- 已注册自定义域名。具体步骤，请参见注册域名。
- 绑定至中国内地Bucket上的域名已在中国工信部备案。

  您可以通过阿里云提供的备案服务进行域名备案。

使用限制

- 每个存储空间最多可以绑定100个域名。一个域名只能绑定在一个存储空间上，每个账号可绑定的域名个数无限制。
- 通过OSS管理控制台绑定自定义域名时，不允许绑定泛域名。通过CDN服务加速OSS时，允许绑定泛域名，但该域名不会在OSS管理控制台显示。

注意事项

您可以通过绑定自定义域名满足以下使用场景。

- 对于2019年09月23日以后在中国内地创建的Bucket，需确保通过浏览器访问Bucket内的图片文件时是预览而非下载行为。
- 将Bucket配置成静态网站时，需确保正确访问静态网站页面，而非下载静态网页。

自定义域名使用规则

例如华东1（杭州）有名为*examplebucket*的Bucket，其根目录下有名为*exampleobject.jpg*的公共读Object，绑定的自定义域名为 `www.example.com` 。则绑定自定义域名前后的访问方式如下：

- 绑定前

  使用Bucket默认域名访问*exampleobject.jpg*，访问地址为 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/exampleobject.jpg` 。

- 绑定后

  使用自定义域名访问*exampleobject.jpg*，访问地址为 `https://www.example.com/exampleobject.jpg` 。

> **说明** 绑定自定义域名后，通过文件URL访问图片时，如果图片仍然无法预览而是以附件的形式下载，则原因可能是浏览器对于部分图片格式不支持预览。您只需要在浏览器上安装支持预览对应格式文件的插件即可。

使用OSS控制台

1. 绑定自定义域名。
   i. 登录OSS管理控制台。
   ii. 单击**Bucket列表**，然后单击目标Bucket名称。
   iii. 单击**传输管理 > 域名管理**。
   iv. 单击**绑定域名**。
   v. 在**绑定域名**面板，输入要绑定的**域名**。

      绑定的域名不支持泛域名，例如 `*.example.com` 。

      如果提示域名冲突，表示该域名已绑定至其他Bucket。此时，您可以更换域名或通过验证域名所有权强制绑定域名。验证域名所有权会解除域名与其他Bucket的绑定关系。
2. 添加CNAME记录。
   - 如果添加的域名为当前账号下管理的域名，开启自动添加CNAME记录。
     a. 在**绑定域名**面板，打开**自动添加CNAME记录**开关。

     > **注意** 如果您绑定的域名已配置过CNAME，则自动添加的CNAME记录会覆盖原有的CNAME记录。

     b. 单击**提交**。
   - 如果添加的域名为非当前账号下的域名，手动添加CNAME记录。

     如果您的域名为非阿里云托管的域名，需在对应的域名解析商处配置云解析。

     此处以非当前账号下阿里云托管的域名为例，手动添加CNAME记录步骤如下：
     a. 登录云解析DNS控制台。
     b. 在域名解析列表中，单击目标域名右侧的**解析设置**。
     c. 单击**添加记录**，填写域名解析信息。

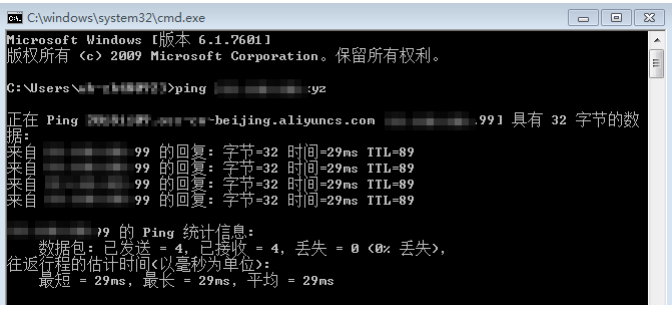
参数	说明
记录类型	选择域名指向的类型。 此处选择**CNAME**。
主机记录	根据域名前缀填写主机记录。 ■ 如果是顶级域名，例如 `aliyun.com` ，输入**@**。 ■ 如果是二级域名，输入二级域名的前缀。例如域名为 `help.aliyun.com` ，输入**help**。 ■ 如果需要所有的二级域名都指向Bucket外网访问域名，输入*****。
解析线路	解析域名时使用的线路。 建议选择**默认**，系统将自动选择最佳线路。
记录值	填写Bucket外网访问域名。Bucket外网访问域名结构为 `BucketName.Endpoint` ，例如华东1（杭州）地域创建了名为examplebucket的存储空间，外网Endpoint为 `oss-cn-hangzhou.aliyuncs.com` ，则填写为 `examplebucket.oss-cn-hangzhou.aliyuncs.com` 。
TTL	域名的更新周期，保留默认值即可。

     d. 单击**确定**。

     新增CNAME记录实时生效，修改CNAME记录最多72小时内生效。

验证CNAME配置是否生效

您可以通过**ping**或**lookup**命令测试您添加的域名，如果被转向 `*.oss-cn-*.aliyuncs.com` ，即表示CNAME配置已经生效。

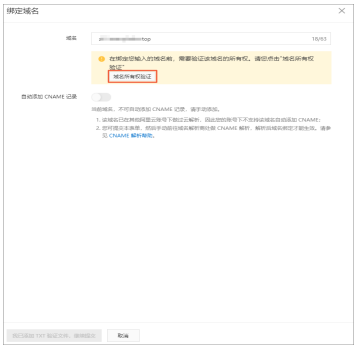


验证域名所有权

绑定自定义域名提示域名冲突时，您可以通过验证域名所有权强制绑定此域名。

1. 单击**域名所有权验证**。

   OSS会随机生成该域名的token，包含**域名**、**主机记录**和**值**，您需要保存这些信息。



2. 在您的域名服务商处添加TXT记录，填写步骤1中保存的**主机记录**和**值**，其他参数保持默认配置。

   具体步骤，请参见手动添加CNAME记录。

3. 在**绑定用户域名**面板，单击**我已添加TXT验证文件，继续提交**。

   如果配置无误，OSS会将该域名绑定在当前Bucket。

解除域名绑定

当您的自定义域名不再使用时，可以手动解除域名绑定。

1. 在目标Bucket管理页面，单击**传输管理 > 域名管理**。
2. 在域名列表中，单击目标域名右侧的**域名绑定配置**。
3. 在**域名绑定配置**面板，单击**解除绑定**，然后单击**确定**。

更多参考

- 如果您希望获得更好的上传、下载体验，需绑定传输加速域名。具体步骤，请参见绑定传输加速域名。
- 如果您希望通过静态网页访问OSS资源，需设置静态网站托管。具体步骤，请参见设置静态网站托管和使用自定义域名设置静态网站托管。
- 如果您希望使用HTTPS协议访问自定义域名，需在OSS控制台上传您的HTTPS证书。具体步骤，请参见证书托管。

5.11. 传输加速

OSS传输加速利用全球分布的云机房，将全球各地用户对您存储空间（Bucket）的访问，经过智能路由解析至就近的接入点，使用优化后的网络及协议，为云存储互联网的上传、下载提供端到端的加速方案。

前提条件

已在实名登记页面完成实名认证。

使用场景

- 远距离数据传输加速

  例如全球性的论坛、Top在线协同办公平台等，部分客户会因传输距离较远导致上传和下载体验非常差。传输加速功能可以让全球各地的客户使用优化后的网络来传输数据，极大地提升上传和下载速度，让不同地域的用户都能有很好的访问体验。

- GB、TB级大文件上传和下载

  通过互联网远距离上传和下载大文件时，经常会因为网络延迟过大而导致传输失败。传输加速功能使用优化的互联网传输链路、调优的协议栈与传输算法，可大幅减少远距离互联网传输超时的比例。您还可以让传输加速功能与分片上传、断点续传下载结合，形成远距离大文件上传和下载的解决方案。

- 非静态、非热点数据下载加速

  例如相册应用、游戏、电商、社交应用的评论内容、企业门户网站、金融类APP等，用户的下载体验直接影响产品竞争力和客户留存率。传输加速功能作为专为OSS上传、下载加速而设计的功能，可以最大限度利用客户端的网络能力，提升用户的下载体验。

注意事项

- 传输加速开启及关闭操作会在30分钟内全网生效。
- 开启传输加速后必须使用OSS的传输加速域名才会提升访问速度。
- 如果您工具中配置的Endpoint为传输加速Endpoint时，您只能操作已开启传输加速功能的Bucket。
- 开启传输加速功能后，OSS提供的其他Endpoint仍可正常使用。在不需要传输加速的场景中，您可以使用默认Endpoint以减少传输加速的费用。
- 传输加速Endpoint仅支持HTTP/HTTPS协议的API接入，不支持RTMP协议等非HTTP/HTTPS协议的API接入。
- 为保证数据传输安全，传输加速后段加速逻辑会视情况选择使用HTTPS协议进行数据传输。所以，客户端使用HTTP协议通过传输加速域名访问OSS时，在OSS的访问日志中看OSS仅支持到的访问协议可能是HTTPS。
- 如果您开启了传输加速功能，且使用传输加速域名访问您的Bucket时，OSS会收取传输加速费用。关于费用说明，请参见传输加速费用。

开启传输加速

通过以下多种方式开启传输加速后，Bucket会在保留默认Endpoint的基础上，新增以下两种传输加速Endpoint。

- 全球加速Endpoint：地址为*oss-accelerate.aliyuncs.com*。传输加速接入点分布在全球各地，全球各地的Bucket均可以使用该域名进行传输加速。
- 非中国内地加速Endpoint：地址为*oss-accelerate-overseas.aliyuncs.com*。传输加速接入点分布在除中国内地以外的各地域，仅在中国香港及海外各地域Bucket绑定未备案的域名做CNAME指向时使用。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**传输管理 > 传输加速**。

4. 单击**设置**并开启传输加速，然后单击**保存**。

使用阿里云SDK

以下仅列举常见SDK开启传输加速的代码示例。关于其他SDK的开启传输加速的代码示例，请参见SDK简介。

Python Golang

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 设置Bucket的传输加速状态。
            // 当设置enabled为true时，表示开启传输加速；当设置enabled为false时，表示关闭传输加速。
            boolean enabled = true;
            ossClient.setBucketTransferAcceleration(bucketName, enabled);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketTransferAcceleration。

使用传输加速

使用浏览器

通过浏览器访问OSS时，将文件URL的Endpoint字段需替换为传输加速Endpoint，例如 `https://test.oss-cn-shenzhen.aliyuncs.com/myphoto.jpg` 需改为 `https://test.oss-accelerate.aliyuncs.com/myphoto.jpg` 。如果文件访问权限为私有，则还需要加上签名信息。

> **说明** 如果您的存储空间已绑定自定义域名，且希望通过自定义域名访问OSS时实现传输加速，您可以通过CNAME配置，将您的域名指向OSS加速域名。配置方式，请参见绑定传输加速域名。

使用命令行工具ossutil

- 替换配置文件的Endpoint为传输加速Endpoint

  通过ossutil访问时，您可以将配置文件内的Endpoint替换为传输加速Endpoint。具体步骤，请参见ossutil。

- 在命令示例中增加 `-e oss-accelerate.aliyuncs.com`

  在使用ossutil相关命令时，在命令示例中增加 `-e oss-accelerate.aliyuncs.com` 。下图表示在cp命令上传场景中使用了传输加速Endpoint：

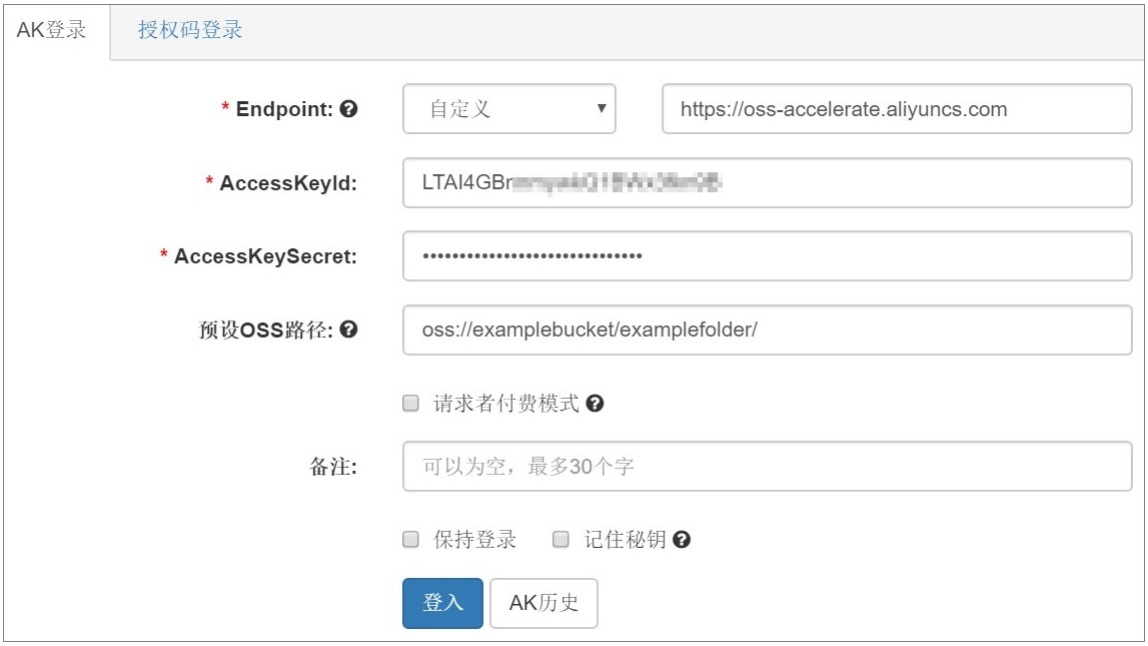
使用图形化管理工具ossbrowser

> **注意** 通过ossbrowser访问OSS时，除了要填写AccessKey（AK）信息以外，还必须指定预设OSS路径。

各配置项说明如下：

参数	说明
Endpoint	选择**自定义**，并填写传输加速Endpoint：*https://oss-accelerate.aliyuncs.com*。
AccessKeyId、**AccessKeySecret**	填写账号的AccessKey（AK）信息。获取AK的方式，请参见获取AccessKey。 **注意** 为保证数据安全，推荐您使用RAM用户的AK登录ossbrowser。使用RAM用户登录之前，需要为RAM用户配置 `AliyunOSSFullAccess`、`AliyunRAMFullAccess` 以及 `AliyunSTSAssumeRoleAccess` 的权限。具体操作请参见权限管理。
预设OSS路径	指定访问某个Bucket或Bucket某个路径下资源的访问权限。预设OSS路径格式为*oss://bucketname/path*。例如授权访问存储空间examplebucket下文件夹examplefolder下的文件或子文件夹，则填写*oss://examplebucket/examplefolder/*。

配置示例如下：



使用阿里云SDK

通过各语言SDK访问OSS时，将Endpoint设置为传输加速Endpoint。以Java SDK的简单上传和简单下载为例：

- 简单上传

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.PutObjectRequest;
import java.io.File;
public class Demo {
    public static void main(String[] args) throws Exception {
        // 填写传输加速Endpoint。以全球加速Endpoint为例。
        String endpoint = "https://oss-accelerate.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，例如exampledir/exampleobject.txt。Object完整路径中不能包含Bucket名称。
        String objectName = "exampledir/exampleobject.txt";
        // 填写本地文件的完整路径，例如D:\\localpath\\examplefile.txt。
        // 如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
        String filePath= "D:\\localpath\\examplefile.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 创建PutObjectRequest对象。
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, objectName, filePath);
            // 如果需要上传时设置存储类型和访问权限，请参考以下示例代码。
            // ObjectMetadata metadata = new ObjectMetadata();
            // metadata.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard.toString());
            // metadata.setObjectAcl(CannedAccessControlList.Private);
            // putObjectRequest.setMetadata(metadata);
            // 上传文件。
            ossClient.putObject(putObjectRequest);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

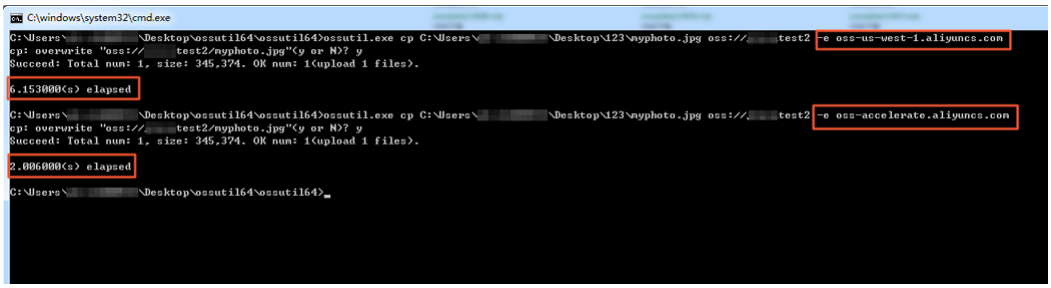
- 简单下载

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.GetObjectRequest;
import java.io.File;
public class Demo {
    public static void main(String[] args) throws Exception {
        // 填写传输加速Endpoint。以全球加速Endpoint为例。
        String endpoint = "https://oss-accelerate.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写不包含Bucket名称在内的Object完整路径，例如testfolder/exampleobject.txt。
        String objectName = "testfolder/exampleobject.txt";
        String filePath = "D:\\localpath\\examplefile.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 下载Object到本地文件，并保存到指定的本地路径中。如果指定的本地文件存在会覆盖，不存在则新建。
            // 如果未指定本地路径，则下载后的文件默认保存到示例程序所属项目对应本地路径中。
            ossClient.getObject(new GetObjectRequest(bucketName, objectName), new File(filePath));
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

测试传输加速效果

使用命令行工具ossutil

通过在ossutil命令示例中测试未开启传输加速前（即使用 `-e oss-us-west-1.aliyuncs.com` ）以及开启传输加速后（即使用 `-e oss-accelerate.aliyuncs.com` ）上传文件所需时间的差异。



使用在线工具

您可以通过OSS全球传输加速效果对比工具测试您本地访问全球各地数据中心时，开启传输加速与未开启传输加速的访问速度。

常见问题

- 传输加速与CDN加速有什么区别？

功能	原理	适用场景
传输加速	通过智能调度的系统、优化的传输链路、调优的协议栈与传输算法，并深度结合OSS服务端的配套策略，提供的端到端的加速方案。	○ 文件上传加速场景。 ○ 远距离的文件上传、下载场景。 ○ 大文件上传、下载场景。 ○ 动态更新文件、非热点文件的下载加速场景。
CDN加速	将OSS的Bucket作为源站，将源内容缓存到边缘节点。当客户读取数据时，会从最适合的节点获取缓存文件，以提升下载速度。	静态热点文件的下载加速场景，即同一地区大量用户同时下载同一个静态文件的场景。

- 如何通过自定义域名使用传输加速服务？

  您可以绑定自定义域名之后，将CNAME指向传输加速域名。具体操作，请参见绑定传输加速域名。

- 为什么使用传输加速Endpoint无法列举Bucket？

  传输加速服务仅针对携带Bucket名称信息的三级域名（格式为 `https://BucketName.oss-accelerate.aliyuncs.com` ）提供解析服务。而列举Bucket的请求域名中不携带Bucket名称信息，因此您无法使用传输加速Endpoint列举Bucket。建议您通过默认Endpoint列举目标地域的Bucket，例如 `https://oss-cn-hangzhou.aliyuncs.com` 。

5.12. 设置跨域资源共享

跨域资源共享CORS（Cross-Origin Resource Sharing）简称跨域访问，是HTML5提供的标准跨域解决方案，允许Web应用服务器进行跨域访问控制，确保跨域数据传输的安全性。

同源检测

跨域访问是浏览器出于安全考虑而设置的一个限制，即同源策略，是用于隔离潜在恶意文件的关键安全机制。当A、B两个网站属于不同域时，来自于A网站页面中的JavaScript代码访问B网站时，浏览器会拒绝该访问。

同协议、同域名（或IP）、以及同端口视为同域。两个页面的协议、域名和端口（若指定了端口）相同，则视为同源。下表给出了相对*http://www.aliyun.com/org/test.htm*的同源检测示例：

URL	访问是否成功	原因
http://www.aliyun.com/org/other.html	是	协议、域名、端口相同
http://www.aliyun.com/org/internal/page.html	是	协议、域名、端口相同
https://www.aliyun.com/page.html	否	协议不同（HTTPS）
http://www.aliyun.com:22/dir/page.html	否	端口不同（22）
http://www.alibabacloud.com/help/other.html	否	域名不同

从上表中可以看出，协议、域名或者端口不同的情况下，浏览器会拒绝该来源的访问。如果要允许这些来源的访问，需要设置跨域资源共享规则。

注意事项

- 每个Bucket最多可以配置10条跨域规则。
- 当OSS收到一个跨域请求（或者OPTIONS请求）时，会读取Bucket对应的CORS规则，然后进行相应的权限检查。OSS会依次检查每一条规则，使用第一条匹配的规则来允许请求并返回对应的Header。如果所有规则都匹配失败，则不附加任何CORS相关的Header。
- 如果您开启了CDN加速，并且需要进行跨域访问时，您需要在CDN控制台配置跨域规则。具体步骤，请参见CDN如何配置跨域资源共享（CORS）。

CORS规则

OSS支持根据需求灵活配置CORS规则，实现允许或者拒绝相应的跨域请求。CORS规则仅用来决定是否附加CORS相关的Header，是否拦截跨域请求由浏览器决定。

以下两种情况下需选中 `返回Vary: Origin` 头以避免本地缓存错乱。

> **注意** 选中 `返回Vary: Origin` 头后，可能会造成浏览器访问次数或者CDN回源次数增加。

- 同时存在CORS和非CORS请求

  例如实际请求中在<img>标签下发起非CORS请求，在fetch下发起CORS请求。

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>CORS Test</title>
</head>
<body>
//非CORS请求。
<img src="https://examplebucket.oss-cn-beijing.aliyuncs.com/exampleobject.txt" alt="">
<script>
  //CORS请求。
  fetch("https://examplebucket.oss-cn-beijing.aliyuncs.com/exampleobject.txt").then(console.log)
</script>
</body>
</html>
```

- Origin头存在多种可能值

  例如实际应用中指定允许的跨域请求来源Origin头为 `http://www.example.com` 以及 `https://www.example.org` 。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**权限管理 > 跨域设置**，然后在**跨域设置**区域，单击**设置**。
4. 单击**创建规则**，在**创建跨域规则**面板设置跨域访问参数。

参数	是否必须	说明

参数	是否必须	说明
来源	是	指定允许的跨域请求的来源。配置规则如下： ○ 允许多条匹配规则，多条规则需换行填写。 ○ 域名需包含协议名，例如HTTP、HTTPS。 ○ 支持通配符星号（*），每条匹配规则允许使用最多一个星号（*）。 ○ 若域名使用的不是默认端口，还需要携带端口号。例如：https://www.example.com:8080。 域名配置示例如下： ○ 匹配指定域名时，填写完整域名，例如：https://www.example.com。 ○ 匹配泛二级域名，可使用通配符星号（*）。例如：https://*.example.com。 ○ 匹配所有域名，可直接填写通配符星号（*）。
允许Methods	是	指定允许的跨域请求方法。
允许Headers	否	指定允许跨域请求的响应头。配置规则如下： ○ 格式为key:value，例如content-type:text/plain，大小写不敏感。 ○ 允许多条匹配规则，多条规则需换行填写。 ○ 每条匹配规则最多使用一个星号（*）通配符。建议没有特殊需求的情况下设置为星号（*）。
暴露Headers	否	指定允许用户从应用程序中访问的响应头。例如一个Javascript的XMLHttpRequest对象。不允许使用星号（*）通配符。
缓存时间	否	指定浏览器对特定资源的预取（OPTIONS）请求返回结果的缓存时间，单位为秒。
返回Vary：Origin	否	配置是否返回Vary：Origin Header。 如果实际应用中同时存在CORS和非CORS请求，或者Origin头有多种可能值时，建议选中**返回Vary：Origin**以避免本地缓存错乱。 **注意** 选中**返回Vary：Origin**后，可能会造成浏览器访问次数或者CDN回源次数增加。

有关以上配置的各项跨域访问参数的更多信息，请参见PutBucketCors。

5. 单击**确定**。

使用阿里云SDK

以下仅列举常见SDK的设置跨域资源共享的代码示例。关于其他SDK的设置跨域资源共享的代码示例，请参见SDK简介。

Nodejs

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.SetBucketCORSRequest;
import java.util.ArrayList;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            SetBucketCORSRequest request = new SetBucketCORSRequest(bucketName);
            // 跨域资源共享规则的容器，每个存储空间最多允许10条规则。
            ArrayList<SetBucketCORSRequest.CORSRule> putCorsRules = new ArrayList<SetBucketCORSRequest.CORSRule>();
            SetBucketCORSRequest.CORSRule corRule = new SetBucketCORSRequest.CORSRule();
            ArrayList<String> allowedOrigin = new ArrayList<String>();
            // 指定允许跨域请求的来源。
            allowedOrigin.add( "http://example.com");
            ArrayList<String> allowedMethod = new ArrayList<String>();
            // 指定允许的跨域请求方法(GET/PUT/DELETE/POST/HEAD)。
            allowedMethod.add("GET");
            ArrayList<String> allowedHeader = new ArrayList<String>();
            // 是否允许预取指令（OPTIONS）中Access-Control-Request-Headers头中指定的Header。
            allowedHeader.add("x-oss-test");
            ArrayList<String> exposedHeader = new ArrayList<String>();
            // 指定允许用户从应用程序中访问的响应头。
            exposedHeader.add("x-oss-test1");
            // AllowedOrigins和AllowedMethods最多支持一个星号（*）通配符。星号（*）表示允许所有的域来源或者操作。
            corRule.setAllowedMethods(allowedMethod);
            corRule.setAllowedOrigins(allowedOrigin);
            // AllowedHeaders和ExposeHeaders不支持通配符。
            corRule.setAllowedHeaders(allowedHeader);
            corRule.setExposeHeaders(exposedHeader);
            // 指定浏览器对特定资源的预取（OPTIONS）请求返回结果的缓存时间，单位为秒。
            corRule.setMaxAgeSeconds(10);
            // 最多允许10条规则。
            putCorsRules.add(corRule);
            // 已存在的规则将被覆盖。
            request.setCorsRules(putCorsRules);
            // 指定是否返回Vary: Origin头。指定为TRUE，表示不管发送的是否为跨域请求或跨域请求是否成功，均会返回Vary: Origin头。指定为False，表示任何情况下都不会返回
Vary: Origin头。
            // request.setResponseVary(Boolean.TRUE);
            ossClient.setBucketCORS(request);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置跨域资源共享的具体操作， 请参见添加或修改CORS配置。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketCors。

常见问题

- 报“No 'Access-Control-Allow-Origin'”错误

  关于出现该错误的原因及解决方案，请参见设置跨域规则后调用OSS时仍然报“No 'Access-Control-Allow-Origin'”的错误。

- 使用CDN域名访问OSS遇到跨域问题

  如果您使用CDN域名访问OSS遇到跨域问题，需在CDN控制台配置跨域规则。具体步骤，请参见CDN如何配置跨域资源共享（CORS）。

- 发送跨域请求时报 `Response to preflight request doesn't pass access control check: The value of the 'Access-Control-Allow-Origin' header in the response must not be the wildcard '*' when the request's credentials mode is 'include'.` 错误

  建议您通过设置 `xhr.withCredentials = false;` 来解决此问题。

5.13. 存储空间标签

您可以通过存储空间（Bucket）的标签功能， 对Bucket进行分类管理，例如通过设置不同的标签来标记不同用途的Bucket，设置对拥有指定标签的Bucket设置访问权限等。

注意事项

Bucket标签使用一组键值对（Key-Value）来标记Bucket，您可以通过Bucket标签标记不同用途的Bucket，并进行分类管理。

- 只有Bucket拥有者及授予 `oss:PutBucketTagging` 权限的用户才能为Bucket设置标签，否则返回403 Forbidden错误，错误码为 `AccessDenied` 。
- 每个Bucket最多可设置20对标签（Key-Value对）。
- Key和Value必须为UTF-8编码。
- Key最大长度为64字符，不能以 `http://` 、 `https://` 、 `Aliyun` 为前缀，且不能为空。
- Value最大长度为128字符，可以为空。

使用场景

当您的Bucket数量较多时，您可以Bucket标签对您的Bucket进行分类，并通过RAM策略授权指定用户（UID为193248792425xxxx）可以管理拥有指定标签的Bucket。例如授权用户A可以列举所有拥有Key为key1、Value为value1标签的Bucket，RAM Policy示例如下：

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": [
                "oss:ListBuckets"
            ],
            "Resource": [
                "acs:oss:*:193248792425xxxx:*"
            ],
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "oss:BucketTag/key1": "value1"
                }
            }
        }
    ]
}
```

您还可以授予该用户其他操作权限（Action），例如向拥有指定标签的Bucket写入数据、查看Bucket相关信息等。关于RAM Policy支持的Action信息，请参见RAM Policy概述。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**基础设置 > Bucket标签**。
4. 在**Bucket标签**区域，单击**设置**。
5. 在Bucket标签右侧输入框分别输入标签的Key和Value。

   如需添加多个标签，请单击右侧的**+**。
6. 单击**保存**。

使用阿里云SDK

以下仅列举常见SDK的设置Bucket标签的代码示例。关于其他SDK的设置Bucket标签的代码示例，请参见SDK简介。

Node.js

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.SetBucketTaggingRequest;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 设置Bucket标签。
            SetBucketTaggingRequest request = new SetBucketTaggingRequest(bucketName);
            // 依次填写Bucket标签的键（例如owner）和值（例如John）。
            request.setTag("owner", "John");
            request.setTag("location", "hangzhou");
            ossClient.setBucketTagging(request);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置Bucket标签的具体操作， 请参见添加或修改Bucket标签。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketTags。

5.14. 回源类型

5.14.1. 回源类型概述

当请求者向您的对象存储OSS请求的数据不存在时，本应返回404错误。如果您设置了回源规则，填写了数据的正确地址，请求者即可通过回源规则从OSS获取到正确的数据。回源类型分为镜像回源和重定向两种，可以满足您对于数据热迁移、特定请求的重定向等需求。

镜像回源和重定向的功能原理及使用场景的说明如下：

- 配置了镜像回源规则后，当请求者访问Bucket中一个不存在的文件（Object）时，OSS会向回源规则指定的源站获取这个文件。在获取到目标文件后，OSS会将文件返回给请求者并存入Bucket。镜像回源主要用于数据无缝迁移到OSS的场景。关于镜像回源的更多信息，请参见镜像回源。
- 配置了重定向规则后，当请求者访问Bucket发生指定错误时，OSS会将请求重定向至回源规则指定的源站。您可以利用这种跳转的功能对文件做重定向以及在此基础之上的各种业务。关于重定向的更多信息，请参见重定向。

5.14.2. 镜像回源

配置了镜像回源规则后，当请求者访问Bucket中一个不存在的文件（Object）时，OSS会向回源规则指定的源站获取这个文件。在获取到目标文件后，OSS会将文件返回给请求者并存入Bucket。

使用限制

- 规则数量

  回源规则最多配置20条，按RuleNumber的先后顺序依次匹配。如果命中当前规则，则后续规则不再匹配。规则未命中表示没有匹配回源条件，与回源后是否成功获取目标文件无关。

- QPS和流量限制

  中国各地域默认QPS为2000、流量为2 Gbit/s；海外各地域默认QPS为1000、流量为1 Gbit/s。

- 回源地址

  回源地址不支持内网地址。回源地址中支持增加图片处理参数，但不支持增加视频截帧参数。

- 默认超时时间

  镜像回源默认超时时间为10秒。

使用场景

镜像回源主要用于数据无缝迁移到OSS的场景。例如某服务已经在自己建立的源站或者在其他云产品上运行。现因业务发展，需要将数据迁移到OSS上，但是又不能停止服务，此时可以在迁移数据的同时，使用镜像回源功能保证业务的正常进行。关于使用案例的更多信息，请参见互联网公司业务无缝迁移至阿里云OSS。

使用流程

镜像回源具体流程如下图所示。

回源规则

- 回源规则触发条件

  只有当GetObject本应该返回404的情况下，OSS才会执行镜像回源，向源站请求文件。

- 回源文件命名规则

  OSS向源站请求的URL为 `http(s)://MirrorURL/ObjectName` ，回源到OSS的文件名为ObjectName。例如某Bucket设置的回源地址为 `https://aliyun.com` ，某用户请求的文件*example.jpg*不在该Bucket中。则OSS会通过 `https://aliyun.com/example.jpg` 获取文件，存储到OSS的文件名为*example.jpg*。

- 回源请求失败返回规则

  如果镜像源也不存在此文件，即镜像源返回给OSS的HTTP状态码为404，那么OSS也会返回404给用户。如果是其他非200的状态码（包括因为网络原因等获取不到文件的错误情况），OSS将返回 `424 MirrorFailed` 给用户。

- x-oss-tag响应头

  通过镜像回源的文件会添加一个 `x-oss-tag` 响应头，值为 `MIRROR` 。格式为 `x-oss-tag:MIRROR` 。

  文件回源到OSS后，只要文件不被覆盖，每次下载这个文件都会添加这个头部，用于表示这个文件来源于镜像回源。

- 回源文件更新规则

  若某个文件已经通过镜像回源到OSS，源站的源文件发生了变化，OSS不会更新该文件。

- 回源文件元信息

  OSS会将源站返回的以下HTTP头存为OSS文件的元信息：

```
Content-Type
Content-Encoding
Content-Disposition
Cache-Control
Expires
Content-Language
Access-Control-Allow-Origin
```

- HTTP请求规则
  - 传给OSS的Header信息不会传递给源站，QueryString信息是否会传递给源站取决于回源规则中的配置。
  - 如果源站是chunked编码返回，那么OSS返回给用户的也是chunked编码。

使用OSS控制台

通过控制台配置多条回源规则时，默认按规则创建时间的先后顺序依次匹配。如果您希望自定义规则匹配顺序，请通过规则右侧的**上移**或**下移**操作来实现。

当请求者访问目标Bucket中不存在的文件时，可以通过指定回源条件和回源地址，从源站中获取目标文件。例如您在华东1（杭州）有名为examplebucket的Bucket，您希望请求者访问Bucket根目录下*examplefolder*目录中不存在的文件时，可以从 `https://www.example.com/` 站点的*examplefolder*目录获取目标文件。配置步骤如下：

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**基础设置 > 镜像回源**。
4. 单击**设置**，然后单击**创建规则**。
5. 在**创建规则**面板，按以下说明配置必要参数，其他参数保留默认配置。

参数	配置
回源类型	选中**镜像**。
回源条件	选中**文件名前缀**，并设置为**examplefolder/**。 **说明**　配置单条回源规则时文件名前缀和后缀可选填；配置多条回源规则时，必须设置不同的文件名前缀或后缀区分不同的回源规则。
回源地址	第一列设置为**https**，第二列设置为**www.example.com**，第三列设置为**examplefolder**。

6. 单击**确定**。

   规则配置完成后的访问流程如下：

   i. 请求者首次访问 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/examplefolder/example.txt` 。

   ii. 如果examplebucket中不存在*examplefolder/example.txt*文件，则OSS向 `https://www.example.com/examplefolder/example.txt` 发起请求。

   iii. 如果获取到目标文件，OSS将*example.txt*存入examplebucket的*examplefolder*目录，并将文件返回给请求者；如果未获取到文件，则返回404错误给请求者。

以上配置步骤仅满足镜像回源的基础应用场景，如果您需要配置其他镜像回源规则以满足特定的应用场景时，请参见镜像回源特殊配置。

使用阿里云SDK

以下仅列举常见SDK的配置镜像回源规则的代码示例。关于其他SDK的配置镜像回源规则代码示例，请参见SDK简介。

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.RoutingRule;
import com.aliyun.oss.model.SetBucketWebsiteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            SetBucketWebsiteRequest request = new SetBucketWebsiteRequest(bucketName);
            // 设置默认主页后，访问以非正斜线（/）结尾的Object，且该Object不存在时的行为。
            //request.setSubDirType(null);
            // 指定访问子目录时，是否支持转到子目录下的默认主页。
            //request.setSupportSubDir(false);
            List<RoutingRule> routingRules = new ArrayList<RoutingRule>();
            RoutingRule rule = new RoutingRule();
            rule.setNumber(1);
            // 只有匹配此前缀的Object才能匹配此规则。
            rule.getCondition().setKeyPrefixEquals("examplebucket");
            // 访问指定Object时，返回status 404才能匹配此规则。
            rule.getCondition().setHttpErrorCodeReturnedEquals(404);
            // 指定跳转的类型。
            rule.getRedirect().setRedirectType(RoutingRule.RedirectType.Mirror);
            // 指定镜像回源的源站地址。例如https://www.example.com/。
            rule.getRedirect().setMirrorURL("<yourMirrorURL>");
            //rule.getRedirect().setMirrorRole("AliyunOSSMirrorDefaultRole");
            // 指定执行跳转或者镜像回源规则时，是否携带请求参数。
            rule.getRedirect().setPassQueryString(true);
            // 与PassQueryString作用相同，优先级高于PassQueryString。只有设置RedirectType为Mirror时生效。
            rule.getRedirect().setMirrorPassQueryString(true);
            // 指定跳转时返回的状态码。只有设置RedirectType为External或者AliCDN时生效。
            //rule.getRedirect().setHttpRedirectCode(302);
            // 指定跳转时的域名，域名需符合域名规范。
            //rule.getRedirect().setHostName("oss.aliyuncs.com");
            // 指定跳转时的协议。只有设置RedirectType为External或者AliCDN时才生效。
            //rule.getRedirect().setProtocol(RoutingRule.Protocol.Https);
            // Redirect时Object名称将替换成ReplaceKeyWith指定的值，ReplaceKeyWith支持设置变量。
            //rule.getRedirect().setReplaceKeyWith("${key}.jpg");
            // 如果设置此字段为true，则Object的前缀将被替换为ReplaceKeyPrefixWith指定的值。
            rule.getRedirect().setEnableReplacePrefix(true);
            // Redirect时Object名称的前缀将替换成该值。
            rule.getRedirect().setReplaceKeyPrefixWith("examplebucket");
            // 是否检查回源body的MD5。只有设置RedirectType为Mirror时生效。
            rule.getRedirect().setMirrorCheckMd5(true);
            RoutingRule.MirrorHeaders mirrorHeaders = new RoutingRule.MirrorHeaders();
            // 是否透传除以下Header之外的其他Header到源站。只有设置RedirectType为Mirror时生效。
            mirrorHeaders.setPassAll(false);
            List passes = new ArrayList<String>();
            passes.add("cache-control");
            // 透传指定的Header到源站。只有设置RedirectType为Mirror时生效。
            mirrorHeaders.setPass(passes);
            List removes = new ArrayList<String>();
            removes.add("content-type");
            // 禁止透传指定的Header到源站。只有设置RedirectType为Mirror时生效。
            mirrorHeaders.setRemove(removes);
            List sets = new ArrayList<Map<String, String>>();
            Map header1 = new HashMap<String, String>();
            header1.put("Key", "key1");
```

```
            header1.put("Value", "value1");
            Map header2 = new HashMap<String, String>();
            header2.put("Key", "key2");
            header2.put("Value", "value2");
            sets.add(header1);
            sets.add(header2);
            // 设置传到源站的Header。不管请求中是否携带这些指定的Header，回源时都会设置这些Header。
            mirrorHeaders.setSet(sets);
            // 指定回源时携带的Header。只有设置RedirectType为Mirror时才生效。
            rule.getRedirect().setMirrorHeaders(mirrorHeaders);
            routingRules.add(rule);
            request.setRoutingRules(routingRules);
            ossClient.setBucketWebsite(request);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil配置镜像回源规则的具体操作，请参见添加或修改Website配置。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketWebsite。

5.14.3. 重定向

配置了重定向规则后，当请求者访问Bucket发生指定错误时，OSS会将请求重定向至回源规则指定的源站。您可以利用这种跳转的功能对文件做重定向以及在此基础之上的各种业务。

使用限制

- 规则数量

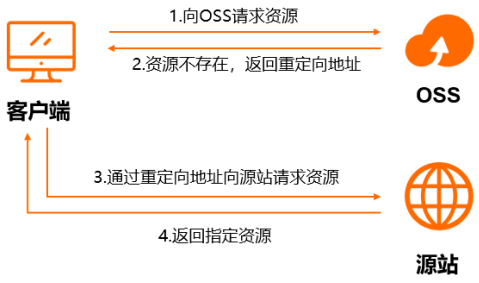
  回源规则最多配置20条，按RuleNumber的先后顺序依次匹配。如果命中当前规则，则后续规则不再匹配。规则未命中表示没有匹配回源条件，与回源后是否成功获取目标文件无关。

- 回源地址

  回源地址不支持内网地址。

使用流程

重定向功能的作用是根据设置的回源条件，以及相应的跳转的配置，向用户返回一个3xx跳转。具体流程如下图所示。



使用场景

- 其他数据源向OSS的无缝迁移

  当您将客户端的数据源异步地迁移至OSS，在此过程中未迁移到OSS的数据通过URL重写的方式返回给用户一个302重定向请求，您的客户端根据302中的Location从数据源读取数据。

- 配置页面跳转功能

  例如您希望隐藏某些前缀开头的Object，给请求者返回一个特殊的页面。

- 配置发生404或500错误时的跳转页面

  发生以上错误时，请求者可以看到一个预先设定的页面，不在系统发生错误的时候向请求者完全暴露OSS错误。

使用OSS控制台

当访问者访问Bucket出错时，可以通过指定回源条件和回源地址，跳转到源站继续访问。例如您在华东1（杭州）有名为examplebucket的Bucket，您希望请求者访问Bucket根目录下*examplefolder*目录中的文件不存在时，跳转到 `https://www.example.com/` 站点的*examplefolder*目录获取目标文件。

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**基础设置 > 镜像回源**。
4. 单击**设置**，然后单击**创建规则**。
5. 在**创建规则**面板，按以下说明配置必要参数，其他参数保留默认配置。

参数	配置
回源类型	选中**重定向**。
回源条件	○ 选中**HTTP状态码**，并设置为**404**。 HTTP状态码的取值范围为400~599。各状态码代表的错误信息，请参见常见错误排查。 ○ 选中**文件名前缀**，并设置为**examplefolder/**。 **说明** 配置单条回源规则时文件名前缀和后缀可选填；配置多条回源规则时，必须设置不同的文件名前缀或后缀区分不同的回源规则。
回源地址	选择**添加前后缀**，并将第一列设置为**https**，第二列设置为**www.example.com**，其他置空。

6. 单击**确定**。

   规则配置完成后的访问流程如下：

   i. 请求者首次访问 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/examplefolder/example.txt` 。
   ii. 如果examplebucket中不存在*examplefolder/example.txt*文件，则OSS向请求者返回301状态码，并提供重定向的地址 `https://www.example.com/examplefolder/example.txt` 。
   iii. 请求者访问 `https://www.example.com/examplefolder/example.txt` 。

   如果您还涉及以下使用场景，可按场景配置以下参数：

场景	参数
OSS文件名前缀与源站不一致。	选中**是否替换或截取前缀**，并设置**回源地址**第三列内容。OSS会将**文件名前缀**的内容替换为**回源地址**第三列的内容。 配置**文件名前缀**后可配置此项。
将OSS请求中的queryString传递到源站。	选中**携带请求字符串**。
需要替换重定向状态码。	重定向规则默认状态码为301，您可以在**重定向Code**下拉框将状态码修改为**302**或**307**。
重定向请求来源为阿里云CDN。	选择是否选中**来源为阿里CDN**。 重定向来源为阿里云CDN的时候，如果选中**来源为阿里CDN**，CDN会自动去跟随重定向规则再去拉取内容；如果不选中**来源为阿里CDN**，则CDN直接返回重定向的地址给客户端。

5.15. 删除存储空间

当您不再需要保留某个存储空间（Bucket）时，可将其删除，以免产生额外费用。

警告 Bucket删除后不可恢复，请谨慎操作。

前提条件

- 已删除Bucket中所有的文件（Object）。
  - 手动删除少量文件的具体操作，请参见删除文件。
  - 如果您的文件数量较多，建议结合生命周期规则进行批量删除。删除大量文件的具体操作，请参见生命周期规则。

  注意

  如果Bucket已开启版本控制，请确保删除Bucket内的所有当前版本和历史版本文件。删除所有版本文件的具体操作，请参见版本控制相关操作。

- 已删除Bucket中因分片上传或断点续传产生的碎片（Part）。删除碎片的具体操作，请参见管理碎片。
- 已删除Bucket中所有的Livechannel。删除Livechannel的具体操作，请参见DeleteLiveChannel。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**基础设置 > 删除Bucket**。
4. 单击**删除Bucket**，然后在弹出的对话框中，单击**确定**。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成删除Bucket的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的删除Bucket的代码示例。关于其他SDK的删除Bucket的代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 删除存储空间。
            ossClient.deleteBucket(bucketName);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil删除Bucket的具体操作， 请参见删除Bucket。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见DeleteBucket。

常见问题

- 无法删除Bucket怎么办？

  Bucket无法删除的原因是Bucket不为空或者权限不足，请按如下方式进行排查：

  - Bucket不为空

    Bucket不为空不允许删除，请确保Bucket中存储的文件（Object）、碎片（Part）以及Livechannel已经全部删除。

  - 权限不足

    Bucket为空，但是RAM用户权限不足。请按如下方式添加或修改权限：

    - 没有 `oss:DeleteBucket` 权限：如果您作为RAM用户无法删除Bucket，则请求拥有管理员权限的RAM用户在RAM Policy为您添加 `oss:DeleteBucket` 权限。
    - `oss:DeleteBucket` 授权效力为Deny：如果您在RAM Policy中拥有 `oss:DeleteBucket` 权限但仍然无法删除Bucket，则Bucket Policy可能包含授权效力为Deny的 `oss:DeleteBucket` 权限。您必须将Deny修改为Allow或者直接删除此Bucket Policy，然后才能删除此Bucket。

- 删除Bucket后，无法再次创建同名Bucket？

  删除Bucket后，需要等待半小时左右的时间才能再次创建同名的Bucket。

5.16. 常见问题

5.16.1. CDN回源到OSS时，如何隐藏OSS返回的报错信息？

如下所示，通过阿里云CDN或者第三方CDN回源到OSS，如果绑定的Bucket对应Endpoint不正确，通过CDN访问会返回如下类似的错误。错误信息中暴露了OSS的外网地址，造成了信息泄露，增加了被攻击的风险。

您需要通过以下两种方法隐藏OSS返回的报错信息：

- 通过静态网站托管配置默认404页

  通过静态网站托管配置默认404页，例如 `https://help.error.html` 。配置完成后，如果CDN访问OSS的资源不存在时，OSS将返回默认404页，从而隐藏报错中的隐私信息。具体操作，请参见设置静态网站托管。

- 通过CDN边缘脚本（EdgeScript）的方式进行改写或重定向

  例如，当用户请求OSS返回403状态码时，通过以下EdgeScript规则，将回源和缓存的URL跳转至 `https://www.error.html` ：

```
$status = get_status()
if eq($status,403){
  rewrite('https://www.error.html, 'redirect')
}
```

  以上EdgeScript规则中的状态码以及跳转的URL可结合实际使用场景相应替换。

  更多信息，请参见定制化改写和重定向。

5.16.2. 为什么CDN回源私有Bucket时，不支持访问Bucket的默认首页？

问题原因：开启回源私有Bucket后，CDN回源时默认会带上签名信息，即非匿名访问。而触发静态网站默认首页的请求必须是匿名请求。

解决方法：通过CDN控制台配置重写规则。其中，待重写的URI配置为支持根目录访问的 `^/$` ，目标URI配置为 `/index.html` ，执行规则选择**Redirect**。配置完成后，当客户端请求 `www.example.com/` 时，CDN节点将返回302让客户端重新请求 `www.example.com/index.html` 的内容。

有关URI重写规则的配置步骤，请参见配置URI重写规则。

6.对象/文件（Object）

6.1. 对象概述

对象（Object）是OSS存储数据的基本单元，也被称为OSS的文件。和传统的文件系统不同，Object没有文件目录层级结构的关系。

Object类型

Object包含以下三种类型：

- 通过简单上传生成的Object类型为Normal。
- 通过分片上传生成的Object类型为Multipart。
- 通过追加上传生成的Object类型为Appendable，且仅支持在Appendable类型的Object后直接追加内容。

> **注意**
>
> 不支持在不同类型的Object之间相互转换。例如，Normal类型的Object无法转换为Multipart或者Appendable类型。

Object信息

Object包含以下信息：

- Key：Object的名称，可用于查询Object。
- Data：您存储的数据，可由任意长度的字节组成。
- Version ID：将Object上传至开启版本控制的存储空间（Bucket）后，OSS生成标识该对象的版本ID。
- Object Meta：Object元信息，是一组键值对，表示了Object的一些属性，例如最后修改时间、大小等信息。您也可以在元信息中存储一些自定义的信息。

访问控制

针对存放在Bucket中Object的访问，OSS提供了多种权限控制策略。例如基于资源的授权策略Bucket Policy和读写权限（ACL）、基于用户的授权策略RAM Policy、通过STS临时授权访问OSS以及通过防盗链对访问来源设置白名单。关于访问控制的更多信息，请参见访问控制概述。

授权访问

OSS资源（Bucket和Object）默认是私有读写权限。如果您希望其他人访问这些资源，必须进行授权。例如您网站上的图片和视频存储在OSS上，您可以通过以下两种方式授权第三方用户通过网站访问这些资源：

- 将资源的读写权限设置为公共读。具体操作，请参见设置Object读写权限。
- 对资源的访问URL进行签名操作。具体操作，请参见授权给第三方访问。

6.2. 对象命名

与传统文件系统中的层级结构不同，OSS内部使用扁平结构存储数据。即所有数据均以对象（Object）的形式保存在存储空间（Bucket）中。对象（Object）是OSS存储数据的基本单元，也被称为OSS的文件。OSS通过键名（Key）唯一标识存储的Object。

命名规范

Object的命名规范如下：

- 使用UTF-8编码。
- 长度必须在1~1023字符之间。
- 不能以正斜线（/）或者反斜线（\）开头。

命名示例

根据Object存储于Bucket内的不同位置，其Key的表示方法也有所区别，具体说明如下：

Object所在Bucket的位置	Key的表示方法
目标存储空间examplebucket根目录下存放了名为exampleobject.txt的Object	exampleobject.txt
目标存储空间examplebucket根目录下的destdir目录中存放了exampleobject.jpg的Object	destdir/exampleobject.jpg

6.3. 上传文件

6.3.1. 简单上传

简单上传指的是使用OSS API中的PutObject接口上传小于5 GB的单个文件（Object），适用于一次HTTP请求交互即可完成上传的场景。

前提条件

已创建存储空间（Bucket）。详情请参见创建存储空间。

注意事项

- 文件大小限制

  简单上传的Object的大小不能超过5 GB。超过5 GB的Object上传请使用分片上传。
- 文件命名规则
  - 使用UTF-8编码。
  - 长度必须在1~1023字符之间。
  - 不能以正斜线（/）或者反斜线（\）字符开头。

- 上传的安全及授权

  为了防止第三方未经授权往您的Bucket里上传数据，OSS提供了Bucket和Object级别的访问权限控制。更多信息，请参见访问控制。

  为了授权给第三方上传，OSS还提供了账号级别的授权。具体操作，请参见授权给第三方上传。

- 防止同名文件被覆盖

  OSS的文件上传默认会覆盖同名文件， 为防止文件被意外覆盖，您可以通过以下方式保护您的文件：

  - 开启版本控制功能

    开启版本控制功能后，被覆盖的文件会以历史版本的形式保存下来。您可以随时恢复历史版本文件。更多信息，请参见版本控制介绍。

  - 在上传请求中携带禁止覆盖同名文件的参数

    在上传请求的Header中携带参数x-oss-forbid-overwrite，并指定其值为*true*。当您上传的文件在OSS中存在同名文件时，该文件会上传失败，并返回 `FileAlreadyExists` 错误。当不携带此参数或此参数的值为*false*时，同名文件会被覆盖。

- 文件上传性能调优

  如果您在上传大量文件时，在命名上使用了顺序前缀（如时间戳或字母顺序），可能会出现大量文件索引集中存储于存储空间中某个特定分区的情况，此时如果您的请求速率过大，会导致请求速率下降。建议您在上传大量文件时，不要使用顺序前缀的文件名，而是将顺序前缀改为随机性前缀。具体操作，请参见OSS性能与扩展性最佳实践。

使用OSS控制台

> **说明** 金融云下的OSS没有公网地域，无法通过控制台上传文件，请通过SDK、ossutil、ossbrowser等方式上传。

1. 登录OSS管理控制台。
2. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket名称。
3. 在**文件管理**页签，单击**上传文件**。
4. 在**上传文件**面板，按如下说明配置各项参数。
   i. 设置基础选项。

参数	说明
上传到	设置文件上传到目标Bucket后的存储路径。 ■ **当前目录**：将文件上传到当前目录。 ■ **指定目录**：将文件上传到指定目录，您需要输入目录名称。若输入的目录不存在，OSS将自动创建对应的文件目录并将文件上传到该目录中。 目录命名规范如下： ■ 请使用符合要求的UTF-8字符；长度必须在1~254字符之间。 ■ 不允许以正斜线（/）或反斜线（\）开头。 ■ 不允许出现连续的正斜线（/）。 ■ 不允许出现名为 `..` 的目录。
文件ACL	设置文件读写权限ACL。 ■ **继承Bucket**：以Bucket读写权限为准。 ■ **私有**（推荐）：只有文件Owner拥有该文件的读写权限，其他用户没有权限操作该文件。 ■ **公共读**：文件Owner拥有该文件的读写权限，其他用户（包括匿名访问者）都可以对文件进行访问，这有可能造成您数据的外泄以及费用激增，请谨慎操作。 ■ **公共读写**：任何用户（包括匿名访问者）都可以对文件进行访问，并且向该文件写入数据。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益。除特殊场景外，不建议您配置公共读写权限。 有关文件ACL的更多信息，请参见Object ACL。
上传加速	开启Bucket传输加速后，如果您希望加速上传文件，请打开**上传加速**开关。 有关传输加速的更多信息，请参见传输加速。
待上传文件	选择您需要上传的文件或文件夹。 您可以单击**扫描文件**或**扫描文件夹**选择本地文件或文件夹，或者直接拖拽目标文件或文件夹到待上传文件区域。 如果上传文件夹中包含了无需上传的文件，请单击目标文件右侧的**移除**将其移出文件列表。 **注意** ■ 在未开启版本控制Bucket中上传文件时，如果上传的文件与已有文件同名，则覆盖已有文件。 ■ 在已开启版本控制Bucket中上传文件时，如果上传的文件与已有文件同名，则上传文件将成为最新版本，已有文件将成为历史版本。

ii. （可选）设置文件存储类型、加密方式等高级选项。

参数	说明
存储类型	设置文件存储类型。 ■ **继承Bucket**：以Bucket存储类型为准。 ■ **标准存储**：适用于访问量较高的文件。 ■ **低频访问存储**：适用于月均访问频率低于1~2次的文件，最少需存储30天，且访问文件时会产生数据取回费用。 ■ **归档存储**：适用于基本不被访问的归档文件，最少需存储60天。文件需解冻（约1分钟）后访问，解冻会产生数据取回费用。 ■ **冷归档**：适用于一些需要长期存储的备份文件、原始数据等，最少需存储180天。文件需解冻后访问，解冻时间根据数据大小和选择的解冻模式决定，解冻会产生数据取回费用。 更多详情，请参见存储类型介绍。

服务端加密方式	设置文件的服务端加密方式。
	■ **继承Bucket**：以Bucket的服务器端加密方式为准。
	■ **OSS完全托管**：使用OSS托管的密钥进行加密。OSS会为每个Object使用不同的密钥进行加密，作为额外的保护，OSS会使用定期轮转的主密钥对加密密钥本身进行加密。
	■ **KMS**：使用KMS默认托管的CMK或指定CMK ID进行加解密操作。KMS对应的**加密秘钥**说明如下：
	  ■ **alias/acs/oss**：使用默认托管的CMK生成不同的密钥来加密不同的Object，并且在Object被下载时自动解密。
	  ■ CMK ID：使用指定的CMK生成不同的密钥来加密不同的Object，并将加密Object的CMK ID记录到Object的元信息中，具有解密权限的用户下载Object时会自动解密。选择指定的CMK ID前，您需在KMS管理控制台创建一个与Bucket相同地域的普通密钥或外部密钥。
	■ **加密算法**：仅支持AES256。
	有关文件ACL的更多信息，请参见Object ACL。

用户自定义元数据	用于为Object添加描述信息。您可以添加多条自定义元信息（User Meta），但所有的自定义元信息总大小不能超过8 KB。添加自定义元信息时，要求参数以 `x-oss-meta-` 为前缀，并为参数赋值，例如**x-oss-meta-location:hangzhou**。

iii. 单击**上传文件**。
此时，您可以在**上传列表**页签查看各个文件的上传进度。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成简单上传的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的简单上传的代码示例。关于其他SDK的简单上传的代码示例，请参见SDK简介。

Ar[redacted]va

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.PutObjectRequest;
import java.io.File;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，完整路径中不能包含Bucket名称，例如exampledir/exampleobject.txt。
        String objectName = "exampledir/exampleobject.txt";
        // 填写本地文件的完整路径，例如D:\\localpath\\examplefile.txt。
        // 如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
        String filePath= "D:\\localpath\\examplefile.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 创建PutObjectRequest对象。
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, objectName, new File(filePath));
            // 如果需要上传时设置存储类型和访问权限，请参考以下示例代码。
            // ObjectMetadata metadata = new ObjectMetadata();
            // metadata.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard.toString());
            // metadata.setObjectAcl(CannedAccessControlList.Private);
            // putObjectRequest.setMetadata(metadata);
            // 上传文件。
            ossClient.putObject(putObjectRequest);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil简单上传的具体操作， 请参见简单上传。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutObject。

更多参考

- 在使用简单上传的情况下，可以携带Object Meta信息对Object进行描述，例如可以设置Content-Type等标准HTTP头，也可以设置自定义信息。关于Object Meta的更多信息，请参见设置文件元信息。
- 在文件上传到OSS后，您可以通过上传回调向指定的应用服务器发起回调请求。具体操作，请参见上传回调。
- 如果上传的是图片，您还可以在上传后对图片进行压缩、添加自定义样式等操作。具体操作，请参见图片处理操作方式。

6.3.2. 分片上传

对象存储OSS提供分片上传功能，可以将待上传的文件分成多个碎片（Part）分别上传，上传完成之后再调用CompleteMultipartUpload接口将这些Part组合成一个Object。

前提条件

已创建存储空间（Bucket）。详情请参见创建存储空间。

使用场景

- 大文件加速上传

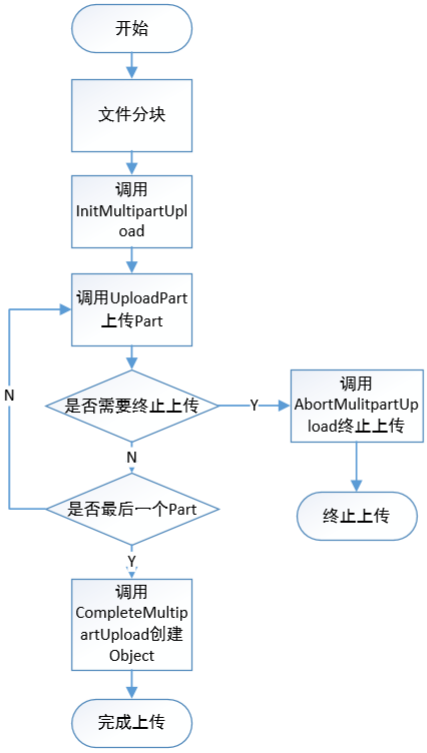
  当文件大小超过5 GB时，使用分片上传可实现并行上传多个Part以加快上传速度。
- 网络环境较差

  网络环境较差时，建议使用分片上传。当出现上传失败的时候，您仅需重传失败的Part。
- 文件大小不确定

  可以在需要上传的文件大小还不确定的情况下开始上传，这种场景在视频监控等行业应用中比较常见。

分片上传流程

分片上传的基本流程如下图所示。



流程说明如下：

1. 将待上传文件按照一定大小进行分片。
2. 使用InitiateMultipartUpload接口初始化一个分片上传任务。
3. 使用UploadPart接口上传分片。

   文件切分成Part之后，文件顺序是通过上传过程中指定的 `partNumber` 来确定，所以您可以并发上传这些碎片。并发数并非越多越快，请结合自身网络状况和设备负载综合考虑。

   如果您希望终止上传任务，可调用AbortMultipartUpload接口，成功上传的Part会一并删除。
4. 使用CompleteMultipartUpload接口将Part组合成一个Object。

使用限制

限制项	规格
文件大小	不超过48.8 TB
Part数量	1~10, 000个
单个Part大小	最小值为100 KB，最大值为5 GB。最后一个Part的大小无限制。
单次ListParts请求返回的Part最大数量	1, 000个
单次ListMultipartUploads请求返回的Multipart Upload事件最大数量	1, 000个

注意事项

- 文件上传性能调优

  如果您在上传大量文件时，在命名上使用了顺序前缀（如时间戳或字母顺序），可能会出现大量文件索引集中存储于存储空间中某个特定分区的情况。此时如果您的请求速率过大，会导致请求速率下降。建议您在上传大量文件时，不要使用顺序前缀的文件名。更多信息，请参见OSS性能与扩展性最佳实践。
- 文件覆盖

  上传同名文件会覆盖OSS中已有文件。您可以通过以下方式防止文件被意外覆盖：
  - 开启版本控制功能

    开启版本控制功能后，被覆盖的文件会以历史版本的形式保存下来，您可以随时恢复历史版本文件。更多信息，请参见版本控制介绍。
  - 在上传请求中携带禁止覆盖同名文件的参数

    在上传请求的header中携带x-oss-forbid-overwrite参数，并指定其值为 `true` 。当您上传的文件在OSS中存在同名文件时，该文件会上传失败，并返回 `FileAlreadyExists` 错误。更多信息，请参见InitiateMultipartUpload。
- 删除Part

分片上传过程被中断后，已上传的Part会一直保存在Bucket中。如果您不再需要这些Part，请通过以下方式删除，以免产生额外存储费用。

- 手动删除Part，请参见管理碎片。
- 通过生命周期规则自动删除Part，请参见设置生命周期规则。

使用阿里云SDK

以下仅列举常见SDK的分片上传的代码示例。关于其他SDK的分片上传的代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，例如exampledir/exampleobject.txt。Object完整路径中不能包含Bucket名称。
        String objectName = "exampledir/exampleobject.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 创建InitiateMultipartUploadRequest对象。
            InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest(bucketName, objectName);
            // 如果需要在初始化分片时设置请求头，请参考以下示例代码。
            // ObjectMetadata metadata = new ObjectMetadata();
            // metadata.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard.toString());
            // 指定该Object的网页缓存行为。
            // metadata.setCacheControl("no-cache");
            // 指定该Object被下载时的名称。
            // metadata.setContentDisposition("attachment;filename=oss_MultipartUpload.txt");
            // 指定该Object的内容编码格式。
            // metadata.setContentEncoding(OSSConstants.DEFAULT_CHARSET_NAME);
            // 指定初始化分片上传时是否覆盖同名Object。此处设置为true，表示禁止覆盖同名Object。
            // metadata.setHeader("x-oss-forbid-overwrite", "true");
            // 指定上传该Object的每个part时使用的服务器端加密方式。
            // metadata.setHeader(OSSHeaders.OSS_SERVER_SIDE_ENCRYPTION, ObjectMetadata.KMS_SERVER_SIDE_ENCRYPTION);
            // 指定Object的加密算法。如果未指定此选项，表明Object使用AES256加密算法。
            // metadata.setHeader(OSSHeaders.OSS_SERVER_SIDE_DATA_ENCRYPTION, ObjectMetadata.KMS_SERVER_SIDE_ENCRYPTION);
            // 指定KMS托管的用户主密钥。
            // metadata.setHeader(OSSHeaders.OSS_SERVER_SIDE_ENCRYPTION_KEY_ID, "9468da86-3509-4f8d-a61e-6eab1eac****");
            // 指定Object的存储类型。
            // metadata.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard);
            // 指定Object的对象标签，可同时设置多个标签。
            // metadata.setHeader(OSSHeaders.OSS_TAGGING, "a:1");
            // request.setObjectMetadata(metadata);
            // 初始化分片。
            InitiateMultipartUploadResult upresult = ossClient.initiateMultipartUpload(request);
            // 返回uploadId，它是分片上传事件的唯一标识。您可以根据该uploadId发起相关的操作，例如取消分片上传、查询分片上传等。
            String uploadId = upresult.getUploadId();
            // partETags是PartETag的集合。PartETag由分片的ETag和分片号组成。
            List<PartETag> partETags =  new ArrayList<PartETag>();
            // 每个分片的大小，用于计算文件有多少个分片。单位为字节。
            final long partSize = 1 * 1024 * 1024L;   //1 MB。
            // 填写本地文件的完整路径。如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
            final File sampleFile = new File("D:\\localpath\\examplefile.txt");
            long fileLength = sampleFile.length();
            int partCount = (int) (fileLength / partSize);
            if (fileLength % partSize != 0) {
                partCount++;
            }
            // 遍历分片上传。
            for (int i = 0; i < partCount; i++) {
                long startPos = i * partSize;
                long curPartSize = (i + 1 == partCount) ? (fileLength - startPos) : partSize;
                InputStream instream = new FileInputStream(sampleFile);
                // 跳过已经上传的分片。
                instream.skip(startPos);
                UploadPartRequest uploadPartRequest = new UploadPartRequest();
                uploadPartRequest.setBucketName(bucketName);
                uploadPartRequest.setKey(objectName);
                uploadPartRequest.setUploadId(uploadId);
                uploadPartRequest.setInputStream(instream);
                // 设置分片大小。除了最后一个分片没有大小限制，其他的分片最小为100 KB。
```

```
                uploadPartRequest.setPartSize(curPartSize);
                // 设置分片号。每一个上传的分片都有一个分片号，取值范围是1~10000，如果超出此范围，OSS将返回InvalidArgument错误码。
                uploadPartRequest.setPartNumber( i + 1);
                // 每个分片不需要按顺序上传，甚至可以在不同客户端上传，OSS会按照分片号排序组成完整的文件。
                UploadPartResult uploadPartResult = ossClient.uploadPart(uploadPartRequest);
                // 每次上传分片之后，OSS的返回结果包含PartETag。PartETag将被保存在partETags中。
                partETags.add(uploadPartResult.getPartETag());
            }
            // 创建CompleteMultipartUploadRequest对象。
            // 在执行完成分片上传操作时，需要提供所有有效的partETags。OSS收到提交的partETags后，会逐一验证每个分片的有效性。当所有的数据分片验证通过后，OSS将把这些分
片组合成一个完整的文件。
            CompleteMultipartUploadRequest completeMultipartUploadRequest =
                    new CompleteMultipartUploadRequest(bucketName, objectName, uploadId, partETags);
            // 如果需要在完成分片上传的同时设置文件访问权限，请参考以下示例代码。
            // completeMultipartUploadRequest.setObjectACL(CannedAccessControlList.Private);
            // 指定是否列举当前UploadId已上传的所有Part。如果通过服务端List分片数据来合并完整文件时，以上CompleteMultipartUploadRequest中的partETags可为null。
            // Map<String, String> headers = new HashMap<String, String>();
            // 如果指定了x-oss-complete-all:yes，则OSS会列举当前UploadId已上传的所有Part，然后按照PartNumber的序号排序并执行CompleteMultipartUpload操作。
            // 如果指定了x-oss-complete-all:yes，则不允许继续指定body，否则报错。
            // headers.put("x-oss-complete-all","yes");
            // completeMultipartUploadRequest.setHeaders(headers);
            // 完成分片上传。
            CompleteMultipartUploadResult completeMultipartUploadResult = ossClient.completeMultipartUpload(completeMultipartUploadRequest);
            System.out.println(completeMultipartUploadResult.getETag());
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil分片上传具体操作， 请参见上传文件。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见InitiateMultipartUpload。

6.3.3. 断点续传上传

通过断点续传上传的方式将文件上传到OSS前，您可以通过Checkpoint文件指定断点记录点。上传过程中，如果出现网络异常或程序崩溃导致文件上传失败时，将从断点记录处继续上传未上传完成的部分。

前提条件

已创建存储空间（Bucket）。详情请参见创建存储空间。

注意事项

当前仅支持通过SDK的方式实现断点续传上传，断点续传上传过程中有以下注意事项：

- 使用断点续传上传时，文件上传的进度信息会记录在Checkpoint文件中，如果上传过程中某一分片上传失败，再次上传时会从Checkpoint文件中记录的点继续上传，从而达到断点续传的效果。上传完成后，Checkpoint文件会被删除。
- SDK会将上传的状态信息记录在Checkpoint文件中，所以要确保程序对Checkpoint文件有写权限。
- 请勿修改Checkpoint文件中携带的校验信息。如果Checkpoint文件损坏，则会重新上传所有分片。
- 如果上传过程中本地文件发生了改变，则会重新上传所有分片。

使用阿里云SDK

以下仅列举常见SDK的断点续传上传的代码示例。关于其他SDK的断点续传上传的代码示例，请参见SDK简介。

Android Java

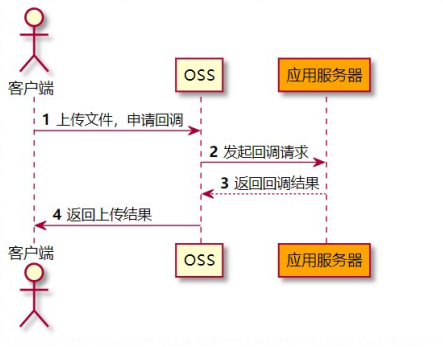
```
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
public class Demo {
    public static void main(String[] args) {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            ObjectMetadata meta = new ObjectMetadata();
            // 指定上传的内容类型。
            meta.setContentType("text/plain");
            // 文件上传时设置访问权限ACL。
            // meta.setObjectAcl(CannedAccessControlList.Private);
            // 通过UploadFileRequest设置多个参数。
            // 依次填写Bucket名称（例如examplebucket）以及Object完整路径（例如exampledir/exampleobject.txt），Object完整路径中不能包含Bucket名称。
            UploadFileRequest uploadFileRequest = new UploadFileRequest("examplebucket","exampledir/exampleobject.txt");
            // 通过UploadFileRequest设置单个参数。
            // 填写本地文件的完整路径，例如D:\\localpath\\examplefile.txt。如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
            uploadFileRequest.setUploadFile("D:\\localpath\\examplefile.txt");
            // 指定上传并发线程数，默认值为1。
            uploadFileRequest.setTaskNum(5);
            // 指定上传的分片大小，单位为字节，取值范围为100 KB~5 GB。默认值为100 KB。
            uploadFileRequest.setPartSize(1 * 1024 * 1024);
            // 开启断点续传，默认关闭。
            uploadFileRequest.setEnableCheckpoint(true);
            // 记录本地分片上传结果的文件。上传过程中的进度信息会保存在该文件中，如果某一分片上传失败，再次上传时会根据文件中记录的点继续上传。上传完成后，该文件会被删除
。
            // 如果未设置该值，默认与待上传的本地文件同路径，名称为${uploadFile}.ucp。
            uploadFileRequest.setCheckpointFile("yourCheckpointFile");
            // 文件的元数据。
            uploadFileRequest.setObjectMetadata(meta);
            // 设置上传回调，参数为Callback类型。
            //uploadFileRequest.setCallback("yourCallbackEvent");
            // 断点续传上传。
            ossClient.uploadFile(uploadFileRequest);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (Throwable ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
                // 关闭OSSClient。
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

6.3.4. 上传回调

对象存储OSS在完成文件（Object）上传时可以提供回调（Callback）给应用服务器。您只需要在发送给OSS的请求中携带相应的Callback参数，即可实现回调。

使用场景

上传回调的一种典型应用场景是结合授权第三方上传时使用。适当使用上传回调机制，能有效降低客户端的逻辑复杂度和网络消耗。上传回调流程如下：



1. 客户端在上传文件到OSS时指定到服务器端的回调。
2. 客户端的上传任务在OSS执行完毕后，OSS会向应用服务器主动发起HTTP请求进行回调。
3. 应用服务器可以及时得到上传完成的通知，进而完成诸如数据库修改等操作，并向OSS返回回调结果。
4. 在回调请求接收到服务器端的响应之后，OSS会将上传结果返回给客户端。

OSS在向应用服务器发送POST回调请求的时候，会在POST请求的Body中包含一些参数来携带特定的信息。这些参数有两种，一种是系统定义的参数，例如Bucket名称、Object名称等；另外一种是自定义的参数，您可以在发送带回调的请求给OSS时，通过使用自定义参数来携带一些和应用逻辑相关的信息，例如发起请求的用户ID等。

注意事项

目前仅简单上传（PutObject）、表单上传（PostObject）、完成分片上传（CompleteMultipartUpload）操作支持使用上传回调。

使用阿里云SDK

以下仅列举常见SDK的上传回调的代码示例。关于其他SDK的上传回调的代码示例，请参见SDK简介。

Android Java

```
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
import java.io.ByteArrayInputStream;
public class Demo {
    public static void main(String[] args) {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，例如exampledir/exampleobject.txt。Object完整路径中不能包含Bucket名称。
        String objectName = "exampledir/exampleobject.txt";
        // 您的回调服务器地址，例如https://example.com:23450或者https://127.0.0.1:9090。
        String callbackUrl = "yourCallbackServerUrl";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            String content = "Hello OSS";
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, objectName,new ByteArrayInputStream(content.getBytes()));
            // 上传回调参数。
            Callback callback = new Callback();
            callback.setCallbackUrl(callbackUrl);
            //（可选）设置回调请求消息头中Host的值，即您的服务器配置Host的值。
            // callback.setCallbackHost("yourCallbackHost");
            // 设置发起回调时请求body的值。
            callback.setCallbackBody("{\\\"mimeType\\\":${mimeType},\\\"size\\\":${size}}");
            // 设置发起回调请求的Content-Type。
            callback.setCalbackBodyType(Callback.CalbackBodyType.JSON);
            // 设置发起回调请求的自定义参数，由Key和Value组成，Key必须以x:开始。
            callback.addCallbackVar("x:var1", "value1");
            callback.addCallbackVar("x:var2", "value2");
            putObjectRequest.setCallback(callback);
            PutObjectResult putObjectResult = ossClient.putObject(putObjectRequest);
            // 读取上传回调返回的消息内容。
            byte[] buffer = new byte[1024];
            putObjectResult.getResponse().getContent().read(buffer);
            // 数据读取完成后，获取的流必须关闭，否则会造成连接泄漏，导致请求无连接可用，程序无法正常工作。
            putObjectResult.getResponse().getContent().close();
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (Throwable ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见Callback。

更多参考

- 关于上传回调中的常见错误及错误排查，请参见上传回调错误及排除。
- 关于基于Post Policy的使用规则在服务端通过各语言SDK代码完成签名，并且设置上传回调，然后通过表单直传数据到OSS的具体操作，请参见服务端签名直传并设置上传回调。
- 关于搭建基于OSS的移动应用数据直传服务并设置上传回调的具体操作，请参见快速搭建移动应用上传回调服务。

6.3.5. 授权给第三方上传

本文介绍如何通过签名URL以及临时访问凭证OSS授权第三方直接上传文件（Object）到OSS。

前提条件

已创建存储空间（Bucket）。详情请参见创建存储空间。

背景信息

在典型的Client/Server架构中，服务器端负责接收并处理客户端的请求，并将OSS作为后端的存储服务。客户端将要上传的文件发送给服务器端，然后服务器端再将数据转发上传到OSS。在这个过程中，一份数据需要在网络上传输两次，分别为从客户端到服务器端，再从服务器端到OSS。当访问量较大时，服务器端需要有足够的带宽资源来满足多个客户端同时上传的需求，这对架构的伸缩性提出了挑战。

功能优势

为了解决以上场景带来的挑战，OSS提供了授权给第三方上传的功能。使用该功能，每个客户端可以直接将文件上传到OSS而不是通过服务器端转发，不仅节省了自建服务器的成本，而且充分利用了OSS的海量数据处理能力，无需考虑带宽和并发限制等，可以让客户专心于业务处理。

目前，您可以通过签名URL以及临时访问凭证STS授权第三方上传。

临时访问凭证

OSS可以通过阿里云STS（Security Token Service）进行临时授权访问。阿里云STS是为云计算用户提供临时访问令牌的Web服务。通过STS，您可以为第三方应用或子用户（即用户身份由您自己管理的用户）颁发一个自定义时效和权限的访问凭证。关于STS的更多信息，请参见STS介绍。

STS的优势如下：

- 您无需透露您的长期密钥（AccessKey）给第三方应用，只需生成一个访问令牌并将令牌交给第三方应用。您可以自定义这个令牌的访问权限及有效期限。
- 您无需关心权限撤销问题，访问令牌过期后自动失效。

> **说明** 您可以通过调用STS服务的AssumeRole接口或者使用各语言STS SDK来获取临时访问凭证。临时访问凭证包括临时访问密钥（AccessKeyId和AccessKeySecret）和安全令牌（SecurityToken）。临时访问凭证有效时间单位为秒，最小值为900，最大值以当前角色设定的最大会话时间为准。更多信息，请参见设置角色最大会话时间。

使用阿里云SDK

以下仅列举常见SDK通过临时访问凭证授权第三方上传文件的代码示例。关于其他SDK的通过临时访问凭证授权第三方上传文件的代码示例，请参见SDK简介。

Python

```
// 填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
// 从STS服务获取的临时访问密钥（AccessKey ID和AccessKey Secret）。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 从STS服务获取的安全令牌（SecurityToken）。
String securityToken = "yourSecurityToken";
// 填写Bucket名称，例如examplebucket。
String bucketName = "examplebucket";
// 填写Object完整路径，完整路径中不能包含Bucket名称，例如exampledir/exampleobject.txt。
String objectName = "exampledir/exampleobject.txt";
// 从STS服务获取临时访问凭证后，您可以通过临时访问密钥和安全令牌生成OSSClient。
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
// 通过STS临时授权上传文件。
// 填写本地文件的完整路径。如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, objectName, new File("D:\\localpath\\examplefile.txt"));
ossClient.putObject(putObjectRequest);
// 关闭OSSClient。
ossClient.shutdown();
```

签名URL

> **注意** 由于STS临时账号以及签名URL均需设置有效时长，当您使用STS临时账号生成签名URL执行相关操作（例如上传、下载文件）时，以最小的有效时长为准。例如您的STS临时账号的有效时长设置为1200秒、签名URL设置为3600秒时，当有效时长超过1200秒后，您无法使用此STS临时账号生成的签名URL上传文件。

您可以将生成的签名URL提供给访客进行临时访问。生成签名URL时，您可以通过指定URL的过期时间来限制访客的访问时长。签名URL的默认过期时间为3600秒，最大值为32400秒。

在URL中加入签名信息，以便将该URL转给第三方实现授权访问。更多信息，请参见在URL中包含签名。

使用阿里云SDK

以下仅列举常见SDK通过签名URL授权第三方上传文件的代码示例。关于其他SDK的通过签名URL授权第三方上传文件的代码示例，请参见SDK简介。

Python

```
import com.aliyun.oss.*;
import com.aliyun.oss.common.utils.DateUtil;
import com.aliyun.oss.model.GeneratePresignedUrlRequest;
import com.aliyun.oss.model.PutObjectResult;
import java.io.File;
import java.io.FileInputStream;
import java.net.URL;
import java.util.*;
public class Demo {
    public static void main(String[] args) throws Throwable {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 从STS服务获取的安全令牌（SecurityToken）。
        String securityToken = "yourSecurityToken";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，例如exampleobject.txt。Object完整路径中不能包含Bucket名称。
        String objectName = "exampleobject.txt";
        // 填写本地文件的完整路径。如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
        String pathName = "D:\\localpath\\examplefile.txt";
        // 从STS服务获取临时访问凭证后，您可以通过临时访问密钥和安全令牌生成OSSClient。
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
        try {
            // 填写签名URL的过期时间。
            Date expiration = null;
            expiration = DateUtil.parseRfc822Date("Wed, 18 Mar 2022 14:20:00 GMT");
            // 生成签名URL。
            GeneratePresignedUrlRequest request = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.PUT);
            // 设置过期时间。
            request.setExpiration(expiration);
            // 设置ContentType。
            request.setContentType("application/txt");
            // 添加用户自定义元信息。
            request.addUserMetadata("author", "aliy");
            // 通过HTTP PUT请求生成签名URL。
            URL signedUrl = ossClient.generatePresignedUrl(request);
            System.out.println("signed url for putObject: " + signedUrl);
            // 使用签名URL发送请求。
            File f = new File(pathName);
            FileInputStream fin = null;
            fin = new FileInputStream(f);
            // 添加PutObject请求头。
            Map<String, String> customHeaders = new HashMap<String, String>();
            customHeaders.put("Content-Type", "application/txt");
            customHeaders.put("x-oss-meta-author", "aliy");
            PutObjectResult result = ossClient.putObject(signedUrl, fin, f.length(), customHeaders);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

6.3.6. 表单上传

表单上传是指使用OSS API中的PostObject请求来完成Object的上传，上传的Object不能超过5 GB。

> **说明** 表单上传的API接口详细信息请参见PostObject。

适用场景

表单上传非常适合嵌入在HTML网页中来上传Object，比较常见的场景是网站应用，以招聘网站为例：

上传方式	不使用表单上传	表单上传

上传方式	不使用表单上传	表单上传
流程对比	1. 网站用户上传简历。 2. 网站服务器回应上传页面。 3. 简历被上传到网站服务器。 4. 网站服务器再将简历上传到OSS。	1. 网站用户上传简历。 2. 网站服务器回应上传页面。 3. 简历上传到OSS。

- 从流程上来说，使用表单上传，少了一步转发流程，更加方便。
- 从架构上来说，原来的上传都统一走网站服务器，上传量过大时，需要扩容网站服务器。采用表单上传后，直接从客户端上传数据到OSS，上传量过大时，由OSS来保障服务质量。

SDK demo

请参见Java SDK。

注意事项

- 文件大小限制

  简单上传的Object的大小不能超过5 GB。超过5 GB的Object上传请使用断点续传上传。

- 文件命名规则
  - 使用UTF-8编码。
  - 长度必须在1~1023字符之间。
  - 不能以正斜线（/）或者反斜线（\）字符开头。

- 文件上传性能调优

  如果您在上传大量文件时，在命名上使用了顺序前缀（如时间戳或字母顺序），可能会出现大量文件索引集中存储于存储空间中某个特定分区的情况，此时如果您的请求速率过大，会导致请求速率下降。建议您在上传大量文件时，不要使用顺序前缀的文件名。将顺序前缀改为随机性前缀的方法请参见OSS性能与扩展性最佳实践。

- 防止同名文件被覆盖

  OSS的文件上传默认会覆盖同名文件， 为防止文件被意外覆盖，您可以通过以下方式保护您的文件：

  - 开启版本控制功能

    开启版本控制功能后，被覆盖的文件会以历史版本的形式保存下来。您可以随时恢复历史版本文件。详情请参见版本控制介绍。

  - 在上传请求中携带禁止覆盖同名文件的参数

    在上传请求的header中携带参数x-oss-forbid-overwrite，并指定其值为*true*。当您上传的文件在OSS中存在同名文件时，该文件会上传失败，并返回 `FileAlreadyExists` 错误。当不携带此参数或此参数的值为*false*时，同名文件会被覆盖。详情请参见PutObject。

流程解析

1. 构建一个Post Policy。

   Post请求的Policy表单域用于验证请求的合法性。例如可以指定上传的大小，可以指定上传的Object名称等，上传成功后客户端跳转到的URL，上传成功后客户端收到的状态码。具体请参见Post Policy。

   以Python代码为例子，Policy是json格式的字符串。

   ```
   # 设置网站用户能上传文件的过期时间为2115-01-27T10:56:19Z（这里为了测试成功写的过期时间很长，实际使用中不建议这样设置），文件大小不超过104857600字节
   policy="{\"expiration\":\"2115-01-27T10:56:19Z\",\"conditions\":[[\"content-length-range\", 0, 104857600]]}"
   ```

2. 将Policy字符串进行base64编码。
3. 用OSS的AccessKeySecret对base64编码后的Policy进行签名。
4. 构建上传的HTML页面。
5. 打开HTML页面，选择文件上传。

完整Python代码示例：

```
#coding=utf8
import md5
import hashlib
import base64
import hmac
from optparse import OptionParser
def convert_base64(input):
    return base64.b64encode(input)
def get_sign_policy(key, policy):
    return base64.b64encode(hmac.new(key, policy, hashlib.sha1).digest())
def get_form(bucket, endpoint, access_key_id, access_key_secret, out):
    #1 构建一个Post Policy
    policy="{\"expiration\":\"2115-01-27T10:56:19Z\",\"conditions\":[[\"content-length-range\", 0, 1048576]]}"
    print("policy: %s" % policy)
    #2 将Policy字符串进行base64编码
    base64policy = convert_base64(policy)
    print("base64_encode_policy: %s" % base64policy)
    #3 用OSS的AccessKeySecret对编码后的Policy进行签名
    signature = get_sign_policy(access_key_secret, base64policy)
    #4 构建上传的HTML页面
    form = '''
    <html>
        <meta http-equiv=content-type content="text/html; charset=UTF-8">
        <head><title>OSS表单上传(PostObject)</title></head>
        <body>
            <form  action="http://%s.%s" method="post" enctype="multipart/form-data">
                <input type="text" name="OSSAccessKeyId" value="%s">
                <input type="text" name="policy" value="%s">
                <input type="text" name="Signature" value="%s">
                <input type="text" name="key" value="upload/${filename}">
                <input type="text" name="success_action_redirect" value="http://oss.aliyun.com">
                <input type="text" name="success_action_status" value="201">
                <input name="file" type="file" id="file">
                <input name="submit" value="Upload" type="submit">
            </form>
        </body>
    </html>
    ''' % (bucket, endpoint, access_key_id, base64policy, signature)
    f = open(out, "wb")
    f.write(form)
    f.close()
    print("form is saved into %s" % out)
if __name__ == '__main__':
    parser = OptionParser()
    parser.add_option("", "--bucket", dest="bucket", help="specify ")
    parser.add_option("", "--endpoint", dest="endpoint", help="specify")
    parser.add_option("", "--id", dest="id", help="access_key_id")
    parser.add_option("", "--key", dest="key", help="access_key_secret")
    parser.add_option("", "--out", dest="out", help="out put form")
    (opts, args) = parser.parse_args()
    if opts.bucket and opts.endpoint and opts.id and opts.key and opts.out:
        get_form(opts.bucket, opts.endpoint, opts.id, opts.key, opts.out)
    else:
        print "python %s --bucket=your-bucket --endpoint=oss-cn-hangzhou.aliyuncs.com --id=your-access-key-id --key=your-access-key-secret --out=out-
put-form-name" % __file__
```

将此段代码保存为*post_object.py*，然后用 `python post_object.py` 来运行。

使用方法：

```
python post_object.py --bucket=您的Bucket --endpoint=Bucket对应的OSS域名 --id=您的AccessKeyId --key=您的AccessKeySecret --out=输出的文件名
```

使用示例：

```
python post_object.py --bucket=oss-sample --endpoint=oss-cn-hangzhou.aliyuncs.com --id=tphpxp --key=ZQNJzf4QJRkrH4 --out=post.html
```

说明

- 构建的表单中 `success_action_redirect value=http://oss.aliyun.com` 表示上传成功后跳转的页面。可以替换成您自己的页面。
- 构建的表单中 `success_action_status value=201` 表示的是上传成功后返回的状态码为201，可以替换。
- 假设指定生成的HTML文件为*post.html*，打开*post.html*，选择文件上传。在本示例中，如果文件上传成功，则跳转到OSS的主页面。

上传的安全及授权

为了防止第三方未经授权往您的Bucket里上传数据，OSS提供了Bucket和Object级别的访问权限控制。详情请参见权限控制。

为了授权给第三方上传，OSS还提供了账号级别的授权。详情请参见授权给第三方上传。

6.3.7. 追加上传

追加上传指的是在已上传的Appendable类型Object后面直接追加内容。

前提条件

已创建存储空间（Bucket）。详情请参见创建存储空间。

背景信息

通过简单上传生成的Object类型为Normal，通过分片上传生成的Object类型为Multipart。这两种类型Object在上传结束之后内容是固定的，只能读取，不能修改。如果Object内容发生了改变，只能重新上传同名的Object来覆盖之前的内容。

由于这一特性，如果通过上述方式上传视频监控、视频直播等领域生成的实时视频流，只能将视频流按照一定规律切分成小块然后不断地上传新的Object，在实际使用中存在以下缺点：

- 软件架构比较复杂，需要考虑文件分块等细节问题。
- 需要有位置保存元信息，比如已经生成的Object列表等，然后每次请求都重复读取元信息来判断是否有新的Object生成。这样对服务器的压力很大，而且客户端每次都需要发送两次网络请求，延时上也会有一定的影响。
- 如果Object切分较小能有效降低数据延时，但是切分过多的Object会引发管理复杂的问题。如果Object切分较大，则数据延时又会明显提升。

为了满足实时更新已上传Object内容的视频流场景，OSS提供了追加上传（AppendObject）的方式，通过这种方式生成的Object的类型为Appendable。Appendable类型Object后面允许直接追加内容，且每次追加上传的数据都能够即时可读。

功能优势

通过追加上传，可在视频数据产生之后即时将数据上传至同一个Object，而客户端只需要定时获取该Object的长度，并与上次读取的长度进行对比。如果发现有新的可读数据，则触发一次读操作来获取新上传的数据部分即可。通过这种方式可以简化架构，增强扩展性。

使用限制

- 大小限制

  Object大小不能超过5 GB。
- 命名限制
  - 使用UTF-8编码。
  - 长度必须在1~1023字符之间。
  - 不能以正斜线（/）或者反斜线（\）字符开头。
- 操作限制
  - 不支持拷贝通过追加上传方式上传的Object，但允许修改Object本身的meta信息。
  - 追加上传不支持上传回调操作。

使用阿里云SDK

以下仅列举常见SDK的追加上传的代码示例。关于其他SDK的追加上传的代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.AppendObjectRequest;
import com.aliyun.oss.model.AppendObjectResult;
import com.aliyun.oss.model.ObjectMetadata;
import java.io.ByteArrayInputStream;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，完整路径中不能包含Bucket名称，例如exampledir/exampleobject.txt。
        String objectName = "exampledir/exampleobject.txt";
        String content1 = "Hello OSS A \n";
        String content2 = "Hello OSS B \n";
        String content3 = "Hello OSS C \n";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            ObjectMetadata meta = new ObjectMetadata();
            // 指定上传的内容类型。
            meta.setContentType("text/plain");
            // 指定该Object的网页缓存行为。
            //meta.setCacheControl("no-cache");
            // 指定该Object被下载时的名称。
            //meta.setContentDisposition("attachment;filename=oss_download.txt");
            // 指定该Object的内容编码格式。
            //meta.setContentEncoding(OSSConstants.DEFAULT_CHARSET_NAME);
            // 该请求头用于检查消息内容是否与发送时一致。
            //meta.setContentMD5("ohhnqLBJFiKkPSBO1eNaUA==");
            // 指定过期时间。
            //try {
            //    meta.setExpirationTime(DateUtil.parseRfc822Date("Wed, 08 Jul 2022 16:57:01 GMT"));
            //} catch (ParseException e) {
            //    e.printStackTrace();
            //}
            // 指定服务器端加密方式。此处指定为OSS完全托管密钥进行加密（SSE-OSS）。
            //meta.setServerSideEncryption(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
```

```
            // 指定Object的访问权限。此处指定为私有访问权限。
            //meta.setObjectAcl(CannedAccessControlList.Private);
            // 指定Object的存储类型。
            //meta.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard);
            // 创建AppendObject时可以添加x-oss-meta-*，继续追加时不可以携带此参数。如果配置以x-oss-meta-*为前缀的参数，则该参数视为元数据。
            //meta.setHeader("x-oss-meta-author", "Alice");
            // 通过AppendObjectRequest设置多个参数。
            AppendObjectRequest appendObjectRequest = new AppendObjectRequest(bucketName, objectName, new ByteArrayInputStream(content1.getBytes()),m
eta);
            // 通过AppendObjectRequest设置单个参数。
            // 设置Bucket名称。
            //appendObjectRequest.setBucketName(bucketName);
            // 设置Object名称。
            //appendObjectRequest.setKey(objectName);
            // 设置待追加的内容。可选类型包括InputStream类型和File类型。此处为InputStream类型。
            //appendObjectRequest.setInputStream(new ByteArrayInputStream(content1.getBytes()));
            // 设置待追加的内容。可选类型包括InputStream类型和File类型。此处为File类型。
            //appendObjectRequest.setFile(new File("D:\\localpath\\examplefile.txt"));
            // 指定文件的元信息，第一次追加时有效。
            //appendObjectRequest.setMetadata(meta);
            // 第一次追加。
            // 设置文件的追加位置。
            appendObjectRequest.setPosition(0L);
            AppendObjectResult appendObjectResult = ossClient.appendObject(appendObjectRequest);
            // 文件的64位CRC值。此值根据ECMA-182标准计算得出。
            System.out.println(appendObjectResult.getObjectCRC());
            // 第二次追加。
            // nextPosition表示下一次请求中应当提供的Position，即文件当前的长度。
            appendObjectRequest.setPosition(appendObjectResult.getNextPosition());
            appendObjectRequest.setInputStream(new ByteArrayInputStream(content2.getBytes()));
            appendObjectResult = ossClient.appendObject(appendObjectRequest);
            // 第三次追加。
            appendObjectRequest.setPosition(appendObjectResult.getNextPosition());
            appendObjectRequest.setInputStream(new ByteArrayInputStream(content3.getBytes()));
            appendObjectResult = ossClient.appendObject(appendObjectRequest);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil追加上传的具体操作， 请参见appendfromfile（追加上传）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见AppendObject。

6.3.8. RTMP推流上传

OSS支持使用RTMP协议推送H264编码的视频流和AAC编码的音频流到OSS。推送到OSS的音视频数据可以点播播放；在对延迟不敏感的应用场景，也可以做直播用途。本文介绍如何推送音视频流到OSS，以及如何播放推送到OSS的音视频数据。

使用限制

通过RTMP协议上传音视频数据有以下限制：

- 只能使用RTMP推流的方式，不支持拉流。
- 上传的音视频数据中必须包含视频流，且视频流格式为H264。
- 上传的音视频数据中可选择是否包含音频流。若包含音频流，则只支持AAC格式的音频流，其他格式的音频流会被丢弃。
- 转储只支持HLS协议。
- 一个LiveChannel同时只能有一个客户端向其推流。

向OSS推送音视频数据

- 获得推流地址

  使用SDK调用PutLiveChannel接口，创建一个LiveChannel，并获取对应的推流地址。如果Bucket的权限控制（ACL）为公共读写（public-read-write），可直接使用获取的推流地址进行推流；如果Bucket ACL为公共读（public-read）或者私有（private），则需要进行签名操作。有关签名的具体步骤，请参见在URL中包含签名。

  以Python SDK为例，获取未签名以及签名推流地址的代码如下：

```
# -*- coding: utf-8 -*-
import oss2
# host以杭州为例，其它Region请按实际情况填写。
host = "oss-cn-hangzhou.aliyuncs.com"
# 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
accessid = "yourAccessyId"
accesskey = "yourAccessSecret"
# 填写Bucket名称。
bucket_name = "yourBucket"
# 填写LiveChannel名称，例如test-channel。
channel_name = "test-channel"
auth = Auth(accessid, accesskey)
bucket = Bucket(auth, host, bucket_name)
channel_cfg = LiveChannelInfo(target = LiveChannelInfoTarget())
channel = bucket.create_live_channel(channel_name, channel_cfg)
publish_url = channel.publish_url
# 生成RTMP推流的签名URL，并设置过期时间为3600秒。
signed_publish_url = bucket.sign_rtmp_url("test-channel", "playlist.m3u8", 3600)
# 打印未签名推流地址。
print('publish_url='+publish_url)
# 打印播放地址。
print('play_url='+play_url)
# 打印签名推流地址。
print('signed_publish_url='+signed_publish_url)
```

获得的推流地址示例如下：

```
publish_url = rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/test-channel
signed_publish_url = rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/your-channel?OSSAccessKeyId=LGarxxxxxxHjKWg6&playlistName=t.m3u8&Expires=
1472201595&Signature=bjKraZTTyzz9%2FpYoomDx4Wgh%2FlM%3D"
```

- 使用ffmpeg进行推流

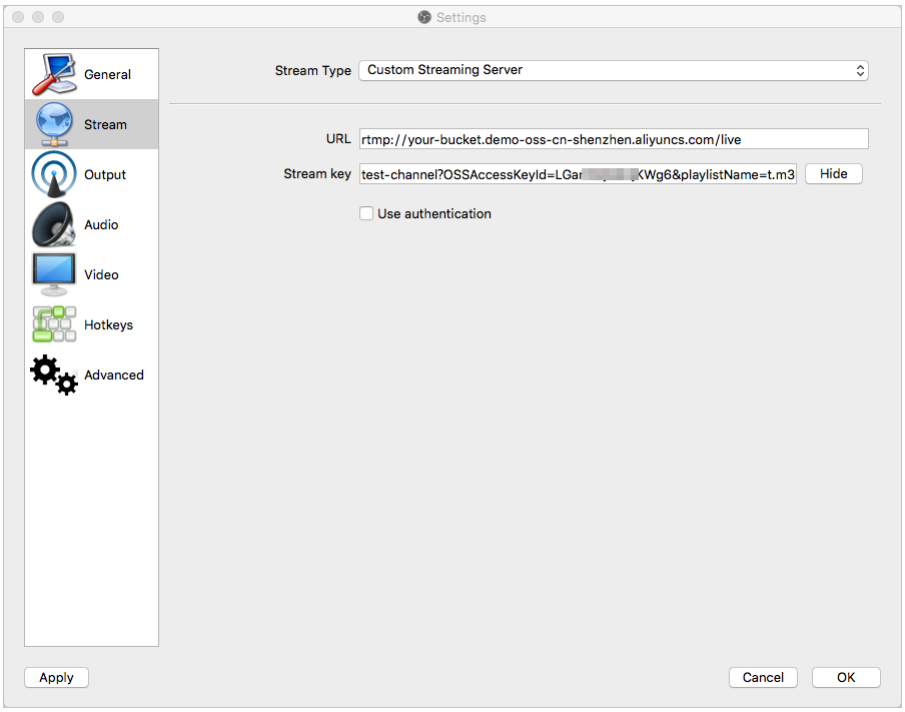
  可以使用ffmpeg推送本地的视频文件到OSS，命令如下：

```
ffmpeg -i 1.flv -c copy -f flv "rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/test-channel?OSSAccessKeyId=LGarxxxxxxHjKWg6&Expires=147219909
5&Signature=%2FAvRo7FTss1InBKgwn7Gz%2FUlp9w%3D"
```

- 使用OBS进行推流

  首先单击**Settings**，在URL文本框中输入前面步骤获取的推流地址，然后单击**OK**开始推流。

  如下图所示，请注意推流地址的拆分方式：



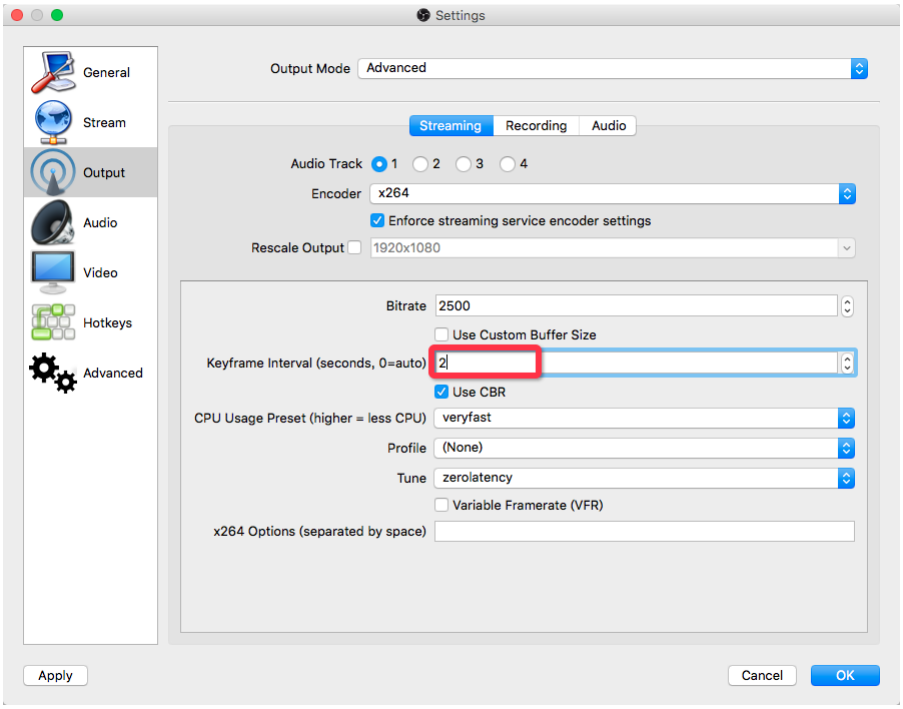
播放推送到OSS的音视频数据

- 直播场景

  在推流的过程中，可以通过HLS协议播放正在推送的内容。各个平台的播放方法如下：

  - 在Android、iOS等移动平台，直接在浏览器输入LiveChannel对应的播放地址即可。
  - Mac OS可以使用safari浏览器进行播放。
  - PC端可以安装VLC多媒体播放器进行播放。安装完成后，在VLC media player页面，选择**媒体(M)打开网络串流(N)**，然后将获取的播放地址play_url填写至**请输入网络URL**文本框。

为了直播流畅，可以设置比较小的FragDuration，例如2s；另外，GOP的大小最好固定且与LiveChannel的FragDuration配置一致。OBS的GOP（即keyframe Interval）设置方法如下：



- 点播场景

  推流的过程中，OSS总是以直播流的方式推送/更新M3U8。因此，对于点播的场景，需要在推流结束后，调用PostVodPlaylist接口来组装一个点播用的m3u8文件，然后使用该文件地址来播放。

  对于点播的场景，可以设置较大的GOP来减少ts文件数，从而降低码率。

6.4. 下载文件

6.4.1. 简单下载

简单下载指的是使用OSS API的GetObject接口，下载已上传的文件（Object），适用于一次HTTP请求交互即可完成下载的场景。

前提条件

如果要下载归档或冷归档类型的Object，请确保该Object已进入解冻状态。具体操作，请参见解冻Object。

使用OSS控制台

1. 登录OSS管理控制台。
2. 在左侧导航栏，单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，单击**文件管理**，然后下载单个或多个文件。
   - 下载单个文件

     方式一：选择目标文件右侧的**更多 > 下载**。

     方式二：单击目标文件的文件名或其右侧的**详情**，在弹出的**详情**面板中单击**下载**。
   - 下载多个文件

     选中多个文件，选择**批量操作 > 下载**。通过OSS控制台可一次批量下载最多100个文件。

   有关如何在受版本控制的Bucket中下载文件的具体操作，请参见版本控制相关操作。

使用图形化管理工具ossbrowser

ossbrowser支持Object级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成简单下载的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的简单下载的代码示例。关于其他SDK的简单下载的代码示例，请参见SDK简介。

```
package com.aliyun.oss.demo;
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.GetObjectRequest;
import java.io.File;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写不包含Bucket名称在内的Object完整路径，例如testfolder/exampleobject.txt。
        String objectName = "testfolder/exampleobject.txt";
        String pathName = "D:\\localpath\\examplefile.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 下载Object到本地文件，并保存到指定的本地路径中。如果指定的本地文件存在会覆盖，不存在则新建。
            // 如果未指定本地路径，则下载后的文件默认保存到示例程序所属项目对应本地路径中。
            ossClient.getObject(new GetObjectRequest(bucketName, objectName), new File(pathName));
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil简单下载的具体操作， 请参见下载文件。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见GetObject。

更多参考

- 为了防止第三方未经授权从您的Bucket里下载数据，OSS提供了Bucket和Object级别的访问权限控制。更多信息，请参见访问控制概述。
- 如果您希望将私有Bucket的Object提供给第三方进行下载，请通过STS临时访问凭证或签名URL的方式授权第三方下载文件。具体操作，请参见授权给第三方下载。
- 如果您希望在下载大文件过程中，从下载中断的位置继续下载未完成的部分，可选择断点续传下载。

6.4.2. 断点续传下载

OSS提供了从Object指定的位置开始下载的功能，在下载大的Object的时候，可以分多次下载。如果下载中断，重启时也可以从上次完成的位置开始继续下载。

前提条件

如果要下载归档或冷归档类型的Object，请确保该Object已进入解冻状态。具体操作，请参见解冻Object。

注意事项

当前仅支持通过SDK的方式实现断点续传下载，断点续传下载过程中有以下注意事项：

- 使用断点续传下载时，文件下载的进度信息会记录在Checkpoint文件中，如果下载过程中某一分片下载失败，再次下载时会从Checkpoint文件中记录的点继续下载，从而达到断点续传的效果。下载完成后，Checkpoint文件会被删除。
- SDK会将下载的状态信息记录在Checkpoint文件中，所以要确保程序对Checkpoint文件有写权限。
- 请勿修改Checkpoint文件中携带的校验信息。如果Checkpoint文件损坏，则会重新下载所有分片。
- 如果下载过程中文件的ETag发生变化、Part丢失或被修改，则重新下载文件。

使用阿里云SDK

以下仅列举常见SDK的断点续传下载的代码示例。关于其他SDK的断点续传下载的代码示例，请参见SDK简介。

C

```
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
public class Demo {
    public static void main(String[] args) {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，例如exampledir/exampleobject.txt。Object完整路径中不能包含Bucket名称。
        String objectName = "exampledir/exampleobject.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 请求10个任务并发下载。
            DownloadFileRequest downloadFileRequest = new DownloadFileRequest(bucketName, objectName);
            // 指定Object下载到本地文件的完整路径，例如D:\\localpath\\examplefile.txt。
            downloadFileRequest.setDownloadFile("D:\\localpath\\examplefile.txt");
            // 设置分片大小，单位为字节，取值范围为100 KB~5 GB。默认值为100 KB。
            downloadFileRequest.setPartSize(1 * 1024 * 1024);
            // 设置分片下载的并发数，默认值为1。
            downloadFileRequest.setTaskNum(10);
            // 开启断点续传下载，默认关闭。
            downloadFileRequest.setEnableCheckpoint(true);
            // 设置断点记录文件的完整路径，例如D:\\localpath\\examplefile.txt.dcp。
            // 只有当Object下载中断产生了断点记录文件后，如果需要继续下载该Object，才需要设置对应的断点记录文件。下载完成后，该文件会被删除。
            //downloadFileRequest.setCheckpointFile("D:\\localpath\\examplefile.txt.dcp");
            // 下载文件。
            DownloadFileResult downloadRes = ossClient.downloadFile(downloadFileRequest);
            // 下载成功时，会返回文件元信息。
            ObjectMetadata objectMetadata = downloadRes.getObjectMetadata();
            System.out.println(objectMetadata.getETag());
            System.out.println(objectMetadata.getLastModified());
            System.out.println(objectMetadata.getUserMetadata().get("meta"));
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (Throwable ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

更多参考

- 为了防止第三方未经授权从您的Bucket里下载数据，OSS提供了Bucket和Object级别的访问权限控制。更多信息，请参见访问控制概述。
- 如果您希望将私有Bucket的Object提供给第三方进行下载，请通过STS临时访问凭证或签名URL的方式授权第三方下载文件。具体操作，请参见授权给第三方下载。

6.4.3. 授权给第三方下载

本文介绍在不提供资源拥有者所属账号的访问密钥（AccessKey）的情况下，通过签名URL以及临时访问凭证的方式授权第三方下载文件（Object）。

临时访问凭证

OSS可以通过阿里云STS（Security Token Service）进行临时授权访问。阿里云STS是为云计算用户提供临时访问令牌的Web服务。通过STS，您可以为第三方应用或子用户（即用户身份由您自己管理的用户）颁发一个自定义时效和权限的访问凭证。关于STS的更多信息，请参见STS介绍。

STS的优势如下：

- 您无需透露您的长期密钥（AccessKey）给第三方应用，只需生成一个访问令牌并将令牌交给第三方应用。您可以自定义这个令牌的访问权限及有效期限。
- 您无需关心权限撤销问题，访问令牌过期后自动失效。

> **说明** 您可以通过调用STS服务的AssumeRole接口或者使用各语言STS SDK来获取临时访问凭证。临时访问凭证包括临时访问密钥（AccessKeyId和AccessKeySecret）和安全令牌（SecurityToken）。临时访问凭证有效时间单位为秒，最小值为900，最大值以当前角色设定的最大会话时间为准。更多信息，请参见设置角色最大会话时间。

使用阿里云SDK

以下仅列举常见SDK的通过STS临时授权第三方下载文件的代码示例。关于其他SDK的通过STS临时授权第三方下载文件的代码示例，请参见SDK简介。

```
// yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
String endpoint = "yourEndpoint";
// 从STS服务获取的临时访问密钥（AccessKey ID和AccessKey Secret）。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 从STS服务获取的安全令牌（SecurityToken）。
String securityToken = "yourSecurityToken";
// 填写Bucket名称，例如examplebucket。
String bucketName = "examplebucket";
// 填写Object完整路径，例如exampleobject.txt。Object完整路径中不能包含Bucket名称。
String objectName = "exampleobject.txt";
// 从STS服务获取临时访问凭证后，您可以通过临时访问密钥和安全令牌生成OSSClient。
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
// 通过STS临时授权将Object下载本地文件。如果指定的本地文件存在则覆盖，不存在则新建。
// 如果未指定本地路径，则下载后的文件默认保存到示例程序所属项目对应本地路径中。
ossClient.getObject(new GetObjectRequest(bucketName, objectName), new File("D:\\localpath\\examplefile.txt"));
// 关闭OSSClient。
ossClient.shutdown();
```

签名URL

注意 由于STS临时账号以及签名URL均需设置有效时长，当您使用STS临时账号生成签名URL执行相关操作（例如上传、下载文件）时，以最小的有效时长为准。例如您的STS临时账号的有效时长设置为1200秒、签名URL设置为3600秒时，当有效时长超过1200秒后，您无法使用此STS临时账号生成的签名URL上传文件。

您可以将生成的签名URL提供给访客进行临时访问。生成签名URL时，您可以通过指定URL的过期时间来限制访客的访问时长。签名URL的默认过期时间为3600秒，最大值为32400秒。

在URL中加入签名信息，以便将该URL转给第三方实现授权访问。更多信息，请参见在URL中包含签名。

注意 通过以下示例生成的签名URL中如果包含特殊符号 `+`，可能出现无法正常访问该签名URL的现象。如需正常访问该签名URL，请将签名URL中的 `+` 替换为 `%2B`。

使用阿里云SDK

以下仅列举常见SDK的通过签名URL授权第三方下载文件的代码示例。关于其他SDK的通过签名URL授权第三方下载文件的代码示例，请参见SDK简介。

```
import com.aliyun.oss.*;
import com.aliyun.oss.common.utils.DateUtil;
import com.aliyun.oss.model.GeneratePresignedUrlRequest;
import com.aliyun.oss.model.OSSObject;
import java.net.URL;
import java.util.*;
public class Demo {
    public static void main(String[] args) throws Throwable {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 从STS服务获取的安全令牌（SecurityToken）。
        String securityToken = "yourSecurityToken";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，例如exampleobject.txt。Object完整路径中不能包含Bucket名称。
        String objectName = "exampleobject.txt";
        // 从STS服务获取临时访问凭证后，您可以通过临时访问密钥和安全令牌生成OSSClient。
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
        try {
            // 填写签名URL的过期时间。
            Date expiration = null;
            expiration = DateUtil.parseRfc822Date("Wed, 18 Mar 2022 14:20:00 GMT");
            // 生成签名URL。
            GeneratePresignedUrlRequest request = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.GET);
            // 设置过期时间。
            request.setExpiration(expiration);
            // 通过HTTP GET请求生成签名URL。
            URL signedUrl = ossClient .generatePresignedUrl(request);
            System.out.println("signed url for getObject: " + signedUrl);
            // 使用签名URL发送请求。
            Map<String, String> customHeaders = new HashMap<String, String>();
            // 添加GetObject请求头。
            customHeaders.put("Range", "bytes=100-1000");
            OSSObject object = ossClient.getObject(signedUrl,customHeaders);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

6.5. 管理文件

6.5.1. 列举文件

存储空间（Bucket）内的文件（Object）默认按照字母序排列。您可以结合实际场景列举当前Bucket的所有Object、指定前缀的Object、指定个数的Object等。

使用OSS控制台

1. 登录OSS管理控制台。
2. 在左侧导航栏，单击**Bucket列表**。
3. 在**Bucket列表**页面，单击目标Bucket名称。
   当前页面将分页显示Bucket内的所有Object，默认每页显示50个Object。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成列举Object的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的简单列举Bucket内所有Object的代码示例。关于其他SDK的不同场景下列举符合指定条件Object的代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
import java.util.List;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 指定前缀，例如exampledir/object。
        String keyPrefix = "exampledir/object";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 列举文件。如果不设置keyPrefix，则列举存储空间下的所有文件。如果设置keyPrefix，则列举包含指定前缀的文件。
            ObjectListing objectListing = ossClient.listObjects(bucketName, keyPrefix);
            List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
            for (OSSObjectSummary s : sums) {
                System.out.println("\t" + s.getKey());
            }
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil列举Object的具体操作，请参见列举Object。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。

您可以通过API中的GetBucket (ListObjects)或GetBucketV2 (ListObjectsV2)接口列举Bucket中的Object。建议您在开发应用程序时使用较新的版本GetBucketV2 (ListObjectsV2)。为保证向后兼容性，OSS继续支持GetBucket (ListObjects)。

6.5.2. 拷贝文件

拷贝文件（Object）是指在不改变文件内容的情况下，将同一地域下的源存储空间（Bucket）内的文件复制到目标Bucket。

使用限制

- 不支持跨地域拷贝Object。例如，不支持将华东1（杭州）地域下Bucke内的Object拷贝到华东2（上海）地域下的Bucket。
- 不支持对通过追加上传方式生成的Object进行拷贝。

注意事项

- 您需要有源Object的读权限及目标Bucket的读写权限，否则无法完成拷贝操作。
- 拷贝文件时，您需要确保源Bucket和目标Bucket均未设置合规保留策略，否则报错 `The object you specified is immutable.` 。
- 拷贝文件时默认会覆盖同名文件，为防止文件被意外覆盖，您可以通过以下方式保护您的文件。
  - 开启版本控制功能

    开启版本控制功能后，被删除或覆盖的文件会以历史版本的形式保存下来。您可以随时恢复历史版本文件。更多信息，请参见版本控制介绍。
  - 在拷贝请求中携带禁止覆盖同名文件的参数

    在拷贝请求的Header中携带 x-oss-forbid-overwrite参数，并指定其值为*true*。当您拷贝的文件在目标Bucket中存在同名文件时，该文件将拷贝失败，并返回 `FileAlreadyExists` 错误。

使用图形化管理工具ossbrowser

通过ossbrowser仅支持拷贝小于5 GB的文件。关于如何使用ossbrowser拷贝文件的具体操作，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK通过CopyObject方法拷贝小于1 GB文件的代码示例。关于其他SDK的拷贝小于1 GB文件以及通过UploadPartCopy方法拷贝大于1 GB文件的代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写源Bucket名称。
        String sourceBucketName = "srcexamplebucket";
        // 填写源Object的完整路径。Object完整路径中不能包含Bucket名称。
        String sourceKey = "srcexampleobject.txt";
        // 填写与源Bucket处于同一地域的目标Bucket名称。
        String destinationBucketName = "desexamplebucket";
        // 填写目标Object的完整路径。Object完整路径中不能包含Bucket名称。
        String destinationKey = "desexampleobject.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 创建CopyObjectRequest对象。
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(sourceBucketName, sourceKey, destinationBucketName, destinationKey);
            // 设置新的文件元信息。
            ObjectMetadata meta = new ObjectMetadata();
            meta.setContentType("text/txt");
            // 指定CopyObject操作时是否覆盖同名目标Object。此处设置为true，表示禁止覆盖同名Object。
            // meta.setHeader("x-oss-forbid-overwrite", "true");
            // 指定拷贝的源地址。
            // meta.setHeader(OSSHeaders.COPY_OBJECT_SOURCE, "/examplebucket/recode-test.txt");
            // 如果源Object的ETag值和您提供的ETag相等，则执行拷贝操作，并返回200 OK。
            // meta.setHeader(OSSHeaders.COPY_OBJECT_SOURCE_IF_MATCH, "5B3C1A2E053D763E1B002CC607C5****");
            // 如果源Object的ETag值和您提供的ETag不相等，则执行拷贝操作，并返回200 OK。
            // meta.setHeader(OSSHeaders.COPY_OBJECT_SOURCE_IF_NONE_MATCH, "5B3C1A2E053D763E1B002CC607C5****");
            // 如果指定的时间等于或者晚于文件实际修改时间，则正常拷贝文件，并返回200 OK。
            // meta.setHeader(OSSHeaders.COPY_OBJECT_SOURCE_IF_UNMODIFIED_SINCE, "2021-12-09T07:01:56.000Z");
            // 如果源Object在指定时间后被修改过，则执行拷贝操作。
            // meta.setHeader(OSSHeaders.COPY_OBJECT_SOURCE_IF_MODIFIED_SINCE, "2021-12-09T07:01:56.000Z");
            // 指定设置目标Object元信息的方式。此处设置为COPY，表示复制源Object的元数据到目标Object。
            // meta.setHeader(OSSHeaders.COPY_OBJECT_METADATA_DIRECTIVE, "COPY");
            // 指定OSS创建目标Object时使用的服务器端加密算法。
            // meta.setHeader(OSSHeaders.OSS_SERVER_SIDE_ENCRYPTION, ObjectMetadata.KMS_SERVER_SIDE_ENCRYPTION);
            // 表示KMS托管的用户主密钥，该参数仅在x-oss-server-side-encryption为KMS时有效。
            // meta.setHeader(OSSHeaders.OSS_SERVER_SIDE_ENCRYPTION_KEY_ID, "9468da86-3509-4f8d-a61e-6eab1eac****");
            // 指定OSS创建目标Object时的访问权限，此处设置为Private，表示只有Object的拥有者和授权用户有该Object的读写权限，其他用户没有权限操作该Object。
            // meta.setHeader(OSSHeaders.OSS_OBJECT_ACL, CannedAccessControlList.Private);
            // 指定Object的存储类型。此处设置为Standard，表示标准存储类型。
            // meta.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard);
            // 指定Object的对象标签，可同时设置多个标签。
            // meta.setHeader(OSSHeaders.OSS_TAGGING, "a:1");
            // 指定设置目标Object对象标签的方式。此处设置为COPY，表示复制源Object的对象标签到目标Object。
            // meta.setHeader(OSSHeaders.COPY_OBJECT_TAGGING_DIRECTIVE, "COPY");
            copyObjectRequest.setNewObjectMetadata(meta);
            // 复制文件。
            CopyObjectResult result = ossClient.copyObject(copyObjectRequest);
            System.out.println("ETag: " + result.getETag() + " LastModified: " + result.getLastModified());
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil拷贝文件的具体操作， 请参见拷贝文件。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见CopyObject。

6.5.3. 解冻文件

归档或冷归档类型的Object需要解冻（Restore）之后才能读取。本文介绍如何解冻归档和冷归档Object。

解冻过程说明

对于归档类型或者冷归档类型的Object，如果需要读取Object，请提前解冻。归档类型的Object解冻有分钟级延迟，冷归档类型的Object解冻有数小时延迟。

归档类型或者冷归档类型的Object在执行解冻前后的状态变换过程如下：

1. Object初始时处于冷冻状态。
2. 提交一次解冻请求后，Object处于解冻中状态。
3. 服务端完成解冻任务后，Object进入解冻状态，此时您可以读取Object。
   - 归档类型Object

     对于归档类型的Object，解冻状态默认持续24小时，24小时内再次调用RestoreObject接口则解冻状态会自动延长24小时，一次解冻流程内可有效调用7次RestoreObject接口达到最长7天的解冻持续时间。您也可以通过传入解冻天数，一次调用RestoreObject接口指定最长7天的解冻持续时间。
   - 冷归档类型Object

     对于冷归档类型的Object，您可以指定解冻天数和解冻优先级，解冻天数最短为1天，最长为365天。不同解冻优先级的首字节取回时间如下：
     - 高优先级（Expedited）：表示1小时内完成解冻。
     - 标准（Standard）：表示2~5小时内完成解冻。如果不传入JobParameters节点，则默认为Standard。
     - 批量（Bulk）：表示5~12小时内完成解冻。
4. 解冻状态结束后，Object再次返回到冷冻状态。

注意事项

- RestoreObject接口只针对归档或冷归档类型的Object，不适用于标准类型和低频访问类型的Object。
- 如果针对该Object第一次调用RestoreObject接口，则返回202。
- 如果针对该Object第一次调用RestoreObject接口，则返回202。如果已经成功调用过RestoreObject接口，且Object已完成解冻，再次调用时返回200 OK。
- 在开启版本控制的Bucket中，Object的各个版本可以对应不同的存储类型。调用RestoreObject接口默认解冻Object当前版本，您可以通过指定versionId的方式来解冻Object指定版本。

计费说明

- 解冻归档存储和冷归档存储类型文件会产生数据取回费用。更多信息，请参见数据处理费用。
- 归档类型Object可达到最长7天的解冻持续时间，冷归档类型Object可达到最长365天的解冻持续时间，在此期间不再重复收取数据取回费用。
- 解冻状态结束后，Object又回到冷冻状态，再次执行解冻操作会收取数据取回费用。
- 冷归档存储类型文件在数据解冻时会生成一份标准存储类型的文件副本用于访问。该文件在解冻时间结束前会以标准存储的存储费率计算临时存储费用。更多信息，请参见临时存储费用。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**文件管理 > 文件管理**。
4. 在待解冻的目标文件右侧的操作栏下，选择 ⋮ **> 解冻**。
   - 归档存储类型Object
     - 解冻需要约1分钟时间，解冻成功后，Object变为解冻状态。
     - 解冻状态默认持续1天，您可以通过ossutil工具或SDK延长解冻时间，最多延长7天，之后文件又回到冷冻状态。
   - 冷归档存储类型Object

     解冻冷归档存储类型Object时，您需要通过**副本有效期**设置Object处于解冻状态的持续时间，取值为1~7，单位为天。 此外，您还可以通过**恢复模式**设置解冻优先级。

     根据解冻文件的大小，实际解冻时间可能会有变化，请以实际解冻时间为准。

使用图形化管理工具ossbrowser

ossbrowser支持Object级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成解冻文件的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的解冻文件的代码示例。关于其他SDK的解冻文件的代码示例，请参见SDK简介。

Android Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写不包含Bucket名称在内的归档类型Object的完整路径。
        String objectName = "exampledir/object";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            ObjectMetadata objectMetadata = ossClient.getObjectMetadata(bucketName, objectName);
            // 校验Object是否为归档类型Object。
            StorageClass storageClass = objectMetadata.getObjectStorageClass();
            if (storageClass == StorageClass.Archive) {
                // 解冻Object。
                ossClient.restoreObject(bucketName, objectName);
                // 等待解冻完成。
                do {
                    Thread.sleep(1000);
                    objectMetadata = ossClient.getObjectMetadata(bucketName, objectName);
                } while (!objectMetadata.isRestoreCompleted());
            }
            // 获取解冻Object。
            OSSObject ossObject = ossClient.getObject(bucketName, objectName);
            ossObject.getObjectContent().close();
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil解冻归档和冷归档Object的具体操作， 请参见restore（解冻文件）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见RestoreObject。

更多参考

当您对归档或冷归档类型Object提交了RestoreObject请求，并且希望了解Object的解冻操作是否完成时，您可以通过调用GetBucket（ListObjects）、GetBucketV2（ListObjectsV2）或者GetBucketVersions(ListObjectVersions)进行查看。

6.5.4. 删除文件

您可以通过多种方式删除存储空间（Bucket）中不再需要保留的文件（Object）。

> **警告** 文件删除后无法恢复，请谨慎操作。

删除规则

OSS支持手动或者自动删除单个或多个文件，删除规则说明如下：

- 手动删除
  - 删除单个文件

    您可以通过OSS控制台、命令行工具ossutil、SDK、图形化管理工具ossbrowser的方式删除单个文件。
  - 删除多个文件

    通过OSS控制台一次最多可删除100个文件，通过SDK一次最多可删除1000个文件，通过命令行工具ossutil以及图形化管理工具ossbrowser一次最多可删除的文件个数无限制。

- 自动删除

  如果需要删除的文件数目较多，且删除的文件有一定的规律，例如需要定期删除指定日期之前的文件，指定前缀的文件，又或者需要清空整个Bucket内的所有文件。此时，推荐您配置生命周期规则自动删除文件。生命周期规则配置完成后，OSS会根据规则自动删除指定文件，从而减少您发送删除请求的次数，以提高删除效率。更多信息，请参见生命周期规则介绍。

使用OSS控制台

1. 登录OSS管理控制台。
2. 在左侧导航栏，选择**文件管理 > 文件管理**。
3. 选中一个或多个文件，选择**批量操作彻底删除**。
4. 在弹出的对话框中，单击**确定**。

使用图形化管理工具ossbrowser

ossbrowser支持Object级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成删除文件的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的删除单个文件的代码示例。关于其他SDK删除单个文件以及多个文件的代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写文件完整路径。文件完整路径中不能包含Bucket名称。
        String objectName = "exampleobject.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 删除文件或目录。如果要删除目录，目录必须为空。
            ossClient.deleteObject(bucketName, objectName);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil删除文件的具体操作， 请参见删除Object。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。

关于删除单个文件API接口说明，请参见DeleteObject。

关于删除多个文件的API接口说明，请参见DeleteMultipleObjects。

6.5.5. 对象标签

OSS支持使用标签对存储空间（Bucket）中的对象（Object）进行分类，您可以针对同标签的Object设置生命周期规则、访问权限等。

标签规则

Object标签使用一组键值对（Key-Value）标记Object，您可以在上传Object时添加标签，也可以为已有Object添加标签。

- 单个Object最多可设置10个标签，Key不可重复。
- 每个Key长度不超过128字符，每个Value长度不超过256字符。
- Key和Value区分大小写。

- 标签合法字符集包括大小写字母、数字、空格和以下符号：

  +-=._:/

  通过HTTP Header的方式设置标签且标签中包含任意字符时，您需要对标签的Key和Value进行URL编码。

注意事项

- 只有Bucket拥有者以及被授予 `oss:PutObjectTagging` 权限的用户拥有读写Object标签的权限。
- 您可以在简单上传、分片上传、追加上传以及拷贝文件过程中为Object设置标签，也可以对已上传Object设置标签。
- Object添加标签后，OSS会按照每小时内统计的标签数量收取标签费用。更多信息，请参见对象标签费用。
- 更改标签信息不会更新Object的Last-Modified时间。
- 跨区域复制时，源Object标签也会复制到目标Bucket。

使用场景

- Object标签结合生命周期

  对于周期性生成且无需长期保存的Object，可以在上传时设置指定的标签，之后通过生命周期规则，将拥有该标签的文件定期删除，从而节省存储费用。例如，通过生命周期规则指定前缀为dir1且拥有Key为key1、Value为value1标签的Object在距离最后一次更新时间30后删除，配置示例如下：

```
<LifecycleConfiguration>
 <Rule>
  <ID>rule1</ID>
  <Prefix>dir1</Prefix>
  <Tag><Key>key1</Key><Value>value1</Value></Tag>
  <Status>Enabled</Status>
  <Expiration>
    <Days>30</Days>
  </Expiration>
 </Rule>
</LifecycleConfiguration>
```

- 授权RAM用户访问指定标签的Object

  例如，您可以通过RAM Policy授权RAM用户访问Key为key2、Value为value2的所有Object，RAM Policy配置示例如下。

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "oss:GetObject",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "oss:ExistingObjectTag/key2": [
            "value2"
          ]
        }
      }
    }
  ]
}
```

  您还可以授予该用户其他操作权限（Action），例如向拥有指定标签的Object写入数据、查看Object相关信息等。关于RAM Policy支持的Action信息，请参见RAM Policy概述。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 单击**文件管理**页签。
4. 为Object设置标签。
   i. 选择需要设置标签的Object。
      - Bucket未开启版本控制

        在目标Object右侧操作栏下，选择**更多 > 标签**。
      - Bucket已开启版本控制

        在指定版本Object右侧操作栏下，选择**更多 > 标签**。

   ii. 在**标签**面板，按标签使用规则说明指定标签的**键**和**值**。
5. 单击**确定**。

使用阿里云SDK

以下仅列举常见SDK在简单上传过程中设置Object标签的代码示例。关于其他SDK在简单上传、分片上传、追加上传、拷贝文件过程中设置Object标签的代码示例，请参见SDK简介。

Node.js

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.*;
import java.io.ByteArrayInputStream;
import java.util.HashMap;
import java.util.Map;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，Object完整路径中不能包含Bucket名称。例如exampledir/exampleobject.txt。
        String objectName = "exampledir/exampleobject.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            Map<String, String> tags = new HashMap<String, String>();
            // 依次填写对象标签的键（例如owner）和值（例如John）。
            tags.put("owner", "John");
            tags.put("type", "document");
            // 在HTTP header中设置标签信息。
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setObjectTagging(tags);
            // 上传文件的同时设置标签信息。
            String content = "<yourtContent>";
            ossClient.putObject(bucketName, objectName, new ByteArrayInputStream(content.getBytes()), metadata);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置Object标签的具体操作，请参见添加或修改Object标签。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutObjectTagging。

6.5.6. 管理文件元信息

文件元信息是对文件的属性描述，包括HTTP标准属性（HTTP Header）和用户自定义元数据（User Meta）两种。您可以通过设置文件HTTP头来自定义HTTP请求的策略，例如文件（Object）缓存策略、强制下载策略等。您还可以通过设置用户自定义元数据来标识Object的用途或属性等。

HTTP标准属性

OSS将为上传至Bucket中的每个Object保留如下HTTP标准属性。

名称	描述
Content-Type	指定Object的文件类型。内容类型决定浏览器将以什么形式、什么编码读取文件。如果没有指定文件类型，则根据文件的扩展名生成。如果文件没有扩展名，则文件类型的默认值 `application/octet-stream` 。Content-Type的常见设置请参见如何设置Content-Type（MIME）？。

名称	描述
Content-Encoding	声明Object的编码方式。您需要按照Object 的实际编码类型填写，否则可能造成客户端（浏览器）解析编码失败或Object下载失败。若Object未编码，请置空此项。取值如下： • *identity*（默认值）：表示Object未经过压缩或编码。 • *gzip*：表示Object采用Lempel-Ziv（LZ77）压缩算法以及32位CRC校验的编码方式。 • *compress*：表示Object采用Lempel-Ziv-Welch（LZW）压缩算法的编码方式。 • *deflate*：表示Object采用zlib结构和deflate压缩算法的编码方式。 • *br*：表示Object采用Brotli算法的编码方式。 关于Content-Encoding的更多信息，请参见RFC2616。 **注意** 如果您希望访问OSS内常见网页静态文件（HTML、Javascript、XML、json）时进行Gzip压缩，您需要置空此项，并在请求中增加 `Accept-Encoding: gzip` 。
Content-Language	声明Object内容使用的语言。例如某个Object使用简体中文编写，则此项可设置为*zh-CN*。
Content-Disposition	指定Object的展示形式。取值如下： • *inline*：预览Object。 **注意** 在以下情况中通过浏览器访问OSS内的Object，即使Content-Disposition取值为inline，也会直接下载Object： ○ Object为网页文件，未使用Bucket绑定的自定义域名进行访问。 ○ Object为图片文件，Object所在的Bucket于2019年09月23日之后创建，且未使用Bucket绑定的自定义域名进行访问。 ○ 请求的Object为浏览器不支持预览的类型。 • *attachment*：将Object下载到本地。例如 `attachment; filename="example.jpg"` ，表示下载Object到本地并以 `example.jpg` 文件名进行保存。 **说明** 通过浏览器将Object下载到本地时，如果Object名称包含星号（*）、正斜线（/）等特殊字符时，可能会出现特殊字符转义的情况。例如，下载 `example*.jpg"` 到本地时， `example*.jpg"` 可能会转义为 `example_.jpg"` 。 有关Content-Disposition的更多信息，请参见RFC2616。
Cache-Control	指定Object的缓存行为。取值如下： • *no-cache*：不可直接使用缓存，而是先到服务端验证Object是否已更新。如果Object已更新，表明缓存已过期，需从服务端重新下载Object；如果Object未更新，表明缓存未过期，此时将使用本地缓存。 • *no-store*：所有内容都不会被缓存。 • *public*：所有内容都将被缓存。 • *private*：所有内容只在客户端缓存。 • *max-age=<seconds>*：缓存内容的相对过期时间，单位为秒。此选项仅在HTTP 1.1中可用。
Expires	缓存内容的绝对过期时间，格式是格林威治时间（GMT）。例如 `2022-10-12T00:00:00.000Z` 。如果Cache-Control设置了 `max-age=<seconds>` ，以 `max-age=<seconds>` 为准。
Last-Modified	Object的最后修改时间。
Content-Length	Object的大小，单位为字节。

用户自定义元数据

您可以在上传Object时，为Object添加自定义元数据（User Meta），用于标识Object的用途或属性等。

- 一个Object可以有多个自定义元数据，但所有的自定义元数据总大小不能超过8 KB。
- 自定义元数据是一组键值对，元数据名称必须以 `x-oss-meta-` 开头。例如 `x-oss-meta-last-modified:20210506` ，可用于记录本地文件最后修改时间为2021年5月6日。
- 调用GetObject或者HeadObject接口时，将在HTTP头部返回自定义元数据。

使用OSS控制台

通过OSS管理控制台一次可批量设置100个Object的文件元信息。

1. 登录OSS管理控制台。
2. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket。
3. 在左侧导航栏，选择**文件管理 > 文件管理**。
4. 通过以下任意方式打开**设置HTTP头**面板。
   - 设置批量Object的HTTP头

     选中一个或多个Object，然后选择**批量操作 > 设置HTTP头**。
   - 设置单个Object的HTTP头

     在目标Object右侧选择**更多 > 设置HTTP头**。
5. 在**设置HTTP头**页面，设置HTTP标准属性以及用户自定义元数据。
6. 单击**确定**。

使用图形化管理工具ossbrowser

ossbrowser支持Object级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成设置Object元信息的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的设置Object元信息的代码示例。关于其他SDK的设置Object元信息的代码示例，请参见SDK简介。

Java

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.common.utils.BinaryUtil;
import com.aliyun.oss.common.utils.DateUtil;
import com.aliyun.oss.model.ObjectMetadata;
import java.io.ByteArrayInputStream;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写不包含Bucket名称在内的Object完整路径，例如testfolder/exampleobject.txt。
        String objectName = "testfolder/exampleobject.txt";
        String content = "Hello OSS";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 创建上传文件的元信息，可以通过文件元信息设置HTTP header。
            ObjectMetadata meta = new ObjectMetadata();
            String md5 = BinaryUtil.toBase64String(BinaryUtil.calculateMd5(content.getBytes()));
            // 开启文件内容MD5校验。开启后OSS会把您提供的MD5与文件的MD5比较，不一致则抛出异常。
            meta.setContentMD5(md5);
            // 指定上传的内容类型。内容类型决定浏览器将以什么形式、什么编码读取文件。如果没有指定则根据文件的扩展名生成，如果没有扩展名则为默认值application/octet-str
eam。
            meta.setContentType("text/plain");
            // 设置内容被下载时的名称。
            meta.setContentDisposition("attachment; filename=\"DownloadFilename\"");
            // 设置上传文件的长度。如超过此长度，则上传文件会被截断，上传的文件长度为设置的长度。如小于此长度，则为上传文件的实际长度。
            meta.setContentLength(content.length());
            // 设置内容被下载时网页的缓存行为。
            meta.setCacheControl("Download Action");
            // 设置缓存过期时间，格式是格林威治时间（GMT）。
            meta.setExpirationTime(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
            // 设置内容被下载时的编码格式。
            meta.setContentEncoding("utf-8");
            // 设置Header。
            meta.setHeader("yourHeader", "yourHeaderValue");
            // 上传文件。
            ossClient.putObject(bucketName, objectName, new ByteArrayInputStream(content.getBytes()), meta);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置Object元信息的具体操作，请参见set-meta（管理文件元信息）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutObject。

6.5.7. 通过目录管理文件

与传统文件系统中的层级结构不同，OSS内部使用扁平结构存储数据。即所有数据均以对象（Object）的形式保存在存储空间（Bucket）中。为方便您对Object进行分组并简化权限管理，您可以创建目录，然后将目标文件存放至指定目录。

目录结构

OSS将以正斜线（/）结尾的Object视为目录。例如目标存储空间examplebucket下的目录及文件结构如下图所示：

```
examplebucket
    └── log/
       ├── date1.txt
       ├── date2.txt
       ├── date3.txt
    └── destfolder/
       └── 2021/
          ├── photo.jpg
```

以上目录结构示意图表明：

- 以log为前缀的文件共有三个，分别为*log/date1.txt*、*log/date2.txt*和*log/date3.txt*。控制台会显示名为log的目录，如果您在控制台中打开该目录，将看到三个文件，分别为*date1.txt*、*date2.txt*和*date3.txt*。
- 以destfolder为前缀的文件为*destfolder/2021/photo.jpg*。控制台将显示名为destfolder的目录，其中包含子目录2021，子目录下包含文件*photo.jpg*。

目录授权

例如，您希望授予第三方用户对上述示例的目标存储空间examplebucket下不同目录或者文件有不同的访问权限，您可以通过以下方式来实现：

- log目录的三个文件*log/date1.txt*、*log/date2.txt*和*log/date3.txt*，分别用于存储某用户近三天访问OSS的日志。经发现近三天出现了访问速度下降、上传文件失败的情况，该用户希望相关技术支持人员可以查看log目录下的所有文件，以协助排查并解决问题。此时，您可以通过Bucket Policy授权用户访问指定资源。具体步骤，请参见配置Bucket Policy。
- *destfolder/2021/photo.jpg*文件为公司全员2021年外出春游合照，希望公司全员都可以查看。此时，您可以将文件读写权限ACL设置为公共读。具体步骤，请参见Object ACL。

创建目录

您可以通过以下多种方式创建目录。目录创建完成后，您可以将文件上传到指定目录。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**文件管理 > 文件管理**。
4. 在**文件管理**页面，单击**新建目录**。
5. 在**新建目录**对话框，输入**目录名**。

   目录命名规范如下：

   - 不允许使用表情符，请使用符合要求的UTF-8字符。
   - 正斜线 `/` 用于分割路径，可快速创建子目录，但不要以正斜线 `/` 或 反斜线 `\` 开头，不要出现连续的正斜线 `/` 。
   - 不允许出现名为 `..` 的子目录。
   - 总长度控制在1~254个字符。
6. 单击**确定**。

使用图形化管理工具ossbrowser

ossbrowser支持Object级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成创建目录的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK创建目录的代码示例。关于其他SDK的创建目录的代码示例，请参见SDK简介。

您可以通过将保存到OSS的文件名称ObjectName或Key设置为以正斜线（/）分隔的方式来创建目录。例如，当您将本地文件*localfile.txt*上传至目标存储空间examplebucket时，如果ObjectName设置为*destfolder/localfile.txt*，则examplebucket下将创建大小为0的目录destfolder，目录下包含了文件*localfile.txt*。

```
Python
```

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.PutObjectRequest;
import java.io.File;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称。
        String bucketName = "examplebucket";
        // 填写Object完整路径，完整路径中不能包含Bucket名称。
        String objectName = "destfolder/localfile.txt";
        // 填写本地文件的完整路径。如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
        String filePath= "D:\\localpath\\localfile.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 创建PutObjectRequest对象。
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, objectName, filePath);
            // 如果需要上传时设置存储类型和访问权限，请参考以下示例代码。
            // ObjectMetadata metadata = new ObjectMetadata();
            // metadata.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard.toString());
            // metadata.setObjectAcl(CannedAccessControlList.Private);
            // putObjectRequest.setMetadata(metadata);
            // 上传文件。
            ossClient.putObject(putObjectRequest);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil创建目录的具体操作，请参见mkdir（创建目录）。

（可选）删除目录

当您不希望保留该目录时，也可以通过以下多种方式删除目录。

使用OSS控制台

1. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket名称。
2. 在左侧导航栏，选择**文件管理 > 文件管理**。
3. 在文件管理页签，根据您的需求删除指定目录。

> **注意** 删除目录及文件期间，请勿刷新或关闭任务列表，否则会导致任务中断。

- 在已开启分层命名空间的Bucket中删除

  单击目标目录右侧的**彻底删除**，然后在弹出的对话框单击**确认**。

  此时，目标目录及其包含的文件会被彻底删除。

- 在未开启分层命名空间的Bucket中删除

  在Bucket未开启分层命名空间，且未开启版本控制的情况下，目录及文件的删除行为与已开启分层命名空间的Bucket中的删除行为一致。

  在Bucket未开启分层命名空间，且已开启版本控制的情况下，对目录及文件进行删除时，会有以下两种情况：

  - 将目录转为历史版本
    a. 在文件列表右上角，将**历史版本**设置为**隐藏**。
    b. 单击目标目录右侧的**删除**，然后在弹出的对话框单击**确认**。

    此时，删除的目录及文件会被转为历史版本，您可以在需要时对目录和文件进行恢复。恢复历史版本文件的具体操作，请参见恢复历史版本Object。

- 将目录彻底删除
  a. 在文件列表右上角，将**历史版本**设置为**显示**。
  b. 单击目标目录右侧的**彻底删除**，然后在弹出的对话框单击**确认**。
  此时，目录及其目录下的文件会被彻底删除。

4. 在弹出的**任务列表**面板查看删除进度。
   删除任务进行期间，您可以进行以下操作：
   - **移除已完成**：单击可移除列表中已完成的删除任务。
   - **全部暂停**：单击可暂停正在进行中的删除任务。任务暂停期间，您可以进行以下操作：
     - 单击目标任务右侧的**开始**，可以重新开始任务。
     - 单击目标任务右侧的**移除**，可以移除该任务。任务移除后，未删除的文件将继续保留。
   - **全部开始**：单击可开始全部暂停中的删除任务。

使用图形化管理工具ossbrowser

ossbrowser支持Object级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成删除目录的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的删除目录的代码示例。关于其他SDK的删除目录的代码示例，请参见SDK简介。

您可以通过指定Prefix的方式删除目录及目录下的所有文件。例如，您希望删除存储空间examplebucket下目录log及该目录下的所有文件，请将示例代码中的Prefix参数指定为*log/*。

```
// yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
String endpoint = "yourEndpoint";
// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 填写Bucket名称。
String bucketName = "examplebucket";
// 填写待删除目录的完整路径，完整路径中不包含Bucket名称。
final String prefix = "log/";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 删除目录及目录下的所有文件。
String nextMarker = null;
ObjectListing objectListing = null;
do {
    ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName)
            .withPrefix(prefix)
            .withMarker(nextMarker);
    objectListing = ossClient.listObjects(listObjectsRequest);
    if (objectListing.getObjectSummaries().size() > 0) {
        List<String> keys = new ArrayList<String>();
        for (OSSObjectSummary s : objectListing.getObjectSummaries()) {
            System.out.println("key name: " + s.getKey());
            keys.add(s.getKey());
        }
        DeleteObjectsRequest deleteObjectsRequest = new DeleteObjectsRequest(bucketName).withKeys(keys).withEncodingType("url");
        DeleteObjectsResult deleteObjectsResult = ossClient.deleteObjects(deleteObjectsRequest);
        List<String> deletedObjects = deleteObjectsResult.getDeletedObjects();
        try {
            for(String obj : deletedObjects) {
                String deleteObj =  URLDecoder.decode(obj, "UTF-8");
                System.out.println(deleteObj);
            }
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
    nextMarker = objectListing.getNextMarker();
} while (objectListing.isTruncated());
// 关闭OSSClient。
ossClient.shutdown();
```

使用命令行工具ossutil

您可以在ossutil的rm命令示例中通过*prefix*选项指定待删除目录名称的方式删除指定目录。具体操作，请参见删除Object。

6.5.8. 单链接限速

客户端访问OSS内的文件（Object）时会占用较大带宽，在某些不容易控制流控的客户端上可能会对其他应用造成影响。为避免此类问题，您可以通过OSS提供的单链接限速功能在上传、下载文件等操作中进行流量控制，以保证其他应用的网络带宽。

注意事项

您可以在PutObject、AppendObject、PostObject、CopyObject、UploadPart、UploadPartCopy、GetObject请求中增加x-oss-traffic-limit参数，并指定限速值。限速值取值范围为819200~838860800，单位为bit/s。

客户端发起请求时限速

通过客户端发起请求时，仅支持通过SDK对其进行限速。以下仅列举常见SDK在客户端发起简单上传或下载请求时进行限速的代码示例。关于其他SDK在客户端发起上传或下载请求时进行限速的代码示例，请参见SDK简介。

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.GetObjectRequest;
import com.aliyun.oss.model.PutObjectRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
public class Demo {
    public static void main(String[] args) throws Throwable {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径，完整路径中不能包含Bucket名称，例如exampledir/exampleobject.txt。
        String objectName = "exampledir/exampleobject.txt";
        // 填写待上传的本地文件的完整路径，例如D:\\localpath\\examplefile.txt。
        // 如果未指定本地路径，则默认从示例程序所属项目对应本地路径中上传文件。
        String localFileName = "D:\\localpath\\examplefile.txt";
        // 填写Object下载到本地文件的完整路径。如果指定的本地文件存在会覆盖，不存在则新建。
        // 如果未指定本地路径，则下载后的文件默认保存到示例程序所属项目对应本地路径中。
        String downLoadFileName = "D:\\localpath\\exampleobject.txt";
        // 限速100 KB/s。
        int limitSpeed = 100 * 1024 * 8;
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 限速上传。
            InputStream inputStream = new FileInputStream(localFileName);
            PutObjectRequest PutObjectRequest = new PutObjectRequest(bucketName, objectName, inputStream);
            PutObjectRequest.setTrafficLimit(limitSpeed);
            ossClient.putObject(PutObjectRequest);
            // 限速下载。
            GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName, objectName);
            getObjectRequest.setTrafficLimit(limitSpeed);
            File localFile = new File(downLoadFileName);
            ossClient.getObject(getObjectRequest, localFile);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

通过文件URL限速

对于公共读或公共读写文件，您仅需要在分享的文件URL后加入限速参数 `x-oss-traffic-limit=<value>` 即可实现限速访问。例如， `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/video.mp4?x-oss-traffic-limit=819200` 表示下载*video.mp4*文件时限速为100 KB/s。限速效果如下图所示：

通过签名URL限速

对于私有文件，您需要通过SDK在生成签名URL时，将限速参数加入签名中一起计算。以下仅列举常见SDK在签名URL中加入限速参数的代码示例。关于其他SDK的在签名URL中加入限速参数的代码示例，请参见SDK简介。

```
Failed to resolve content from t1280319.dita#concept_1614716/codeblock_ukn_qy3_342
```

6.5.9. 使用SelectObject查询文件

您可以使用SelectObject用目标文件执行SQL语句，返回执行结果。

背景信息

目前Hadoop 3.0已经支持OSS在EMR上运行Spark、Hive、Presto等服务，同时阿里云MaxCompute以及Data Lake Analytics均支持从OSS直接处理数据。

OSS提供的GetObject接口决定了大数据平台只能把OSS数据全部下载到本地然后进行分析过滤，在很多查询场景下浪费了大量带宽和客户端资源。

SelectObject接口是对上述问题的解决方案。其核心思想是大数据平台将条件、Projection下推到OSS层，让OSS做基本的过滤，从而只返回有用的数据。客户端一方面可以减少网络带宽，另一方面也减少了数据的处理量，从而节省了CPU和内存用来做其他更多的事情。这使得基于OSS的数据仓库、数据分析成为一种更有吸引力的选择。

支持的文件类型

以下内容是对SelectObject支持的文件类型、支持的SQL语法等的详细介绍。

- RFC 4180标准的CSV（包括TSV等类CSV文件，文件的行列分隔符以及Quote字符都可自定义）。
- JSON文件，且文件编码为UTF-8。JSON支持DOCUMENT和LINES两种文件。
  - DOCUMENT是指整个文件是单一的JSON对象。
  - LINES表示整个文件由一行行的JSON对象组成，每一行是一个JSON对象（但整个文件本身并不是一个合法的JSON对象），行与行之间以换行分隔符隔开。OSS Select可以支持常见的\n，\r\n等分隔符，且无需用户指定。
- 标准存储类型和低频访问存储类型的文件。归档存储和冷归档存储类型文件需要先执行解冻操作。
- OSS完全托管加密、KMS托管主密钥加密的文件。

支持的SQL语法

- SQL语句： Select From Where
- 数据类型：string、int（64bit）、double（64bit）, decimal（128bit）、timestamp、bool
- 操作： 逻辑条件（AND,OR,NOT）， 算术表达式（+-*/%）， 比较操作（>,=, <, >=, <=, !=），String 操作 (LIKE, ||)

> **注意** LIKE模糊匹配时对字母大小写敏感。

支持的数据类型

OSS中的CSV数据默认都是String类型，您可以使用CAST函数实现数据转换。

通过SQL查询语句将_1和_2转换为int的示例： `Select * from OSSOBject where cast (_1 as int) > cast(_2 as int)`

同时，对于SelectObject支持在Where条件中进行隐式转换，例如下面语句中的第一列和第二列将被转换成int：

```
Select _1 from ossobject where _1 + _2 > 100
```

对于JSON文件，如果在SQL中未指定cast函数，则其类型根据JSON数据的实际类型而定，标准JSON内建的数据类型包括null、bool、int64、double、string等类型。

常见的SQL用例

常见的SQL用例包括CSV及JSON两种。

- CSV

应用场景	SQL语句
返回前10行数据	select * from ossobject limit 10
返回第1列和第3列的整数，并且第1列大于第3列	select _1, _3 from ossobject where cast(_1 as int) > cast(_3 as int)
返回第1列以'陈'开头的记录的个数（注：此处like后的中文需要用UTF-8编码）	select count(*) from ossobject where _1 like '陈%'
返回所有第2列时间大于2018-08-09 11:30:25且第3列大于200的记录	select * from ossobject where _2 > cast('2018-08-09 11:30:25' as timestamp) and _3 > 200
返回第2列浮点数的平均值，总和，最大值，最小值	select AVG(cast(_2 as double)), SUM(cast(_2 as double)), MAX(cast(_2 as double)), MIN(cast(_2 as double))
返回第1列和第3列连接的字符串中以'Tom'为开头以'Anderson'结尾的所有记录	select * from ossobject where (_1 || _3) like 'Tom%Anderson'
返回第1列能被3整除的所有记录	select * from ossobject where (_1 % 3) = 0
返回第1列大小在1995到2012之间的所有记录	select * from ossobject where _1 between 1995 and 2012
返回第5列值为N,M,G,L的所有记录	select * from ossobject where _5 in ('N', 'M', 'G', 'L')
返回第2列乘以第3列比第5列大100以上的所有记录	select * from ossobject where _2 * _3 > _5 + 100

- JSON

  假设JSON文件如下：

```
{
  "contacts":[
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}, ..... #此处省略其他类似的节点
]}
```

SQL用例如下：

应用场景	SQL语句
返回所有age是27的记录	select * from ossobject.contacts[*] s where s.age = 27
返回所有的家庭电话	select s.number from ossobject.contacts[*].phoneNumbers[*] s where s.type = “home”
返回所有单身的记录	select * from ossobject s where s.spouse is null
返回所有没有孩子的记录	select * from ossobject s where s.children[0] is null 说明 目前没有专用的空数组的表示方法，用以上语句代替。

使用场景

SelectObject通常用于大文件分片查询、JSON文件查询、日志文件分析等场景。

- 大文件分片查询

  和GetObject提供的基于Byte的分片下载类似，SelectObject也提供了分片查询的机制，包括以下两种分片方式：

  - 按行分片：常用的分片方式，然而对于稀疏数据来说，按行分片可能会导致分片时负载不均衡。
  - 按Split分片：Split是OSS用于分片的一个概念，一个Split包含多行数据，每个Split的数据大小大致相等。

  > **说明** 按Spit分片比按行分片更加高效。

  如果确定CSV文件列中不包含换行符，则基于Bytes的分片由于不需要创建Meta，其使用更为简便。如果列中包含换行符或者是JSON文件时，则使用以下步骤：

  i. 调用CreateSelectObjectMeta API获得该文件的总的Split数。理想情况下如果该文件需要用SelectObject，则该API最好在查询前进行异步调用，这样可以节省扫描时间。
  ii. 根据客户端资源情况选择合适的并发度n，用总的Split数除以并发度n得到每个分片查询应该包含的Split个数。
  iii. 在请求Body中用诸如split-range=1-20的形式进行分片查询。
  iv. 合并结果。

- JSON文件查询

  查询JSON文件时，在SQL的From语句中尽可能缩小From后的JSON Path范围。

  如下是JSON文件示例：

```
{
  "contacts":[
{
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ]
}
]}
```

如果要查找所有postalCode为10021开头的streetAddress，SQL可以写为 `select s.address.streetAddress from ossobject.contacts[*] s where s.address.postalCode like '10021%'` 或者 `select s.streetAddress from ossobject.contacts[*].address s where s.postalCode like '10021%'`

由于 `select s.streetAddress from ossobject.contacts[*].address s where s.postalCode like '10021%'` 的JSON Path更加精确，因此性能更优。

- 在JSON文件中处理高精度浮点数

  在JSON文件中需要进行高精度浮点数的数值计算时，建议设置ParseJsonNumberAsString选项为true，同时将该值cast成Decimal。比如一个属性a值为123456789.123456789，用 `select s.a from ossobject s where cast(s.a as decimal) > 123456789.12345` 就可以保持原始数据的精度不丢失。

使用OSS控制台

> **注意** 通过控制台仅支持从128 MB以下的文件中提取40 MB以下的数据记录。

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**文件管理 > 文件管理**。
4. 在文件管理页面，选择目标文件对应的**更多 > 选取内容**。
5. 在**选取内容**页面设置相关参数。
   - 文件类型：按文件实际情况选择文件的类型，可选项为：*CSV*和*JSON*。
   - 分隔符（针对CSV文件）：选择半角逗号（,）或自定义分隔符。
   - 标题行（针对CSV文件）：选择文件第一行是否包含列标题。
   - JSON格式符（针对JSON文件）：选择您的JSON文件对应的格式。
   - 压缩格式：选择您当前的文件是否为压缩文件。目前压缩文件仅支持GZIP文件。
6. 单击**显示文件预览**可预览文件。

   > **注意** 预览标准存储类型文件时，会产生Select扫描费用。预览低频访问、归档和冷归档存储类型文件时，会产生Select扫描费用和数据取回费用。

7. 单击**下一步**，输入SQL语句并执行。
8. 查看执行结果。单击**下载**，下载所选取的内容到本地。

   假设名为*People*的CSV文件有3列数据，分别是姓名、公司和年龄。
   - 如果想查找年龄大于50岁，并且名字以Lora开头的人（其中_1,_2,_3是列索引，代表第一列、第二列、第三列），可以执行如下SQL语句：

     ```
     select * from ossobject where _1 like 'Lora*' and _3 > 50
     ```

   - 如果想统计这个文件有多少行，最大年龄与最小年龄是多少，可以执行如下SQL语句：

     ```
     select count(*), max(cast(_3 as int)), min(cast(_3 as int)) from oss_object
     ```

使用阿里云SDK

当前仅支持通过Java SDK和Python SDK查询文件。

Python

```
import com.aliyun.oss.model.*;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
/**
 * Examples of create select object metadata and select object.
```

```
 *
 */
public class SelectObjectSample {
    // yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
    private static String endpoint = "yourEndpoint";
    // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
    private static String accessKeyId = "yourAccessKeyId";
    private static String accessKeySecret = "yourAccessKeySecret";
    // 填写Bucket名称，例如examplebucket。
    private static String bucketName = "examplebucket";
    public static void main(String[] args) throws Exception {
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        // 填写Object完整路径后，根据SELECT语句查询文件中的数据。Object完整路径中不能包含Bucket名称。
        // 填写CSV格式的Object完整路径。
        selectCsvSample("test.csv", ossClient);
        // 填写JSON格式的Object完整路径。
        selectJsonSample("test.json", ossClient);
        ossClient.shutdown();
    }
    private static void selectCsvSample(String key, OSS ossClient) throws Exception {
        // 填写上传的内容。
        String content = "name,school,company,age\r\n" +
                "Lora Francis,School A,Staples Inc,27\r\n" +
                "Eleanor Little,School B,\"Conectiv, Inc\",43\r\n" +
                "Rosie Hughes,School C,Western Gas Resources Inc,44\r\n" +
                "Lawrence Ross,School D,MetLife Inc.,24";
        ossClient.putObject(bucketName, key, new ByteArrayInputStream(content.getBytes()));
        SelectObjectMetadata selectObjectMetadata = ossClient.createSelectObjectMetadata(
                new CreateSelectObjectMetadataRequest(bucketName, key)
                        .withInputSerialization(
                                new InputSerialization().withCsvInputFormat(
                                        // 填写内容中不同记录之间的分隔符，例如\r\n。
                                        new CSVFormat().withHeaderInfo(CSVFormat.Header.Use).withRecordDelimiter("\r\n"))));
        System.out.println(selectObjectMetadata.getCsvObjectMetadata().getTotalLines());
        System.out.println(selectObjectMetadata.getCsvObjectMetadata().getSplits());
        SelectObjectRequest selectObjectRequest =
                new SelectObjectRequest(bucketName, key)
                        .withInputSerialization(
                                new InputSerialization().withCsvInputFormat(
                                        new CSVFormat().withHeaderInfo(CSVFormat.Header.Use).withRecordDelimiter("\r\n")))
                        .withOutputSerialization(new OutputSerialization().withCsvOutputFormat(new CSVFormat()));
        selectObjectRequest.setExpression("select * from ossobject where _4 > 40");// 使用SELECT语句查询文件中的数据。
        OSSObject ossObject = ossClient.selectObject(selectObjectRequest);
        // 读取内容。
        BufferedReader reader = new BufferedReader(new InputStreamReader(ossObject.getObjectContent()));
        while (true) {
            String line = reader.readLine();
            if (line == null) {
                break;
            }
            System.out.println(line);
        }
        reader.close();
        ossClient.deleteObject(bucketName, key);
    }
    private static void selectJsonSample(String key, OSS ossClient) throws Exception {
        // 填写上传的内容。
        final String content = "{\n" +
                "\t\"name\": \"Lora Francis\",\n" +
                "\t\"age\": 27,\n" +
                "\t\"company\": \"Staples Inc\"\n" +
                "}\n" +
                "{\n" +
                "\t\"name\": \"Eleanor Little\",\n" +
                "\t\"age\": 43,\n" +
                "\t\"company\": \"Conectiv, Inc\"\n" +
                "}\n" +
                "{\n" +
                "\t\"name\": \"Rosie Hughes\",\n" +
                "\t\"age\": 44,\n" +
                "\t\"company\": \"Western Gas Resources Inc\"\n" +
                "}\n" +
                "{\n" +
                "\t\"name\": \"Lawrence Ross\",\n" +
                "\t\"age\": 24,\n" +
                "\t\"company\": \"MetLife Inc.\"\n" +
                "}";
        ossClient.putObject(bucketName, key, new ByteArrayInputStream(content.getBytes()));
        SelectObjectRequest selectObjectRequest =
                new SelectObjectRequest(bucketName, key)
                        .withInputSerialization(new InputSerialization()
                                .withCompressionType(CompressionType.NONE)
                                .withJsonInputFormat(new JsonFormat().withJsonType(JsonType.LINES)))
                        .withOutputSerialization(new OutputSerialization()
                                .withCrcEnabled(true)
                                .withJsonOutputFormat(new JsonFormat()))
                        .withExpression("select * from ossobject as s where s.age > 40"); // 使用SELECT语句查询文件中的数据。
```

```
        OSSObject ossObject = ossClient.selectObject(selectObjectRequest);
        // 读取内容。
        BufferedReader reader = new BufferedReader(new InputStreamReader(ossObject.getObjectContent()));
        while (true) {
            String line = reader.readLine();
            if (line == null) {
                break;
            }
            System.out.println(line);
        }
        reader.close();
        ossClient.deleteObject(bucketName, key);
    }
}
```

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见SelectObject。

6.6. 常见问题

6.6.1. OSS有带宽和QPS限制吗？

使用OSS进行数据的上传、下载等操作，对于带宽和每秒请求数QPS有一定的限制。

您可以通过ossutil工具的probe命令检测网络状态，详情请参见probe（探测状态）。

限制项	说明
带宽	中国内地各地域10 Gbit/s、其他地域5 Gbit/s。如达到该阈值，请求会被流控。 说明 当请求被流控时，请求返回的Header中会携带 `x-oss-qos-delay-time: number` 。其中 `number` 为请求被流控的时长，单位为ms。 上传类请求会返回精确的被流控的时长；下载类请求会返回根据流控程度和文件大小估算出的被流控的时长。 如果您的业务（如大数据离线处理等）有更大的带宽（10 Gbit/s~100 Gbit/s）需求，请联系技术支持。
每秒请求数QPS（Query Per Second）	单个账号的总QPS为10,000，但在不同的读写方式下，实际能达到的值如下： • 顺序读写：2,000 如果您在上传大量文件时，在命名上使用了顺序前缀（如时间戳或字母顺序），可能会出现大量文件索引集中存储于存储空间中某个特定分区的情况，此时如果您的请求速率过大，会导致请求速率下降。建议您在上传大量文件时，不要使用顺序前缀的文件名。关于如何将顺序前缀改为随机性前缀的方法，请参见OSS性能与扩展性最佳实践。 • 非顺序读写：10,000 如您的业务有更大的QPS需求，请联系技术支持。

6.6.2. 上传Object后如何获取访问URL？

本文举例说明如何在上传文件（Object）后获取文件的访问地址。

公共读Object

如果文件的读写权限ACL为公共读，即该文件允许匿名访问，那么文件URL的格式为 `https://BucketName.Endpoint/ObjectName` 。其中，ObjectName需填写包含文件夹以及文件后缀在内的该文件的完整路径。各地域的Endpoint信息介绍，请参见访问域名和数据中心。

例如华东1（杭州）地域下名为bucketexample的Bucket下有名为example的文件夹，文件夹内有个名为example.jpg的文件。则该文件URL为：

- 外网访问URL： `https://bucketexample.oss-cn-hangzhou.aliyuncs.com/example/example.jpg`
- 内网访问URL（供同地域ECS实例访问）： `https://bucketexample.oss-cn-hangzhou-internal.aliyuncs.com/example/example.jpg`

> **注意** 如需确保通过文件URL访问文件时是预览行为，您需要绑定自定义域名并添加CNAME记录。详情请参见使用自有域名访问OSS资源。

私有Object

如果文件读写权限ACL为私有，则必须进行签名操作。私有文件URL的格式为 `https://BucketName.Endpoint/Object?签名参数` 。您可以通过以下任意方法获取文件URL并设置URL的有效时长。

- 控制台

  您可以通过OSS控制台获取文件URL。具体操作，请参见分享文件。文件URL的有效时长因账号类型存在差异。例如，阿里云账号可设置的文件URL有效时长最大为32400秒（9小时），RAM用户以及STS用户可设置的文件URL有效时长最大为3600秒（1小时）。如需获取更长时效的文件URL，请使用命令行工具ossutil、图形化工具ossbrowser或SDK。

- 命令行工具ossutil

  请参见ossutil-sign。

- 图形化工具ossbrowser

  请参见ossbrowser快速入门。

- SDK

  请参见：

  - Java
  - Python

- Go
- PHP
- C
- .NET
- Android
- iOS
- Node.js
- Browser.js

自有域名Object

如果文件所在的Bucket绑定了自定义域名，则文件URL的格式为 `https://YourDomainName/ObjectName` ，其中ObjectName需填写包含文件夹以及文件后缀在内的该文件的完整路径。

例如您在华东1（杭州）地域下的存储空间bucketexample，绑定了自有域名 `img.example.com` 。且该bucket下有名为example的文件夹，文件夹内有名为*example.jpg*的文件，则该文件URL为 `https://img.example.com/example/example.jpg` 。

6.6.3. 如何设置Content-Type（MIME）？

Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。多用于指定一些客户端自定义的文件，以及一些媒体文件的打开方式。

OSS默认自动识别文件类型。例如上传的文件类型为.jpg，OSS会自动将该文件识别为图片文件。您可以通过控制台、图形化工具ossbrowser、命令行工具ossutil、各语言SDK例如Java SDK、Python SDK、PHP SDK、Go SDK等方式来修改文件类型。

常见Content-Type（MIME）列表如下：

文件扩展名	Content-Type(Mime-Type)	文件扩展名	Content-Type(Mime-Type)
（二进制流，未知的文件类型）	application/octet-stream	.tif	image/tiff
0.001	application/x-001	0.301	application/x-301
0.323	text/h323	0.906	application/x-906
0.907	drawing/907	.a11	application/x-a11
.acp	audio/x-mei-aac	.ai	application/postscript
.aif	audio/aiff	.aifc	audio/aiff
.aiff	audio/aiff	.anv	application/x-anv
.apk	application/vnd.android.package-archive	.asa	text/asa
.asf	video/x-ms-asf	.asp	text/asp
.asx	video/x-ms-asf	.au	audio/basic
.avi	video/avi	.awf	application/vnd.adobe.workflow
.biz	text/xml	.bmp	application/x-bmp
.bot	application/x-bot	.c4t	application/x-c4t
.c90	application/x-c90	.cal	application/x-cals
.cat	application/vnd.ms-pki.seccat	.cdf	application/x-netcdf
.cdr	application/x-cdr	.cel	application/x-cel
.cer	application/x-x509-ca-cert	.cg4	application/x-g4
.cgm	application/x-cgm	.cit	application/x-cit
.class	java/	.cml	text/xml
.cmp	application/x-cmp	.cmx	application/x-cmx
.cot	application/x-cot	.crl	application/pkix-crl
.crt	application/x-x509-ca-cert	.csi	application/x-csi
.css	text/css	.cut	application/x-cut
.dbf	application/x-dbf	.dbm	application/x-dbm
.dbx	application/x-dbx	.dcd	text/xml
.dcx	application/x-dcx	.der	application/x-x509-ca-cert
.dgn	application/x-dgn	.dib	application/x-dib
.dll	application/x-msdownload	.doc	application/msword

文件扩展名	Content-Type(Mime-Type)	文件扩展名	Content-Type(Mime-Type)
.docx	application/vnd.openxmlformats-officedocument.wordprocessingml.document	.dot	application/msword
.dotx	application/vnd.openxmlformats-officedocument.wordprocessingml.template	.drw	application/x-drw
.dtd	text/xml	.dwf	Model/vnd.dwf
.dwf	application/x-dwf	.dwg	application/x-dwg
.dxb	application/x-dxb	.dxf	application/x-dxf
.edn	application/vnd.adobe.edn	.emf	application/x-emf
.eml	message/rfc822	.ent	text/xml
.epi	application/x-epi	.eps	application/x-ps
.eps	application/postscript	.etd	application/x-ebx
.exe	application/x-msdownload	.fax	image/fax
.fdf	application/vnd.fdf	.fif	application/fractals
.fo	text/xml	.frm	application/x-frm
.g4	application/x-g4	.gbr	application/x-gbr
.	application/x-	.gif	image/gif
.gl2	application/x-gl2	.gp4	application/x-gp4
.hgl	application/x-hgl	.hmr	application/x-hmr
.hpg	application/x-hpgl	.hpl	application/x-hpl
.hqx	application/mac-binhex40	.hrf	application/x-hrf
.hta	application/hta	.htc	text/x-component
.htm	text/html	.html	text/html
.htt	text/webviewhtml	.htx	text/html
.icb	application/x-icb	.ico	image/x-icon
.ico	application/x-ico	.iff	application/x-iff
.ig4	application/x-g4	.igs	application/x-igs
.iii	application/x-iphone	.img	application/x-img
.ins	application/x-internet-signup	.ipa	application/vnd.iphone
.isp	application/x-internet-signup	.IVF	video/x-ivf
.java	java/*	.jfif	image/jpeg
.jpe	image/jpeg	.jpe	application/x-jpe
.jpeg	image/jpeg	.jpg	image/jpeg
.jpg	application/x-jpg	.js	application/x-javascript
.jsp	text/html	.la1	audio/x-liquid-file
.lar	application/x-laplayer-reg	.latex	application/x-latex
.lavs	audio/x-liquid-secure	.lbm	application/x-lbm
.lmsff	audio/x-la-lms	.ls	application/x-javascript
.ltr	application/x-ltr	.m1v	video/x-mpeg
.m2v	video/x-mpeg	.m3u	audio/mpegurl
.m4e	video/mpeg4	.mac	application/x-mac
.man	application/x-troff-man	.math	text/xml
.mdb	application/msaccess	.mdb	application/x-mdb
.mfp	application/x-shockwave-flash	.mht	message/rfc822
.mhtml	message/rfc822	.mi	application/x-mi

文件扩展名	Content-Type(Mime-Type)	文件扩展名	Content-Type(Mime-Type)
.mid	audio/mid	.midi	audio/mid
.mil	application/x-mil	.mml	text/xml
.mnd	audio/x-musicnet-download	.mns	audio/x-musicnet-stream
.mocha	application/x-javascript	.movie	video/x-sgi-movie
.mp1	audio/mp1	.mp2	audio/mp2
.mp2v	video/mpeg	.mp3	audio/mp3
.mp4	video/mp4	.mpa	video/x-mpg
.mpd	application/vnd.ms-project	.mpe	video/x-mpeg
.mpeg	video/mpg	.mpg	video/mpg
.mpga	audio/rn-mpeg	.mpp	application/vnd.ms-project
.mps	video/x-mpeg	.mpt	application/vnd.ms-project
.mpv	video/mpg	.mpv2	video/mpeg
.mpw	application/vnd.ms-project	.mpx	application/vnd.ms-project
.mtx	text/xml	.mxp	application/x-mmxp
.net	image/pnetvue	.nrf	application/x-nrf
.nws	message/rfc822	.odc	text/x-ms-odc
.out	application/x-out	.p10	application/pkcs10
.p12	application/x-pkcs12	.p7b	application/x-pkcs7-certificates
.p7c	application/pkcs7-mime	.p7m	application/pkcs7-mime
.p7r	application/x-pkcs7-certreqresp	.p7s	application/pkcs7-signature
.pc5	application/x-pc5	.pci	application/x-pci
.pcl	application/x-pcl	.pcx	application/x-pcx
.pdf	application/pdf	.pdb	chemical/x-pdb
.pdx	application/vnd.adobe.pdx	.pfx	application/x-pkcs12
.pgl	application/x-pgl	.pic	application/x-pic
.pko	application/vnd.ms-pki.pko	.pl	application/x-perl
.plg	text/html	.pls	audio/scpls
.plt	application/x-plt	.png	image/png
.png	application/x-png	.pot	application/vnd.ms-powerpoint
.potx	application/vnd.openxmlformats-officedocument.presentationml.template	.ppa	application/vnd.ms-powerpoint
.ppm	application/x-ppm	.pps	application/vnd.ms-powerpoint
.ppsx	application/vnd.openxmlformats-officedocument.presentationml.slideshow	.ppt	application/vnd.ms-powerpoint
.ppt	application/x-ppt	. pptx	application/vnd.openxmlformats-officedocument.presentationml.presentation
.pr	application/x-pr	.prf	application/pics-rules
.prn	application/x-prn	.prt	application/x-prt
.ps	application/x-ps	.ps	application/postscript
.ptn	application/x-ptn	.pwz	application/vnd.ms-powerpoint
.r3t	text/vnd.rn-realtext3d	.ra	audio/vnd.rn-realaudio
.ram	audio/x-pn-realaudio	.ras	application/x-ras
.rat	application/rat-file	.rdf	text/xml
.rec	application/vnd.rn-recording	.red	application/x-red

文件扩展名	Content-Type(Mime-Type)	文件扩展名	Content-Type(Mime-Type)
.rgb	application/x-rgb	.rjs	application/vnd.rn-realsystem-rjs
.rjt	application/vnd.rn-realsystem-rjt	.rlc	application/x-rlc
.rle	application/x-rle	.rm	application/vnd.rn-realmedia
.rmf	application/vnd.adobe.rmf	.rmi	audio/mid
.rmj	application/vnd.rn-realsystem-rmj	.rmm	audio/x-pn-realaudio
.rmp	application/vnd.rn-rn_music_package	.rms	application/vnd.rn-realmedia-secure
.rmvb	application/vnd.rn-realmedia-vbr	.rmx	application/vnd.rn-realsystem-rmx
.rnx	application/vnd.rn-realplayer	.rp	image/vnd.rn-realpix
.rpm	audio/x-pn-realaudio-plugin	.rsml	application/vnd.rn-rsml
.rt	text/vnd.rn-realtext	.rtf	application/msword
.rtf	application/x-rtf	.rv	video/vnd.rn-realvideo
.sam	application/x-sam	.sat	application/x-sat
.sdp	application/sdp	.sdw	application/x-sdw
.sis	application/vnd.symbian.install	.sisx	application/vnd.symbian.install
.sit	application/x-stuffit	.slb	application/x-slb
.sld	application/x-sld	.sldx	application/vnd.openxmlformats-officedocument.presentationml.slide
.slk	drawing/x-slk	.smi	application/smil
.smil	application/smil	.smk	application/x-smk
.snd	audio/basic	.sol	text/plain
.sor	text/plain	.spc	application/x-pkcs7-certificates
.spl	application/futuresplash	.spp	text/xml
.ssm	application/streamingmedia	.sst	application/vnd.ms-pki.certstore
.stl	application/vnd.ms-pki.stl	.stm	text/html
.sty	application/x-sty	.svg	image/svg+xml
.swf	application/x-shockwave-flash	.tdf	application/x-tdf
.tg4	application/x-tg4	.tga	application/x-tga
.tif	image/tiff	.tif	application/x-tif
.tiff	image/tiff	.tld	text/xml
.top	drawing/x-top	.torrent	application/x-bittorrent
.tsd	text/xml	.txt	text/plain
.uin	application/x-icq	.uls	text/iuls
.vcf	text/x-vcard	.vda	application/x-vda
.vdx	application/vnd.visio	.vml	text/xml
.vpg	application/x-vpeg005	.vsd	application/vnd.visio
.vsd	application/x-vsd	.vss	application/vnd.visio
.vst	application/vnd.visio	.vst	application/x-vst
.vsw	application/vnd.visio	.vsx	application/vnd.visio
.vtx	application/vnd.visio	.vxml	text/xml
.wav	audio/wav	.wax	audio/x-ms-wax
.wb1	application/x-wb1	.wb2	application/x-wb2
.wb3	application/x-wb3	.wbmp	image/vnd.wap.wbmp
.wiz	application/msword	.wk3	application/x-wk3

文件扩展名	Content-Type(Mime-Type)	文件扩展名	Content-Type(Mime-Type)
.wk4	application/x-wk4	.wkq	application/x-wkq
.wks	application/x-wks	.wm	video/x-ms-wm
.wma	audio/x-ms-wma	.wmd	application/x-ms-wmd
.wmf	application/x-wmf	.wml	text/vnd.wap.wml
.wmv	video/x-ms-wmv	.wmx	video/x-ms-wmx
.wmz	application/x-ms-wmz	.wp6	application/x-wp6
.wpd	application/x-wpd	.wpg	application/x-wpg
.wpl	application/vnd.ms-wpl	.wq1	application/x-wq1
.wr1	application/x-wr1	.wri	application/x-wri
.wrk	application/x-wrk	.ws	application/x-ws
.ws2	application/x-ws	.wsc	text/scriptlet
.wsdl	text/xml	.wvx	video/x-ms-wvx
.xap	application/x-silverlight-app	.x_b	application/x-x_b
.xdp	application/vnd.adobe.xdp	.xdr	text/xml
.xfd	application/vnd.adobe.xfd	.xfdf	application/vnd.adobe.xfdf
.xhtml	text/html	.xls	application/vnd.ms-excel
.xls	application/x-xls	.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
.xltx	application/vnd.openxmlformats-officedocument.spreadsheetml.template	.xlw	application/x-xlw
.xml	text/xml	.xpl	audio/scpls
.xq	text/xml	.xql	text/xml
.xquery	text/xml	.xsd	text/xml
.xsl	text/xml	.xslt	text/xml
.xwd	application/x-xwd	.x_t	application/x-x_t
.yaml	text/vnd.yaml	.yml	text/vnd.yml
.webp	image/webp	无	无

6.6.4. OSS有哪些批量操作？

对象存储OSS提供丰富的访问和管理文件（Object）的方式，本文介绍如何批量管理Object。

批量上传

您可以使用以下方法，批量上传文件：

- ossimport工具

  支持从服务器本地、第三方云存储（S3、Azure、腾讯COS等）、OSS等数据源将数据批量迁移到OSS，特别适合数据量很大的情况。详情请参见说明及配置。
- ossutil工具

  使用ossutil工具的**cp**命令，结合-r（--recursive）选项，可批量上传文件到OSS。详情请参见上传文件。
- ossbrowser工具

  使用ossbrowser工具批量选中文件后上传到OSS。详情请参见上传文件。
- OSS控制台

  使用OSS控制台批量选中文件后上传到OSS。详情请参见上传文件。

批量下载

您可以使用以下方法，批量下载文件：

- ossutil工具

  使用ossutil工具的**cp**命令，结合-r（--recursive）选项，将指定文件目录内的文件批量下载到本地。详情请参见下载文件。
- ossbrowser工具

  使用ossbrowser工具勾选多个文件或文件目录，将文件或文件目录批量下载到本地。详情请参见下载文件。
- OSS控制台

  使用OSS控制台勾选多个文件，将文件批量下载到本地。详情请参见下载文件。

- 文件打包后下载

  结合函数计算服务，您可以将批量文件打包后下载到本地。详情请参见使用函数计算打包下载OSS文件。

批量复制

您可以使用以下方法，批量复制文件：

- 跨区域复制

  通过跨区域复制可以对指定前缀文件进行批量复制。您还可以选择是否同步历史数据、是否同步删除操作。详情请参见设置跨区域复制。
- ossutil工具

  使用ossutil工具的**cp**命令，结合-r（--recursive）选项，将指定文件目录内的文件批量复制到另一个文件目录或同账号下的另一个存储空间内。详情请参见复制文件。
- ossbrowser工具

  使用ossbrowser工具勾选多个文件夹或文件，将一个或多个文件复制到另一个文件目录或同账号下另一个存储空间内。详情请参见复制文件。

批量删除

您可以使用以下方法，批量删除文件：

> **警告** 文件删除后不可恢复，请谨慎操作。

- OSS SDK

  使用SDK批量删除文件。
  - Java SDK
  - Python SDK
  - Go SDK
  - C++ SDK

  更多语言的SDK示例请参见SDK 参考。
- OSS API

  通过OSS的DeleteMultipleObjects接口可批量删除文件。详情请参见DeleteMultipleObjects。
- ossutil工具

  使用ossutil的**rm**命令，结合-r（--recursive）选项，将指定前缀的文件批量删除。详情请参见删除文件。
- ossbrowser工具

  使用ossbrowser工具勾选多个文件或文件目录，批量删除。详情请参见删除文件。
- OSS控制台
  - 使用OSS控制台勾选多个文件，批量删除。详情请参见删除文件。

    您也可以直接删除某个文件目录，文件目录内的文件会同时被删除。
  - 使用OSS控制台的碎片管理功能，批量删除碎片。详情请参见管理碎片。
- 生命周期规则

  通过生命周期规则批量自动删除您的文件。详情请参见基于最后一次修改时间的生命周期规则介绍。

批量修改文件存储类型

您可以使用以下方法，批量修改文件的存储类型：

- ossutil工具

  使用ossutil的**set-meta**命令，结合-r（--recursive）选项，批量修改指定文件的存储类型。详情请参见set-meta（管理文件元信息）。
- 生命周期规则

  通过生命周期规则批量自动修改文件的存储类型。详情请参见基于最后一次修改时间的生命周期规则介绍。

批量修改文件访问权限（ACL）

您可以通过ossutil工具批量修改文件ACL：

- 使用**set-acl**命令，结合-r（--recursive）选项，批量修改指定文件的ACL。详情请参见set-acl（设置或修改ACL）。
- 使用**set-meta**命令结合-r（--recursive）选项，通过修改指定文件的meta信息来修改文件的ACL。详情请参见set-meta（管理文件元信息）。

批量解冻文件

您可以使用以下方法，批量将归档和冷归档存储“冷冻”状态的文件恢复为可读：

- ossutil工具

  使用**restore**命令，结合-r（--recursive）选项，批量恢复冷冻状态的文件为可读状态。详情请参见restore（解冻文件）。
- ossbrowser工具

  使用ossbrowser工具勾选需要解冻的文件，批量解冻。

批量设置文件Meta信息

您可以使用以下方法，批量修改文件的Meta信息：

- ossutil工具

  使用ossutil的**set-meta**命令，结合-r（--recursive）选项，批量修改指定文件的meta信息。详情请参见set-meta（管理文件元信息）。

  此命令可用于批量修改文件的存储类型及文件访问权限。
- OSS控制台

在控制台上勾选需要修改Http head信息的文件，批量设置文件的Meta信息。详情请参见设置文件元信息。

6.6.5. OSS怎样上传下载文件夹（目录）？

阿里云OSS的数据模型为扁平型结构，所有文件（Object）都直接隶属于其对应的存储空间（Bucket）。因此，OSS缺少文件系统中类似于文件夹与子文件夹的层次结构。为了方便管理，OSS管理控制台将所有文件名以正斜线（/）结尾的文件显示为文件夹，实现类似于Windows文件夹的基本功能。

例如*abc/efg/123.jpg*这个路径的文件，在OSS管理控制台上看起来就是*123.jpg存放在abc文件夹下的efg子文件夹中*。

您可以通过以下方法上传或下载文件夹：

- OSS管理控制台：图形化管理工具，提供类似Windows资源管理器的功能，使用简单。
  - 上传文件夹：在上传时，直接将文件夹拖拽到上传区域，即可保留文件夹的结构。具体操作，请参见上传文件。
  - 下载文件夹：OSS控制台暂不支持直接下载文件夹，您可以在本地创建文件夹后，将Bucket中的文件批量下载到指定文件夹中。具体操作，请参见下载文件。
- ossbrowser：图形化管理工具，提供类似Windows资源管理器的功能，使用简单。
  - 上传文件夹：在指定的Bucket或目录内，单击**目录**，之后选中需要上传的文件夹即可。您也可以直接将文件夹拖拽到ossbrowser中。具体操作，请参见上传文件。
  - 下载文件夹：单击指定文件夹右侧的下载，即可下载文件夹。具体操作，请参见下载文件。
- ossutil：命令行管理工具，提供方便、简洁、丰富的OSS管理命令，操作性能好。
  - 上传文件夹：在上传文件时携带-r选项可上传文件夹。具体操作，请参见上传文件。
  - 下载文件夹：在下载文件时携带-r选项可下载文件夹。具体操作，请参见下载文件
- SDK：提供丰富、完整的各类语言SDK demo，易于开发。
  - 上传文件夹：SDK不支持直接上传文件夹，您可以在上传时设置相同的文件名前缀，并使用正斜线（/）隔开。例如上传*a.txt*、*b.txt*、*c.txt*三个文件到abc文件夹，在上传时设置ObjectName为*abc/a.txt*、*abc/b.txt*、*abc/c.txt*即可。
  - 下载文件夹：SDK不支持直接下载文件夹，您可以在下载时将文件下载到同一个本地文件夹中。

6.6.6. OSS如何限制上传文件类型及大小？

对象存储OSS本身并不限制上传的文件类型、大小。如果有相关需求的话，需要您自行在业务层面实现。本文介绍通过Web直传时如何限制上传文件的类型和大小。

前提条件

已完成Web端直传环境安装。

Web端直传操作方法请参见JavaScript客户端签名直传。

配置步骤

您可以利用Plupload的属性filters设置上传的过滤条件，如设置只能上传图片、上传文件的大小、不能有重复上传等。

1. 打开*upload.js*文件。
2. 在 `var uploader = new plupload.Uploader` 实例下增加如下字段后保存。

```
    filters: {
        mime_types : [ //只允许上传图片和zip文件
        { title : "Image files", extensions : "jpg,gif,png,bmp" },
        { title : "Zip files", extensions : "zip" }
        ],
        max_file_size : '400kb', //最大只能上传400KB的文件
        prevent_duplicates : true //不允许选取重复文件
    },
```

   - mime_types：限制上传的文件后缀。
   - max_file_size：限制上传的文件大小。
   - prevent_duplicates：限制不能重复上传。
3. 打开*index.html*文件，测试上传。

   单击**选择文件**后发现，仅可以选择JPG、GIF、PNG、BMP、ZIP格式的文件，且文件大小不能超过400KB。

6.6.7. OSS MD5一致性校验说明

OSS上的Object会有ETag标签，ETag主要是用来判断服务端数据是否存在变化。但是ETag不一定等同于文件的MD5值，所以不建议作为校验数据一致性的依据。

如果需要校验上传到OSS的文件和本地文件是否一致，可以在上传文件时携带文件的Content-MD5值。OSS会在接收文件时，将文件的MD5值和Content-MD5进行比对，两者一致时才可以上传成功，从而保证上传数据的一致性。

以消息内容“123456789”为例，以下详细说明正确及错误计算该字符串的Content-MD5的方法。

- 正确计算示例
  i. 先计算MD5加密的二进制数组（128位）。
  ii. 对该二进制数组进行base64编码（而不是对32位字符串编码）。

  以Python为例：

```
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")   //在Python 3中此处需要改为hash.update(b"0123456789")。
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
```

  hash.digest()，计算出二进制数组（128位）。

```
>>> hash.digest()
'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
```

- 错误计算示例

> **说明** 常见错误是直接对计算出的32位字符串进行base64编码。

```
# hash.hexdigest()，计算得到可见的32位字符串编码。
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
# 错误的MD5值进行base64编码后的结果。
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

6.6.8. OSS的gzip压缩如何使用？

OSS通过在Get请求的Header中添加Accept-Encoding为gzip，对常见网页静态文件（HTML、Javascript、XML、json）内容进行gzip压缩。

前提条件

- 文件大于或者等于1 KB。
- Conetnt-Type必须为text/cache-manifest、text/xml、text/plain、text/css、application/javascript、application/x-javascript、application/rss+xml、application/json或者text/json。

API示例

- 请求示例

```
GET /ossutil.txt HTTP/1.1
Host: agent-test1.oss-cn-qingdao.aliyuncs.com
User-Agent: curl/7.47.0
Accept: */*
Accept-Encoding: gzip
```

- 返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Thu, 23 May 2019 02:03:39 GMT
Content-Type: text/plain; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
x-oss-request-id: 5CE5FF7BFEC931F2900F9F2A
Last-Modified: Thu, 23 May 2019 02:01:11 GMT
x-oss-object-type: Normal
x-oss-hash-crc64ecma: 316181249502703****
x-oss-storage-class: Standard
Content-MD5: XSpWpgD//mzytMaVJCE7****
x-oss-server-time: 0
Content-Encoding: gzip
[965 bytes data]
```

6.6.9. 如何配置HTTPS请求和证书？

如果您的用户域名需要通过HTTPS的方式访问OSS服务，必须购买相应的数字证书并进行证书托管。详情请参见证书托管。

OSS默认支持HTTP和HTTPS两种访问方式。如果Bucket Owner（UID为175708322470****）需要强制匿名用户对目标存储空间examplebucket内资源的所有请求访问方式为HTTPS，请通过策略语法的方式配置以下Bucket Policy：

```
{
  "Version": "1",
  "Statement": [{
      "Effect": "Deny",
      "Action": [
        "oss:*"
      ],
      "Principal": [
        "*"
      ],
      "Resource": [
        "acs:oss:*:175708322470****:examplebucket",
        "acs:oss:*:175708322470****:examplebucket/*"
      ],
      "Condition": {
        "Bool": {
          "acs:SecureTransport": [
            "false"
          ]
        }
      }
    }
      ]
  }
```

有关策略语法涉及的各个元素的说明，请参见RAM Policy概述。

有关Bucket Policy的配置详情，请参见通过Bucket Policy授权用户访问指定资源。

6.6.10. OSS文件删除或覆盖后能不能恢复？

OSS后端的冗余机制是针对服务器、硬件等出现故障进行数据恢复。阿里云无法恢复经您手动删除、覆盖或者配置规则自动删除的OSS数据。

数据删除和覆盖说明

OSS服务条款与服务等级协议中关于数据删除和覆盖的说明如下：

- 服务条款中关于“用户业务数据”的说明如下：
  - 您可自行对您的用户业务数据进行删除、更改等操作。如您自行释放服务或删除数据的，阿里云将删除您的数据，按照您的指令不再保留该数据。
  - 用户业务数据一经删除，即不可恢复；您应自行承担数据因此被删除所引发的后果和责任，您理解并同意，阿里云没有继续保留、导出或者返还用户业务数据的义务。
- 服务等级协议中关于“数据可销毁性”服务要求如下：

  在用户主动删除数据或用户服务期满后需要销毁数据时，阿里云将自动清除对应物理服务器上磁盘和内存数据，使得数据无法恢复。

可能会造成数据删除或覆盖的操作

以下方式可能导致您的数据被删除或覆盖，请慎重操作。

- 通过OSS控制台、命令行工具ossutil、图形化工具ossbrowser、SDK等方式删除文件。详情请参见删除文件。
- 通过OSS控制台、命令行工具ossutil、图形化工具ossbrowser、SDK等方式上传同名文件到OSS，会导致OSS内已有文件被覆盖。
- 若您在生命周期规则中配置了定期删除文件的规则，OSS会根据生命周期的配置定期删除符合条件的文件。详情请参见设置生命周期规则。
- 若您配置了跨区域复制规则，且选择的是**增/删/改同步**，则对源存储空间（Bucket）进行文件修改或删除操作时，操作会同步到目的Bucket。详情请参见设置跨区域复制。
- 没有正确的配置Bucket访问权限，导致文件被他人恶意删除或覆盖。访问权限相关说明请参见访问控制概述。

如何避免误删或误覆盖

您可以通过以下方式避免文件被误删或误覆盖：

- 开启版本控制

  开启版本控制功能后，误删除、误覆盖的文件会以历史版本的形式存储在Bucket中，您可以随时恢复历史版本。详情请参见版本控制介绍。
- 使用跨区域复制备份文件

  您可以使用跨区域复制的**增/改同步**功能，将数据备份到不同地域的另一个Bucket中。详情请参见设置跨区域复制。
- 使用定时备份功能定时备份文件

  您可以配置定时备份功能，将您的文件定时备份到混合云备份中。当您的文件丢失时，可第一时间恢复您的文件。详情请参见设置定时备份。
- 配置正确的访问权限

  为Bucket的访问者配置正确的访问权限，配置时注意遵循以下原则：
  - 不使用主账号访问OSS。
  - 读写分离，对于只需要读数据的业务，只使用具有读权限的子账号或STS临时凭证。
  - 开放给临时用户访问时，推荐使用STS的临时凭证来访问OSS。
  - 针对不同的业务，授予“够用且最小的范围”的OSS权限。
  - 妥善保管数据访问的凭据，如阿里云账号密码、RAM子账号访问凭据。

  权限配置详情请参见访问控制概述。

6.6.11. 匿名用户无法访问公共读的Object

当您的文件（Object）设置为公共读后，所有用户都可以访问您的Object。但以下设置会导致匿名用户无法访问公共读的Object。

设置了请求者付费模式

开启请求者付费模式后，读取存储空间（Bucket）内数据时产生的流量费用和请求费用由请求者支付，Bucket拥有者仅支付存储费用。所以请求方必须提供身份验证信息，以便OSS能够识别请求方，从而对请求方而非Bucket拥有者收取请求所产生的费用。匿名用户访问时不会携带身份验证信息，所以会导致匿名用户访问失败。详情请参见请求者付费模式。

解决方案：

- 由Bucket拥有者生成一个带签名的文件URL给匿名用户访问，详情请参见上传Object后如何获取访问URL？。
- 关闭请求者付费模式，详情请参见设置请求者付费模式。

设置了Bucket Policy

Bucket Policy是阿里云OSS推出的针对Bucket的授权策略，您可以通过Bucket Policy禁止或允许其他用户访问您的OSS资源。所以，若您Bucket Policy设置了某些影响匿名用户访问的策略，也会导致匿名用户无法访问。Bucket Policy介绍请参见通过Bucket Policy授权用户访问指定资源。

解决方案：

排查您的Bucket Policy，修改或删除影响匿名用户访问的策略。

6.6.12. 为什么文件签名URL过期后仍可以访问？

通常情况下，已过期的文件签名URL无法被继续访问。若该URL已在本地浏览器打开，则在浏览器缓存的有效期内，您依然可以通过浏览器缓存继续访问该URL。

若您希望文件签名URL过期即失效，建议您修改文件的元信息，将Cache-Control配置为 `no-cache` 或将Expires设置为小于文件签名URL过期时间的值。详情请参见管理文件元信息。

> **注意** 该配置会让浏览器不再缓存OSS文件或缓存时间变短，可能会造成OSS的请求数和外网流出流量增加，建议您合理配置。

6.6.13. 哪些操作会影响OSS文件的LastModified属性？

LastModified（最后修改时间）是OSS文件（Object）的一个重要属性，在计费、增量迁移、生命周期规则等场景中都会涉及。您在使用OSS时，部分针对Object的操作可能会更新Object的LastModified，下表列举了针对Object的常见操作。

常见操作	使用接口	是否会更新Object的LastModified
修改ACL	CopyObject	是
	PutObjectACL	是
修改usermeta	CopyObject	是
修改存储类型	通过覆写修改：CopyObject	是
	通过生命周期修改：CommitTransition	否
修改加密算法	CopyObject	是
覆写Object	上传同名文件覆写：PutObject	是
	拷贝同名文件覆写：CopyObject	是
添加或修改Object标签	PutObjectTagging	否
删除Object标签	DeleteObjectTagging	否
归档、冷归档类型文件的解冻	RestoreObject	否

注意

- 通过控制台、ossutil、ossbrowser、SDK等方式修改Object存储类型时，都是通过覆写Object并指定新的存储类型的方式实现的。
- 覆写Object以及通过CopyObject修改Object的存储类型、加密方式会更新Object的LastModified，若上述操作涉及的Object类型为低频、归档、冷归档存储类型且存储未满规定时长，还会产生存储不足规定时长容量费用。详情请参见存储费用。

  例如某个低频存储类型的Object，您在其存储了12天后通过OSS控制台将其修改为归档存储类型，此时Object的LastModified将被相应修改。另外，因低频访问类型文件最低存储时长要求为30天，还将产生18天的存储不足规定时长容量费用。

6.6.14. 签名URL可以修改过期时间吗？

签名URL生成之后无法再修改，若您需要调整签名URL的有效期，请重新生成。详情请参见在URL中包含签名。

6.6.15. 如何修改、更新、编辑文件？

OSS本身不支持文件编辑功能（append类型Object追加数据除外）。如果需要修改已上传的Object，需要重新调用PutObject接口进行上传操作。

6.6.16. OSS中可以重命名Bucket吗？是否支持Object迁移？

OSS的Bucket不支持重命名。若需要其他名称，建议您重新创建Bucket，将原Bucket的文件迁移到新创建的Bucket后，删除原文件和原Bucket即可。

若您原Bucket内的文件较少，您可以通过拷贝文件的方式迁移您的数据。详情请参见拷贝文件。

若您原Bucket内的文件较多，可以使用以下方式迁移您的数据：

- 通过跨区域复制功能迁移数据，详情请参见跨区域复制。
- 通过在线迁移服务迁移数据，详情请参见阿里云OSS之间迁移教程。
- 通过ossimport工具迁移数据，详情请参见数据迁移工具ossimport。

6.6.17. 通过浏览器访问Bucket或Object时提示“TypeError: Failed to fetch”

问题分析：出现这种情况一般有如下可能。

- 网络连接异常导致访问失败。
- Bucket或Object的命名中包含ad字符（例如adtest、aadb），被浏览器的广告过滤插件过滤了。

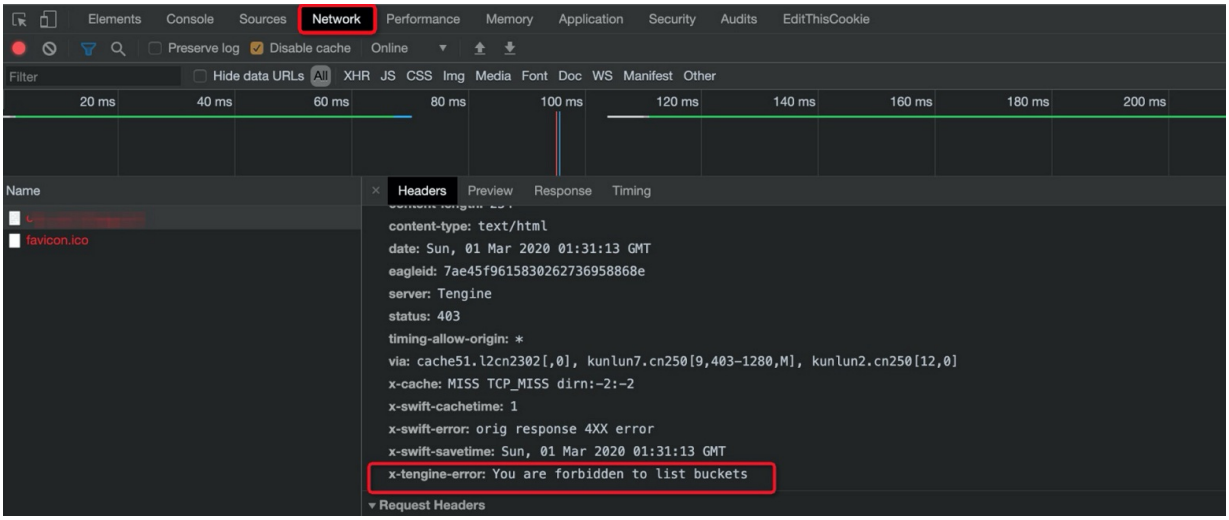
解决方案：

- 网络异常：检查您的网络，排除异常问题后重试。
- 广告过滤插件：
  - 禁用广告插件，或者将OSS的域名设置为白名单。
  - 为Bucket或Object命名时，不要包含ad字符。

6.6.18. 访问CDN加速域名报错“You are forbidden to list buckets”

本文介绍开启CDN回源私有Bucket的情况下，访问CDN加速域名出现报错的原因及解决方法。

问题现象：开启CDN回源私有Bucket的情况下，访问CDN加速域名报错 `You are forbidden to list buckets` 。



问题原因：在开启CDN回源至私有Bucket的情况下，访问CDN加速域名相当于Get Bucket(List Objects)请求，默认会被CDN拒绝。

解决方法：在CDN侧对根域名URL重写为指向根域名URL下的某个文件，例如将CDN加速域名 `example.aliyundoc.com` 重写为 `example.aliyundoc.com/index.html` 。有关重写规则的具体操作，请参见配置URI重写规则。

6.6.19. 为什么数据丢失了？

OSS是分布式存储产品，通过数据自动多重冗余备份保证数据的持久性。因此正常情况下，OSS本身是不会导致数据出现丢失的情况的。

以下情况可能会导致您的数据被删除：

- 生命周期规则

  若您配置过自动删除文件的生命周期规则，生命周期会根据您设置的周期自动删除数据。建议您根据您的需求合理配置生命周期规则。详情请参见基于最后一次修改时间的生命周期规则介绍。

- Bucket设置为允许所有人读写
  - Bucket的读写权限为公共读写：Bucket的读写权限设置为公共读写后，所有人都可以读写Bucket内的文件，建议您非必要情况下，不要设置Bucket的读写权限为公共读写。详情请参见Bucket ACL。
  - Bucket Policy设置允许所有人读写Bucket内文件：Bucket设置了允许所有人读写的策略后，所有人都可以读写Bucket内的文件，建议您非必要情况下，不要设置这样的策略。详情请参见通过Bucket Policy授权用户访问指定资源。
- 拥有Bucket管理权限的账号泄露：拥有Bucket管理权限的账号和密码、AccessKey泄露后，获得账号的人员可以随意操作您Bucket内的文件。建议您在日常管理时使用RAM用户，并给予RAM用户最小够用的管理权限。发现账号泄露的第一时间要修改RAM用户的密码，禁用AccessKey，防止损失进一步扩大。详情请参见RAM用户概览。
- 管理人员误删：OSS内文件一旦被删除，将无法找回。为防止数据误覆盖、删除，建议您使用以下功能。
  - 跨区域复制：将您Bucket内的数据备份到其他地域的其他存储空间中，源Bucket的文件被删除，还可以在备份的Bucket内找到备份文件。详情请参见跨区域复制。
  - 定时备份：开启定时备份功能后，OSS会定期将您的数据备份至混合云备份服务中，并在您出现文件丢失时，及时恢复。详情请参见设置定时备份。
  - 版本控制：开启版本控制功能后，OSS会在文件被覆盖或删除时，将文件以历史版本的形式保存下来，并在您需要的时候恢复。详情请参见版本控制介绍。
  - 合规保留策略：对于非常重要的数据，您可以开启合规保留策略，文件在保留周期内无法被覆盖、删除。详情请参见合规保留策略。

7.数据安全

7.1. 访问控制

7.1.1. 访问控制概述

默认情况下，为保证存储在OSS中数据的安全性，OSS资源（包括Bucket和Object）默认为私有权限，只有资源拥有者或者被授权的用户允许访问。如果要授权第三方用户访问或使用自己的OSS资源，可以通过多种权限控制策略向他人授予资源的特定权限。

针对存放在Bucket的Object的访问，OSS提供了以下权限控制策略：

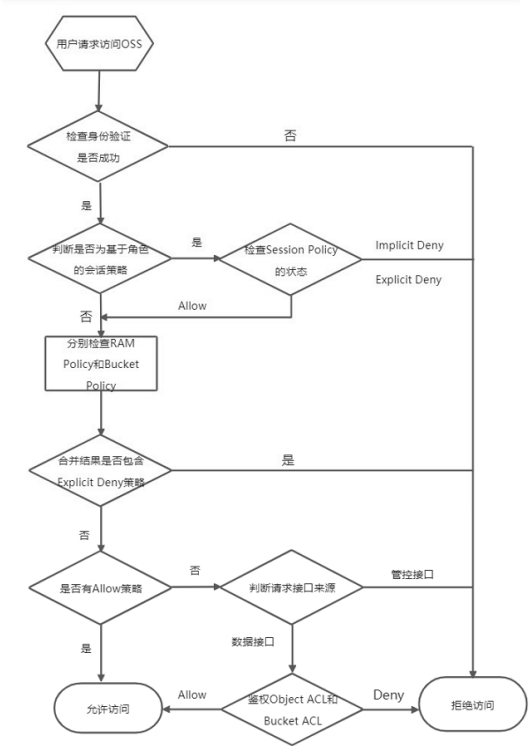
类型	说明	适用场景
RAM Policy	RAM（Resource Access Management）是阿里云提供的资源访问控制服务。RAM Policy是基于用户的授权策略。通过设置RAM Policy，您可以集中管理您的用户（比如员工、系统或应用程序），以及控制用户可以访问您名下哪些资源的权限，比如限制您的用户只拥有对某一个Bucket的读权限。	• 对同一账号下的不同RAM用户授予相同权限。 • 对所有OSS资源或者多个Bucket配置相同权限。 • 配置OSS服务级别的权限，例如列举某一账号下的所有Bucket。 • 临时授权访问OSS时，限制临时访问密钥的权限。
Bucket Policy	Bucket Policy是基于资源的授权策略。相比于RAM Policy，Bucket Policy操作简单，支持在控制台直接进行图形化配置，并且Bucket拥有者直接可以进行访问授权，无需具备RAM操作权限。Bucket Policy支持向其他账号的RAM用户授予访问权限，以及向匿名用户授予带特定IP条件限制的访问权限。	• 对同一账号下的不同RAM用户授予不同权限。 • 要进行跨账号或对匿名用户授权。
Bucket ACL	您可以在创建Bucket时设置读写权限ACL，也可以在Bucket创建后的任意时间内根据自己的业务需求随时修改ACL，该操作只有Bucket的拥有者可以执行。Bucket ACL分为*public-read-write*（公共读写）、*public-read*（公共读）和*private*（私有）三种。	对单个Bucket内的所有Object设置相同的访问权限。
Object ACL	除Bucket级别ACL以外，OSS还提供了Object级别的ACL。您可以在上传Object时设置相应的ACL，也可以在Object上传后的任意时间内根据自己的业务需求随时修改ACL。Object ACL分为*继承Bucket*、*public-read-write*（公共读写）、*public-read*（公共读）和*private*（私有）四种。	对单个Object单独授权。 例如，已通过RAM Policy或者Bucket Policy将Bucket内的所有Object或者与指定Prefix匹配的Object的访问权限设置为私有，但是考虑到您需要将其中某个Object开放给所有互联网匿名用户访问，则选择Object ACL，并将ACL设置为*public-read*。

7.1.2. OSS鉴权详解

默认情况下，为保证存储在OSS中数据的安全性，OSS资源（包括Bucket和Object）默认为私有权限，只有资源拥有者或者被授权的用户允许访问。如果要授权他人访问或使用自己的OSS资源，可以通过多种权限控制策略向他人授予资源的特定权限。仅当所有的权限策略通过OSS鉴权后允许访问授权资源。

鉴权说明

收到用户请求时，OSS会通过身份验证、基于角色的会话策略、基于身份的策略（RAM Policy）、Bucket Policy、Object ACL、Bucket ACL等鉴权结果来判断是允许或拒绝该请求。



以上鉴权流程包含的权限状态说明如下：

- Allow：允许访问请求，即比对Policy命中了Allow规则。

- Explicit Deny：显式拒绝访问请求，即比对Policy命中了Deny规则
- Implicit Deny：隐式拒绝访问请求，即Policy不存在、或比对Policy没有命中Allow或Deny规则。

鉴权流程

1. 检查身份验证是否成功

   用户请求进入OSS后，OSS会对请求携带的签名和服务端计算的签名进行比对。

   - 请求签名不匹配，则拒绝访问。
   - 请求签名匹配，则继续判断是否为基于角色的会话策略。
2. 判断是否为基于角色的会话策略

   如果判断结果是基于角色的会话策略，则OSS会对Session Policy进行权限比对。

   - 比对结果为Explicit Deny或Implicit Deny，则拒绝访问。
   - 比对结果为Allow，则继续检查RAM Policy和Bucket Policy。

   如果判断结果不是基于角色的会话策略，也会继续检查RAM Policy和Bucket Policy。
3. 分别检查RAM Policy和Bucket Policy

   RAM Policy是基于身份的策略。您可以使用RAM Policy控制用户可以访问您名下哪些资源的权限。对于用户级别的访问，需要根据请求的账号类别判断允许或拒绝访问请求。

   - 如果使用阿里云账号AccessKey访问，直接返回Implicit Deny。
   - 如果使用RAM用户AccessKey或STS的AccessKey访问，但是访问的Bucket不属于阿里云账号或者RAM角色Owner，直接返回Implicit Deny。
   - 调用RAM服务提供的鉴权接口对普通请求进行身份鉴权，OSS支持RAM服务通过账号和Bucket所属资源组进行鉴权。检查返回结果为Allow、Explicit Deny或Implicit Deny。

   Bucket Policy是基于资源的授权策略，Bucket Owner可以通过Bucket Policy为RAM用户或其他账号授权Bucket或Bucket内资源精确的操作权限。

   - 如果未设置Bucket Policy，则直接返回Implicit Deny。
   - 如果设置了Bucket Policy，则检查Bucket Policy返回结果是Allow、Explicit Deny或者Implicit Deny。
4. 合并结果中是否存在Explicit Deny策略

   如果存在，则拒绝访问。如果不存在，则检查是否存在Allow策略。

   i. 检查RAM Policy或Bucket Policy中是否存在Allow策略。

      如果存在Allow策略，则允许访问。如果不存在Allow策略，则判断请求来源。

   ii. 判断请求来源。

      如果是管控类API请求，则拒绝访问。如果为数据类API请求，则继续Object ACL或Bucket ACL的鉴权。

      管控类API请求包括Service操作（GetService (ListBuckets)）、Bucket相关操作（例如PutBucket、GetBucketLifecycle等）、LiveChannel相关操作（例如PutLiveChannel、DeleteLiveChannel等）。

      数据类API请求包括Object相关操作，例如PutObject、GetObject等。
5. 鉴权Object ACL和Bucket ACL

   根据Object ACL进行鉴权时，需要结合请求用户是否为Bucket Owner，以及请求类型为读请求或写请求进行判断。

   - 如果判断结果是Allow，则允许访问。
   - 如果判断结果是Deny，则拒绝访问。

   如果Object ACL为继承Bucket，则继续检查Bucket ACL。

   根据Bucket ACL进行鉴权时，需要结合请求用户是否为Bucket Owner进行判断。

   - 如果判断结果是Allow，则允许访问。
   - 如果判断结果是Deny，则拒绝访问。

7.1.3. Bucket ACL

读写权限ACL用于定义用户或用户组被授予的访问权限。收到某个资源的请求后，OSS会检查相应的ACL以验证请求者是否拥有所需的访问权限。您可以在创建存储空间（Bucket）时设置Bucket ACL，也可以在创建Bucket后根据自身的业务需求修改Bucket ACL。仅Bucket拥有者可以执行修改Bucket ACL的操作。

注意事项

- 如果您在上传文件（Object）时未指定文件的ACL，则文件的ACL均默认继承Bucket ACL。
- 修改Bucket ACL会影响Bucket内所有ACL为继承Bucket的文件。

读写权限类型

Bucket包含以下三种读写权限：

权限值	权限描述
public-read-write	公共读写：任何人（包括匿名访问者）都可以对该Bucket内文件进行读写操作。 **警告** 互联网上任何用户都可以对该Bucket内的文件进行访问，并且向该Bucket写入数据。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益。除特殊场景外，不建议您配置公共读写权限。
public-read	公共读：只有该Bucket的拥有者可以对该Bucket内的文件进行写操作，任何人（包括匿名访问者）都可以对该Bucket中的文件进行读操作。 **警告** 互联网上任何用户都可以对该Bucket内文件进行访问，这有可能造成您数据的外泄以及费用激增，请谨慎操作。

权限值	权限描述
private（默认值）	私有：只有Bucket的拥有者可以对该Bucket内的文件进行读写操作，其他人无法访问该Bucket内的文件。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏选择**权限管理 > 读写权限**。
4. 在**读写权限**区域单击**设置**，按实际需求修改Bucket ACL。
5. 单击**保存** 。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成修改Bucket ACL的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的修改存储空间ACL的代码示例。关于其他SDK的修改存储空间ACL代码示例，请参见SDK简介。

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.CannedAccessControlList;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 设置存储空间的读写权限。例如将examplebucket的读写权限ACL设置为私有Private。
            ossClient.setBucketAcl(bucketName, CannedAccessControlList.Private);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置或修改Bucket ACL的具体操作，请参见设置或修改Bucket ACL。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketAcl。

更多参考

除Bucket ACL以外，OSS还提供了Object ACL、Bucket Policy、RAM Policy的权限控制策略。更多信息，请参见访问控制概述。

7.1.4. Object ACL

读写权限ACL用于定义用户或用户组被授予的访问权限。收到某个资源的请求后，OSS会检查相应的ACL以验证请求者是否拥有所需的访问权限。您可以在上传Object时设置相应的ACL，也可以在Object上传后的任意时间内根据自己的业务需求随时修改ACL。

注意事项

- 如果没有设置Object的权限，即Object的ACL为default，Object的权限和Bucket的权限一致。
- 如果设置了Object的权限，Object的权限大于Bucket的权限，则以设置的Object的权限为准。例如，设置了Object的权限是public-read，则无论Bucket是什么权限，该Object都可以被身份验证访问和匿名访问。

读写权限类型

Object包含以下4种读写权限：

权限值	权限描述
public-read-write	公共读写：任何人（包括匿名访问者）都可以对该Object进行读写操作。 **警告** 互联网上任何用户都可以对该Object进行访问，并且向该Object写入数据。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益。除特殊场景外，不建议您配置公共读写权限。
public-read	公共读：只有该Object的拥有者可以对该Object进行写操作，任何人（包括匿名访问者）都可以对该Object进行读操作。 **警告** 互联网上任何用户都可以对该Object进行访问，这有可能造成您数据的外泄以及费用激增，请谨慎操作。
private	私有：只有Object的拥有者可以对该Object进行读写操作，其他人无法访问该Object。 **说明** 您可以通过文件URL将您存储空间内的私有Object分享给您的合作伙伴访问。详情请参见在URL中包含签名。
default	默认：该Object遵循Bucket的读写权限，即Bucket是什么权限，Object就是什么权限。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 单击**文件管理**页签。
4. 选择需要设置读写权限的Object。单击目标Object的文件名，在该Object的**详情**面板单击**设置读写权限**。
   您也可以将鼠标移至目标Object右侧的**更多**，在下拉菜单单击**设置读写权限**。
5. 在**设置读写权限**对话框设置Object的读写权限。
6. 单击**确定**。

使用图形化管理工具ossbrowser

ossbrowser支持Object级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成修改Object ACL的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的修改Object ACL的代码示例。关于其他SDK的修改Object ACL代码示例，请参见SDK简介。

Ad[illegible]va

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.CannedAccessControlList;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写不包含Bucket名称在内的Object完整路径，例如testfolder/exampleobject.txt。
        String objectName = "testfolder/exampleobject.txt";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 设置Object的访问权限为公共读。
            ossClient.setObjectAcl(bucketName, objectName, CannedAccessControlList.PublicRead);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置或修改Object ACL的具体操作，请参见设置或修改Object ACL。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutObjectACL。

更多参考

除Object ACL以外，OSS还提供了Bucket ACL、Bucket Policy、RAM Policy的权限控制策略。更多信息，请参见访问控制概述。

7.1.5. Bucket Policy

7.1.5.1. Bucket Policy概述

Bucket Policy是阿里云OSS推出的针对Bucket的授权策略，您可以通过Bucket Policy授权其他用户访问您指定的OSS资源。

使用场景

Bucket Policy通常应用于以下场景的授权访问：

- 需要进行跨账号或对匿名用户授权访问或管理整个Bucket或Bucket内的部分资源。
- 需要对同账号下的不同RAM用户授予访问或管理Bucket资源的不同权限，例如只读、读写或完全控制的权限。

使用OSS控制台

方式一：图形化配置Bucket Policy

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 单击**文件管理**页签，然后单击**授权**。

   您也可以通过单击**权限管理 > Bucket授权策略 > 设置**，添加授权策略。
4. 在**图形设置**页面，单击**新增授权**。
5. 在**新增授权**页面配置各项参数，然后单击**确定**。

配置项	说明

配置项	说明
授权资源	授权整个Bucket或Bucket内的部分资源供其他用户访问。 ○ **整个Bucket**：授权策略针对整个Bucket生效。 ○ **指定资源**：授权策略只针对指定的资源生效。您可以配置多条针对指定资源的授权策略。 ■ 针对目录级别授权 授权访问目录下的所有子目录和文件时，需在目录结尾处加上星号（*）。例如授权访问abc目录下的所有子目录和文件，则填写为*abc/**。 ■ 针对指定文件授权 授权访问目录下的指定文件时，需填写不包含Bucket名称在内的文件的完整路径，例如授权访问abc目录下的myphoto.png文件，则填写为*abc/myphoto.png*。
授权用户	通过选择不同类型的账号将资源授权给不同用户进行访问。 ○ **匿名账号（*）**：如果您需要给所有用户授权访问指定资源，请选中此项。 ○ **子账号**：如果您需要给当前账号下的RAM用户授权访问指定资源，请选中此项，并从下拉菜单中选择目标RAM用户。若需要授权的RAM用户较多时，建议直接在搜索框输入RAM用户名称关键字进行模糊匹配。 **注意** 您的账号必须是阿里云账号，或拥有此Bucket管理权限及RAM控制台ListUsers权限的RAM用户，否则无法查看当前账号的RAM用户列表。给RAM用户授予ListUsers权限的具体操作请参见为RAM用户授权。 ○ **其他账号**：如果您需要给其他阿里云账号、RAM用户以及通过STS生成的临时用户授予访问权限，请选中此项。 ■ 当您需要给其他阿里云账号或RAM用户授权时，请输入被授权账号的UID。 ■ 当您需要给STS临时用户授权时，输入格式为 `arn:sts::{RoleOwnerUid}:assumed-role/{RoleName}/{RoleSessionName}` 。例如生成临时用户时使用的角色为testrole，角色拥有者的阿里云账号UID为12345，生成临时用户时指定的RoleSessionName为testsession。此时应填写 `arn:sts::12345:assumed-role/testrole/testsession` 。当您需要给所有临时用户授权时，请使用通配符星号（*）。例如配置为 `arn:sts::*:*/*/*` 。生成临时授权用户的操作请参见使用STS临时访问凭证访问OSS。 **注意** 当被授权的用户是STS临时用户时，该账号无法通过OSS控制台访问授权资源，您可以通过命令行工具ossutil、OSS SDK、OSS API访问授权资源。
授权操作	您可以通过**简单设置**和**高级设置**两种方式进行授权操作。 ○ 简单设置 选中此项后，您可以结合实际场景按照如下说明配置相应的访问权限。将鼠标悬停在每一种访问权限右侧对应的 ?，可获取各访问权限对应的Action列表。 ■ **只读**：对相关资源拥有查看、列举及下载权限。 ■ **读/写**：对相关资源有读和写权限。 ■ **完全控制**：对相关资源有读、写、删除等所有操作权限。 ■ **拒绝访问**：拒绝对相关资源的所有操作。 **注意** ■ 若针对某用户同时配置了多条Bucket Policy规则，则该用户所拥有的权限是所有Policy规则的叠加。当这些Bucket Policy中包含拒绝访问权限时，遵循拒绝访问权限优先原则。例如针对某用户第一次设置了只读权限，第二次设置了读/写权限，则该用户最终的权限为读/写。如果第三次设置了拒绝访问权限，则该用户最终的权限为拒绝访问。 ■ 只读、读/写、完全控制对应的授权效力为Allow，拒绝访问对应的授权效力为Deny。 ○ 高级设置 选中此项后，您需要根据以下说明完成相关配置。 ■ **效力**：包含允许（Allow）和拒绝（Deny）两种授权效力。 ■ **操作**：支持配置所有OSS支持的Action。有关Action分类的更多信息，请参见RAM Policy概述。
条件（可选）	您还可以在基础设置和高级设置模式下选中此项，用于限定只有满足条件的用户能够访问OSS资源。 ○ **访问方式**：默认支持HTTP和HTTPS两种访问方式。如果您希望当前授权策略通过HTTPS的方式来访问Bucket资源，请选择**HTTPS**。如果您希望当前授权策略通过HTTP的方式来访问Bucket资源，请选择**HTTP**。相比HTTP，HTTPS具有更高的安全性。 如果您需要强制Bucket内资源的所有请求访问方式为其中一种，例如HTTPS，您需要通过策略语法的方式来实现。具体设置方法，请参见如何配置HTTPS请求和证书？。 ○ **IP =**：设置IP等于某个IP地址或IP地址段。如有多个IP地址，各个IP地址之间用英文逗号（,）分隔。 ○ **IP ≠**：设置IP不等于某个IP地址或IP地址段。如有多个IP地址，各个IP地址之间用英文逗号（,）分隔 。 ○ **VPC**：下拉选择专有云网络VPC ID。有关创建专有网络的具体步骤，请参见创建专有网络。

6. 单击**确定**。

方式二：通过策略语法配置Bucket Policy

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 单击**文件管理**页签，然后单击**授权**。
4. 在**策略语法**页面，单击**编辑**。

   您可以根据实际使用场景，编辑不同的策略语法，用于实现更精细的权限管理。以下为资源拥有者（UID为 `174649585760xxxx` ）为不同授权场景配置的Bucket Policy示例。

- 允许匿名用户列举存储空间examplebucket下所有文件的权限。

```
{
    "Statement": [
        {
            "Action": [
                "oss:ListObjects",
                "oss:ListObjectVersions"
            ],
            "Effect": "Allow",
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
            ]
        },
    ],
    "Version": "1"
}
```

- 示例2：拒绝源IP地址不在 `192.168.0.0/16` 范围内的匿名用户对存储空间examplebucket执行任何操作。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "oss:*",
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
            ],
            "Condition":{
                "NotIpAddress": {
                    "acs:SourceIp": ["192.168.0.0/16"]
                }
            }
        }
    ]
}
```

- 示例3：允许指定的RAM用户（UID为 `20214760404935xxxx` ）拥有目标存储空间examplebucket下 `hangzhou/2020` 和 `hangzhou/2015` 目录的只读权限。

```
{
    "Statement": [
        {
            "Action": [
                "oss:GetObject",
                "oss:GetObjectAcl",
                "oss:GetObjectVersion",
                "oss:GetObjectVersionAcl"
            ],
            "Effect": "Allow",
            "Principal": [
                "20214760404935xxxx"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket/hangzhou/2020/*",
                "acs:oss:*:174649585760xxxx:examplebucket/hangzhou/2015/*"
            ]
        },
        {
            "Action": [
                "oss:ListObjects",
                "oss:ListObjectVersions"
            ],
            "Condition": {
                "StringLike": {
                    "oss:Prefix": [
                        "hangzhou/2020/*",
                        "hangzhou/2015/*"
                    ]
                }
            },
            "Effect": "Allow",
            "Principal": [
                "20214760404935xxxx"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
            ]
        }
    ],
    "Version": "1"
}
```

5. 单击**保存**。

使用图形化管理工具ossbrowser

ossbrowser支持Bucket级别的操作与控制台支持的操作类似，请按照ossbrowser界面指引完成修改Bucket ACL的操作。关于如何使用ossbrowser，请参见快速使用ossbrowser。

使用阿里云SDK

以下仅列举常见SDK的配置Bucket Policy的代码示例。关于其他SDK的配置Bucket Policy的代码示例，请参见SDK简介。

Nodejs

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 填写policyText。
            String policyText = "{\"Statement\": [{\"Effect\": \"Allow\", \"Action\": [\"oss:GetObject\", \"oss:ListObjects\"], \"Resource\": [\"acs:
oss:*:*:*/user1/*\"]}], \"Version\": \"1\"}";
            // 设置授权策略。
            ossClient.setBucketPolicy(bucketName, policyText);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置或修改Bucket ACL的具体操作，请参见bucket-policy（授权策略）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketAcl。

访问授权资源

Bucket Policy配置完成后，您可以通过以下方式访问授权资源：

- 文件URL（仅当授权对象为匿名用户时）

  在浏览器上，使用Bucket默认域名或自有域名加文件路径进行访问。例如 `http://mybucket.oss-cn-beijing.aliyuncs.com/file/myphoto.png` 。详情请参见OSS访问域名使用规则。
- 控制台

  登录OSS控制台，在左边菜单栏单击**我的访问路径**后的加号（+），添加授权访问的Bucket和文件路径。具体操作，请参见设置我的访问路径。
- 命令行工具ossutil

  使用被授权的账号通过ossutil访问授权资源。具体操作，请参见ossutil。
- 图形化工具ossbrowser

  使用被授权的账号登录ossbrowser，登录时在**预设OSS路径**栏输入被授权访问的文件目录。具体操作，请参见ossbrowser。
- OSS SDK

  支持通过Java、PHP、Node.js、Python、Browser.js、.NET、Android、Go、iOS、C++、C SDK访问授权资源。

更多参考

- 教程示例：通过Bucket Policy限制公网访问OSS
- 教程示例：基于Bucket Policy实现跨部门数据共享
- 教程示例：基于Bucket Policy实现跨账号访问OSS

7.1.5.2. Bucket Policy常见示例

Bucket Policy是阿里云OSS推出的针对Bucket的授权策略，您可以通过Bucket Policy授权其他用户访问您指定的OSS资源。例如，您可以对同账号以及跨账号下的不同RAM用户，或者匿名用户等授予访问或管理Bucket资源的不同权限，例如只读、读写权限等。

通用说明

以下均为资源拥有者（即UID为 `174649585760xxxx` 的Bucket Owner）通过Bucket Policy授权指定用户（例如UID为 `27737962156157xxxx` 的RAM用户）不同权限的示例。与RAM Policy不同的是，Bucket Policy还包含了用于指定授权用户的Principal元素。Bucket Policy的其他元素，例如Action，Condition等用法遵循RAM Policy的语法规则。有关各元素的使用详情，请参见RAM Policy概述。

注意事项

配置Bucket Policy时，如果授权用户（Principal）选择了匿名账号（*），且不包含Condition的情况下，则Bucket Policy仅对Bucket Owner以外的所有用户生效。详情请参见示例三。

配置Bucket Policy时，如果授权用户（Principal）选择了匿名账号（*），且包含Condition的情况下，则Bucket Policy会对包含Bucket Owner在内的所有用户生效。详情请参见示例四。

示例一：授予指定RAM用户对某个Bucket的读写权限

以下示例用于授权UID为 `27737962156157xxxx` 以及 `20214760404935xxxx` 的RAM用户拥有目标存储空间examplebucket的读写权限：

```
{
    "Version": "1",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "oss:GetObject",
            "oss:PutObject",
            "oss:GetObjectAcl",
            "oss:PutObjectAcl",
            "oss:ListObjects",
            "oss:AbortMultipartUpload",
            "oss:ListParts",
            "oss:RestoreObject",
            "oss:GetVodPlaylist",
            "oss:PostVodPlaylist",
            "oss:PublishRtmpStream",
            "oss:ListObjectVersions",
            "oss:GetObjectVersion",
            "oss:GetObjectVersionAcl",
            "oss:RestoreObjectVersion"
        ],
        "Principal": [
            "27737962156157xxxx",
            "20214760404935xxxx"
        ],
        "Resource": [
            "acs:oss:*:174649585760xxxx:examplebucket/*"
        ]
      }, {
        "Effect": "Allow",
        "Action": [
            "oss:ListObjects",
            "oss:GetObject"
        ],
        "Principal": [
            "27737962156157xxxx",
            "20214760404935xxxx"
        ],
        "Resource": [
            "acs:oss:*:174649585760xxxx:examplebucket"
        ],
        "Condition": {
            "StringLike": {
                "oss:Prefix": [
                    "*"
                ]
            }
        }
      }
    ]
}
```

示例二：授予指定用户拥有某个Bucket下指定目录的只读权限

以下示例用于授权UID为 `20214760404935xxxx` 的RAM用户拥有目标存储空间examplebucket下 `hangzhou/2020` 和 `shanghai/2015` 目录的只读权限。

```
{
     "Version": "1",
    "Statement": [
        {
            "Action": [
                "oss:GetObject",
                "oss:GetObjectAcl",
                "oss:GetObjectVersion",
                "oss:GetObjectVersionAcl"
            ],
            "Effect": "Allow",
            "Principal": [
                "20214760404935xxxx"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket/hangzhou/2020/*",
                "acs:oss:*:174649585760xxxx:examplebucket/shanghai/2015/*"
            ]
        },
        {
            "Action": [
                "oss:ListObjects",
                "oss:ListObjectVersions"
            ],
            "Condition": {
                "StringLike": {
                    "oss:Prefix": [
                        "hangzhou/2020/*",
                        "shanghai/2015/*"
                    ]
                }
            },
            "Effect": "Allow",
            "Principal": [
                "20214760404935xxxx"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
            ]
        }
    ]
}
```

示例三：授予匿名用户仅拥有列举某个Bucket下所有文件的权限

以下示例用于授予匿名用户仅拥有列举目标存储空间examplebucket下所有文件的权限：

```
{
    "Version": "1",
    "Statement": [
    {
        "Action": [
            "oss:ListObjects",
            "oss:ListObjectVersions"
        ],
        "Effect": "Allow",
        "Principal": [
            "*"
        ],
        "Resource": [
            "acs:oss:*:174649585760xxxx:examplebucket"
        ]
    }
  ]
}
```

示例四：授予仅允许指定VPC ID且满足指定IP地址段的用户访问某个Bucket资源的权限

以下示例用于授予仅允许指定VPC ID为 `t4nlw426y44rd3iq4****` ，且满足该VPC ID范围内IP地址段为 `192.168.0.0/16` 的用户访问目标存储空间examplebucket的权限：

```
{
    "Version": "1",
        "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
                ],
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
                ],
            "Condition": {
                "StringNotEquals": {
                    "acs:SourceVpc": [
                        "vpc-t4nlw426y44rd3iq4****"
                        ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
                ],
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
                ],
            "Condition": {
                "StringEquals": {
                    "acs:SourceVpc": [
                        "vpc-t4nlw426y44rd3iq4****"
                        ]
                },
                "NotIpAddress": {
                    "acs:SourceIp": [
                        "192.168.0.0/16"
                        ]
                }
            }
        }
    ]
}
```

示例五：授予仅允许指定VPC ID且满足指定公网地址的用户访问某个Bucket资源的权限

以下示例用于授予仅允许指定VPC ID为 `t4nlw426y44rd3iq4****` ，且满足公网IP地址为 `192.0.2.0` 的用户访问目标存储空间examplebucket的权限：

```
{
    "Version": "1",
        "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
                ],
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
                ],
            "Condition": {
                "StringNotLike": {
                    "acs:SourceVpc": [
                        "vpc-*"
                        ]
                },
                "NotIpAddress": {
                    "acs:SourceIp": [
                        "192.0.2.0"
                        ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
                ],
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
                ],
            "Condition": {
                "StringLike": {
                    "acs:SourceVpc": [
                        "vpc-*"
                        ]
                },
                "StringNotEquals": {
                    "acs:SourceVpc": [
                        "vpc-t4nlw426y44rd3iq4****"
                        ]
                }
            }
        }
    ]
}
```

示例六：拒绝非指定VPC ID的用户访问某个Bucket资源的权限

以下示例用于拒绝VPC ID不为 `t4nlw426y44rd3iq4****` 的用户访问目标存储空间examplebucket的权限：

```
{
    "Version": "1",
        "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
                ],
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
                ],
            "Condition": {
                "StringNotEquals": {
                    "acs:SourceVpc": [
                        "vpc-t4nlw426y44rd3iq4****"
                        ]
                }
            }
        }
    ]
}
```

示例七：拒绝非指定公网IP地址的用户访问某个Bucket资源的权限

以下示例用于拒绝公网IP地址不为 `192.0.2.0` 的用户访问目标存储空间examplebucket的权限：

```
{
    "Version": "1",
        "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
                ],
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
                ],
            "Condition": {
                "StringNotLike": {
                    "acs:SourceVpc": [
                        "vpc-*"
                        ]
                },
                "NotIpAddress": {
                    "acs:SourceIp": [
                        "192.0.2.0"
                        ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
                ],
            "Principal": [
                "*"
            ],
            "Resource": [
                "acs:oss:*:174649585760xxxx:examplebucket"
                ],
            "Condition": {
                "StringLike": {
                    "acs:SourceVpc": [
                        "vpc-*"
                        ]
                }
            }
        }
    ]
}
```

7.1.5.3. 教程示例：通过Bucket Policy限制公网访问OSS

Bucket Policy是阿里云对象存储OSS推出的针对存储空间（Bucket）的授权策略，您可以通过Bucket Policy限制通过公网访问您指定的OSS资源。

场景描述

企业A在华东1（杭州）地域创建了名为examplebucket的存储空间，examplebucket的目录*examplefolder*下存放了大量的企业内部资料。企业A不希望指定的合作伙伴以RAM用户的方式通过公网访问*examplefolder*下的资源。

为满足企业A的以上需求，您可以通过策略语法的方式配置Bucket Policy。

操作方式

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击**examplebucket**。
3. 单击**文件管理**页签，然后单击**授权**。
4. 在**策略语法**页面，单击**编辑**，然后输入以下Policy。

```
{
    "Version": "1",
    "Statement": [{
        "Effect": "Deny",
        "Action": [
            "oss:RestoreObject",
            "oss:ListObjects",
            "oss:AbortMultipartUpload",
            "oss:PutObjectAcl",
            "oss:GetObjectAcl",
            "oss:ListParts",
            "oss:DeleteObject",
            "oss:PutObject",
            "oss:GetObject",
            "oss:GetVodPlaylist",
            "oss:PostVodPlaylist",
            "oss:PublishRtmpStream",
            "oss:ListObjectVersions",
            "oss:GetObjectVersion",
            "oss:GetObjectVersionAcl",
            "oss:RestoreObjectVersion"
        ],
        "Principal": [
            <! --以下为RAM用户的UID。-->
            "26642223584287****",
            "27658173539067****",
            "24430533117653****"
        ],
        "Resource": [
            <! --137918634953****为examplebucket拥有者的UID。-->
            "acs:oss:*:137918634953****:examplebucket/examplefolder/*"
        ],
        "Condition": {
            "StringNotEquals": {
                "acs:SourceVpc": [
                    "vpc-*"
                ]
            }
        }
    }, {
        "Effect": "Deny",
        "Action": [
            "oss:ListObjects",
            "oss:GetObject"
        ],
        "Principal": [
            "26642223584287****",
            "27658173539067****",
            "24430533117653****"
        ],
        "Resource": [
            "acs:oss:*:137918634953****:examplebucket"
        ],
        "Condition": {
            "StringLike": {
                "oss:Prefix": [
                    "examplefolder/*"
                ]
            },
            "StringNotEquals": {
                "acs:SourceVpc": [
                    "vpc-*"
                ]
            }
        }
    }]
}
```

5. 单击**保存**。

相关文档

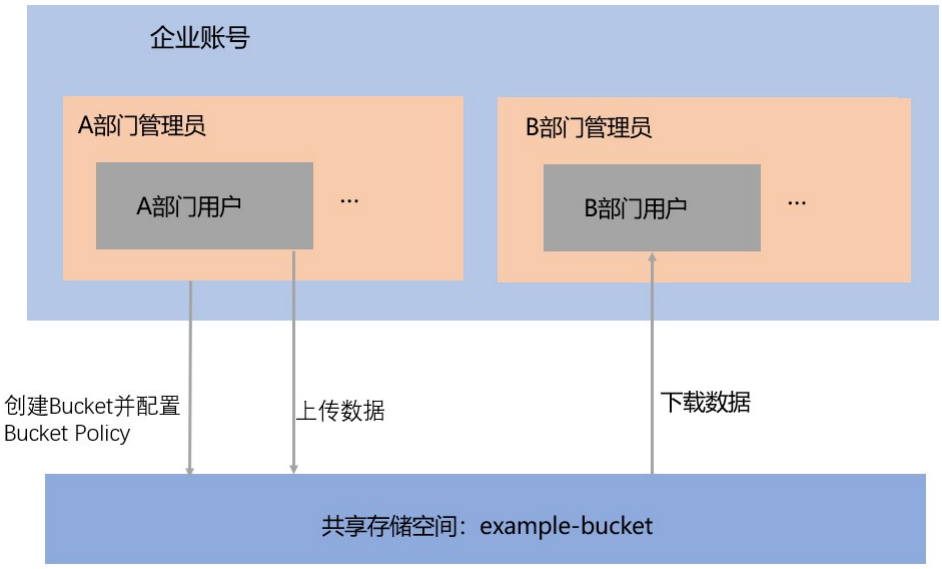
- 企业不同部门或项目之间需要共享数据。如果您希望其他部门的用户可以下载本部分的共享数据，禁止写入和删除数据，以降低共享数据被误删、篡改的风险，请参见基于Bucket Policy实现跨部门数据共享的教程示例。
- 如果您需要对同账号以及跨账号下的RAM用户，或者匿名用户等授予访问或管理Bucket资源的不同权限，例如只读、读写权限等，请参见Bucket Policy常见示例。

7.1.5.4. 教程示例：基于Bucket Policy实现跨部门数据共享

企业不同部门或项目之间需要共享数据，本部门允许其他部门的用户下载共享数据，禁止写入和删除数据，以降低共享数据被误删、篡改的风险。

背景信息

部门A将存储在example-bucket存储空间（Bucket）中的数据共享给部门B的用户，并允许部门B的用户下载数据。本文介绍如何以最小权限原则对共享数据进行权限控制。在本场景下两个部门的管理员、用户与共享存储空间之间的逻辑关系如下图所示：



实现流程

在此场景下，A部门的管理员可以通过配置Bucket Policy，授予B部门用户允许下载，但禁止写入和删除共享数据的权限。具体配置流程如下：

- 步骤1：创建Bucket

  A部门管理员创建一个用于共享数据的Bucket（example-bucket）。

- 步骤2：授予上传权限

  A部门管理员通过Bucket Policy，授权本部门用户上传共享数据的权限。

- 步骤3：授予允许下载、禁止写入和删除的权限

  A部门管理员通过配置Bucket Policy，授权B部门用户允许下载、禁止写入和删除的权限。

- 步骤4：上传数据

  A部门用户上传数据到example-bucket。

- 步骤5：验证权限

  验证B部门用户对共享数据仅拥有下载权限、但无法对共享数据进行写入和删除操作。

前提条件

- 企业账号已通过访问控制RAM创建A部门管理员用户及其普通用户、B部门管理员用户及其普通用户。

  有关创建RAM用户的详情，请参见创建RAM用户。

- 已获取RAM用户UID。

  有关查看RAM用户基本信息，例如UID等，详情请参见查看RAM用户基本信息。

- 已为RAM用户授权

  在创建管理员用户时，A部门的管理员由于要执行创建Bucket、配置Bucket Policy等操作，因此需要管理员所属用户组具有AliyunOSSFullAccess权限。有关RAM用户授权详情，请参见为RAM用户授权。

步骤1：创建Bucket

A部门管理员创建Bucket的具体步骤如下：

1. 使用A部门管理员账号登录OSS管理控制台。
2. 单击**Bucket列表**，之后单击**创建Bucket**。
3. 在**创建Bucket**页面配置Bucket参数。

   结合本示例场景，请将Bucket命名为example-bucket。有关配置Bucket其他各项参数的详情，请参见创建存储空间。

4. 单击**确定**。

步骤2：授予上传权限

部门A的管理员为本部门的用户配置允许上传共享数据的权限，具体步骤如下：

1. 单击步骤1中创建的example-bucket。
2. 单击**权限管理 > Bucket授权策略 > 设置**。
3. 在**授权**对话框，单击**新增授权**。
4. 在**新增授权**页面，结合本示例场景配置各项参数说明如下：

配置项	说明
授权资源	选择**整个Bucket**，授权策略对整个Bucket生效。

配置项	说明
授权用户	选择**选择子账号**。 您可以从下拉菜单中选择部门A管理员账号下允许上传数据的用户账号。若部门A管理员下的用户账号较多时，您也可以直接在搜索框输入子账号名称，搜索支持模糊匹配。
授权操作	选择**读/写**。 表示被授权用户可以对指定资源执行读取和写入操作。

5. 单击**确定**。
   此时，允许A部门用户上传数据的权限配置完成。

步骤3：授予允许下载、禁止写入和删除的权限

部门A的管理员为部门B的用户配置允许下载共享数据的权限。具体步骤如下：

1. 单击步骤1中创建的example-bucket。
2. 单击**权限管理 > Bucket授权策略 > 设置**。
3. 在**授权**对话框，单击**新增授权**。
4. 在**新增授权**页面，结合本示例场景配置各项参数说明如下：

配置项	说明
授权资源	选择**整个Bucket**，授权策略对整个Bucket生效。
授权用户	选择**其他账号**：输入被授权下载共享数据B用户的UID。
授权操作	选择**只读**。 表示对共享资源example-bucket中的数据拥有查看、列举及下载权限，但无法对共享数据执行写入和删除操作。

5. 单击**确定**。
   此时，部门A的管理员为部门B的用户配置授予允许下载、禁止写入和删除的权限已完成。

步骤4：上传数据

A部门用户上传数据到example-bucket。具体步骤如下：

1. 使用A部门用户账号登录OSS管理控制台。
2. 单击**Bucket列表**，之后单击目标Bucket（example-bucket）。
3. 单击**文件管理 > 上传文件**。
4. 在**上传文件**页面，设置上传文件的参数。
   选择将文件上传至example-bucket当前目录。有关上传文件的ACL、以及文件上传方式的说明请参见上传文件。
5. 在**上传任务**页面等待任务完成，之后关闭对话框。
   此时，A用户已完成将数据上传至共享Bucket。

步骤5：验证权限

权限授予成功后，通过OSS控制台验证B部门用户对共享数据仅拥有下载权限、但无法对共享数据进行写入和删除操作。

1. 使用B部门用户账号登录OSS管理控制台。
2. 单击**Bucket列表**，之后单击目标Bucket（example-bucket）。
3. 单击**文件管理**。
4. 验证权限。
   i. 验证B部门用户对共享数据的下载权限。
      单击example-bucket中任意目标文件右侧的**更多 > 下载**。
      - 下载失败，表示下载权限配置失败，请检查权限配置是否正确。
      - 下载成功，表示下载权限配置成功。
   ii. 验证B部门用户对共享数据的上传权限。
      参考步骤4上传文件。
      - 上传失败，表示上传权限配置成功。
      - 上传成功，表示上传权限配置失败，请检查权限配置是否正确。
   iii. 验证B部门用户对共享数据的删除权限。
      单击example-bucket中任意目标文件右侧的**更多 > 删除**。
      - 删除失败，表示删除权限配置成功。
      - 删除成功，表示删除权限配置失败，请检查权限配置是否正确。

7.1.5.5. 教程示例：基于Bucket Policy实现跨账号访问OSS

阿里云OSS的资源默认都是私有的，若您希望您的合作伙伴可以访问您的OSS资源，可以通过Bucket Policy授予合作伙伴访问Bucket的权限。

背景信息

公司A希望其合作公司B可以访问自己的OSS资源，但又不方便开放RAM用户给B公司。此时，A公司可以通过Bucket Policy授予合作伙伴访问Bucket的权限。B公司账号获得授权之后，可以在控制台添加A公司OSS资源的访问路径进行访问。

添加Bucket Policy

- B公司账号
  i. 登录RAM访问控制台，创建RAM用户。
     详细配置方法请参见创建RAM用户。
  ii. 在RAM访问控制台，单击**用户**。
  iii. 单击刚刚创建的RAM用户的用户名，查看并记录RAM用户的UID号。
- A公司账号
  i. 登录阿里云OSS控制台。
  ii. 单击**Bucket列表**，之后单击目标Bucket名称。
  iii. 单击**文件管理 > 授权 > 新增授权**。

  > **说明** 您也可以单击**权限管理 > Bucket授权策略 > 设置 > 新增授权**。

  iv. 在**新增授权**的对话框，填写授权策略。其中，**授权用户**选择**其他账号**，并填写B公司RAM用户的UID号。其他参数请参考Bucket Policy。
  v. 单击**确定**。

登录RAM用户账号并添加访问路径

Bucket Policy添加完成之后，您还需要登录B公司RAM用户账号，添加A公司的Bucket访问路径。配置步骤如下：

1. 通过RAM账号登录链接登录B公司RAM用户。
2. 打开OSS控制台。
3. 单击左侧菜单栏**我的访问路径**后的加号（+），添加A公司授权访问的Bucket路径信息。
   - **区域**：下拉选择A公司授权访问的Bucket所在地域。
   - **Bucket**：输入A公司授权访问的Bucket名称。
   - **访问路径**：添加A公司授权访问的Bucket访问路径。例如，仅允许访问根目录*abc*下*test*目录，则添加*abc/test*。

您也可以获取AccessKey，并通过AccessKey使用ossutil、ossbrowser等工具访问被授权的Bucket。

7.1.6. RAM Policy

7.1.6.1. RAM Policy概述

RAM Policy是基于用户的授权策略。您可以使用RAM Policy控制用户可以访问您名下哪些资源的权限。

背景信息

- RAM Policy语法和结构

  RAM Policy包含版本号（Version）和授权语句（Statement）。每条授权语句又包含授权效力（Effect）、操作（Action）、资源（Resource）以及限制条件（Condition，可选项）。有关权限策略的语法和结构的详情，请参见权限策略语法和结构。

  其中OSS中Version、Statement以及Effect的应用规则与RAM相同。OSS的Action、Resource以及Condition说明请参见：

  - OSS Action分类
  - OSS Resource说明
  - OSS Condition说明
- OSS常用的权限策略
  - AliyunOSSFullAccess：为RAM用户授予OSS的完全管理权限。
  - AliyunOSSReadOnlyAccess：为RAM用户授予OSS的只读访问权限。
- OSS访问控制方式

  有关OSS包含的访问控制方式的更多信息，请参见访问控制概述。

OSS Action分类

Action分为Service级别操作、Bucket级别操作以及Object级别的操作。

- Service级别

API	Action	接口描述
GetService (ListBuckets)	oss:ListBuckets	列举请求者拥有的所有Bucket。

- Bucket级别

API	Action	接口描述
PutBucket	oss:PutBucket	创建Bucket。
GetBucket (ListObjects)	oss:ListObjects	列举Bucket中所有Object的信息。
GetBucketInfo	oss:GetBucketInfo	查看Bucket相关信息。
GetBucketLocation	oss:GetBucketLocation	查看Bucket位置信息。

API	Action	接口描述
PutBucketVersioning	oss:PutBucketVersioning	设置指定Bucket的版本控制状态。
GetBucketVersioning	oss:GetBucketVersioning	获取指定Bucket的版本控制状态。
GetBucketVersions(ListObjectVersions)	oss:ListObjectVersions	列出Bucket中包括删除标记（Delete Marker）在内的所有Object的版本信息。
PutBucketAcl	oss:PutBucketAcl	设置或修改Bucket的访问权限（ACL）。
GetBucketAcl	oss:GetBucketAcl	设置或修改Bucket的ACL。
DeleteBucket	oss:DeleteBucket	删除某个Bucket。
PutBucketLogging	oss:PutBucketLogging	开启Bucket日志转存功能。
GetBucketLogging	oss:GetBucketLogging	查看Bucket日志转存配置。
DeleteBucketLogging	oss:DeleteBucketLogging	关闭Bucket日志转存功能。
PutBucketWebsite	oss:PutBucketWebsite	设置Bucket为静态网站托管模式并设置其跳转规则（RoutingRule）。
GetBucketWebsite	oss:GetBucketWebsite	查看Bucket的静态网站托管状态以及跳转规则。
DeleteBucketWebsite	oss:DeleteBucketWebsite	关闭Bucket的静态网站托管模式以及跳转规则。
PutBucketReferer	oss:PutBucketReferer	设置Bucket的防盗链。
GetBucketReferer	oss:GetBucketReferer	查看Bucket的防盗链（Referer）相关配置。
PutBucketLifecycle	oss:PutBucketLifecycle	设置Bucket的生命周期规则。
GetBucketLifecycle	oss:GetBucketLifecycle	查看Bucket的生命周期规则（Lifecycle）。
DeleteBucketLifecycle	oss:DeleteBucketLifecycle	删除Bucket的生命周期规则。
ListMultipartUploads	oss:ListMultipartUploads	列举所有执行中的Multipart Upload事件，即已经初始化但还未完成（Complete）或者还未中止（Abort）的Multipart Upload事件。
PutBucketCors	oss:PutBucketCors	设置指定Bucket的跨域资源共享CORS（Cross-Origin Resource Sharing）规则。
GetBucketCors	oss:GetBucketCors	获取指定Bucket当前的跨域资源共享CORS规则。
DeleteBucketCors	oss:DeleteBucketCors	关闭指定Bucket对应的跨域资源共享CORS功能并清空所有规则。
PutBucketPolicy	oss:PutBucketPolicy	设置指定Bucket的授权策略（Policy)。
GetBucketPolicy	oss:GetBucketPolicy	获取指定Bucket的授权策略。
DeleteBucketPolicy	oss:DeleteBucketPolicy	删除指定Bucket的授权策略。
PutBucketTags	oss:PutBucketTagging	添加或修改指定Bucket的标签。
GetBucketTags	oss:GetBucketTagging	获取Bucket的标签。
DeleteBucketTags	oss:DeleteBucketTagging	删除Bucket的标签。
PutBucketEncryption	oss:PutBucketEncryption	配置Bucket的加密规则。
GetBucketEncryption	oss:GetBucketEncryption	获取Bucket的加密规则。
DeleteBucketEncryption	oss:DeleteBucketEncryption	删除Bucket的加密规则。
PutBucketRequestPayment	oss:PutBucketRequestPayment	设置请求者付费模式。
GetBucketRequestPayment	oss:GetBucketRequestPayment	获取请求者付费模式配置信息。
PutBucketReplication	oss:PutBucketReplication	设置Bucket的数据复制规则。
GetBucketReplication	oss:GetBucketReplication	获取Bucket已设置的数据复制规则。
DeleteBucketReplication	oss:DeleteBucketReplication	停止Bucket的数据复制并删除Bucket的复制配置。
GetBucketReplicationLocation	oss:GetBucketReplicationLocation	获取可复制到的目标Bucket的所在地域。
GetBucketReplicationProgress	oss:GetBucketReplicationProgress	获取Bucket的数据复制进度。
PutBucketInventory	oss:PutBucketInventory	配置Bucket的清单（Inventory）规则。
GetBucketInventory	oss:GetBucketInventory	查看Bucket中指定的清单任务。
ListBucketInventory	oss:GetBucketInventory	批量获取Bucket中所有清单任务。
DeleteBucketInventory	oss:DeleteBucketInventory	删除Bucket中指定的清单任务。
PutStyle	oss:PutStyle	设置图片样式。

API	Action	接口描述
GetStyle	oss:GetStyle	获取图片样式。
ListStyle	oss:ListStyle	列举图片样式。
DeleteStyle	oss:DeleteStyle	删除图片样式。

- Object级别

API	Action	接口描述
PutObject	oss:PutObject	上传文件（Object）。
PostObject	oss:PutObject	通过HTML表单上传的方式将Object上传到指定Bucket。
AppendObject	oss:PutObject	以追加写的方式上传Object。
InitiateMultipartUpload	oss:PutObject	在使用Multipart Upload模式传输数据前，通知OSS初始化一个分片上传（Multipart Upload）事件。
UploadPart	oss:PutObject	根据指定的Object名和uploadId来分块（Part）上传数据
CompleteMultipartUpload	oss:PutObject	在将所有数据Part都上传完成后，需调用此接口来完成整个Object的分片上传。
AbortMultipartUpload	oss:AbortMultipartUpload	取消MultipartUpload事件并删除对应的Part数据。
PutSymlink	oss:PutObject	为OSS的目标文件（TargetObject）创建软链接（Symlink）。
GetObject	oss:GetObject	获取某个Object。
HeadObject	oss:GetObject	取某个Object的元信息。
GetObjectMeta	oss:GetObject	获取Object的元数据信息，包括该Object的ETag、Size、LastModified信息。
SelectObject	oss:GetObject	对目标文件执行SQL语句，返回执行结果。
GetSymlink	oss:GetObject	获取目标文件的软链接。
DeleteObject	oss:DeleteObject	删除某个Object。
DeleteMultipleObjects	oss:DeleteObject	删除同一个Bucket中的多个Object。
CopyObject	oss:GetObject,oss:PutObject	拷贝同一地域下相同或不同Bucket之间的Object。
UploadPartCopy	oss:GetObject,oss:PutObject	在UploadPart请求的基础上增加一个请求头x-oss-copy-source来调用UploadPartCopy接口，实现从一个已存在的Object中拷贝数据来上传一个Part。
ListParts	oss:ListParts	列举指定Upload ID所属的所有已经上传成功的Part。
PutObjectACL	oss:PutObjectAcl	修改Bucket下某个Object的ACL。
GetObjectACL	oss:GetObjectAcl	获取Bucket下某个Object的ACL。
RestoreObject	oss:RestoreObject	解冻归档类型（Archive）或冷归档（Cold Archive）的Object。
PutObjectTagging	oss:PutObjectTagging	设置或更新Object的标签（Tagging）信息。
GetObjectTagging	oss:GetObjectTagging	获取Object的标签信息。
DeleteObjectTagging	oss:DeleteObjectTagging	删除指定Object的标签信息。
GetObject（请求参数中指定versionId）	oss:GetObjectVersion	下载指定版本Object。
PutObjectACL（请求参数中指定versionId）	oss:PutObjectAcl	修改Bucket下指定版本Object的ACL。
GetObjectACL（请求参数中指定versionId）	oss:GetObjectVersionAcl	获取Bucket下指定版本Object的ACL。
RestoreObject（请求参数中指定versionId）	oss:RestoreObjectVersion	解冻归档类型（Archive）或冷归档（Cold Archive）的指定版本Object。
DeleteObject（请求参数中指定versionId）	oss:DeleteObjectVersion	删除指定版本Object。
PutObjectTagging（请求参数中指定versionId）	oss:PutObjectVersionTagging	设置或更新指定版本Object的标签（Tagging）信息。
GetObjectTagging（请求参数中指定versionId）	oss:GetObjectVersionTagging	获取指定版本Object的标签信息。
DeleteObjectTagging（请求参数中指定versionId）	oss:DeleteObjectVersionTagging	删除指定版本Object的标签信息。
PutLiveChannel	oss:PutLiveChannel	通过RTMP协议上传音视频数据前，必须先调用该接口创建一个LiveChannel。
ListLiveChannel	oss:ListLiveChannel	列举指定的LiveChannel。

API	Action	接口描述
DeleteLiveChannel	oss:DeleteLiveChannel	删除指定的LiveChannel。
PutLiveChannelStatus	oss:PutLiveChannelStatus	在启用（enabled）和禁用（disabled）两种状态之间进行切换。
GetLiveChannelInfo	oss:GetLiveChannel	获取指定LiveChannel的配置信息。
GetLiveChannelStat	oss:GetLiveChannelStat	获取指定LiveChannel的推流状态信息。
GetLiveChannelHistory	oss:GetLiveChannelHistory	获取指定LiveChannel的推流记录。
PostVodPlaylist	oss:PostVodPlaylist	为指定的LiveChannel生成一个点播用的播放列表。
GetVodPlaylist	oss:GetVodPlaylist	查看指定LiveChannel在指定时间段内推流生成的播放列表。
ImgSaveAs	oss:PostProcessTask	保存处理后的图片至指定Bucket。

OSS Resource说明

在OSS中，Resource指代某个具体资源或者某些资源，支持通配符星号（*）。单个RAM Policy允许包含多个Resource。

Resource规则：`acs:oss:{region}:{bucket_owner}:{bucket_name}/{object_name}`

针对Bucket级别的Resource设置，不需要在`{bucket_name}`之后添加正斜线（/）以及`{object_name}`，即`acs:oss:{region}:{bucket_owner}:{bucket_name}`。region字段当前仅支持设置为通配符星号（*）。

OSS Condition说明

Condition代表Policy授权的条件。OSS支持的Condition如下：

Condition	说明
acs:SourceIp	指定普通IP网段，支持通配符星号（*）。
acs:UserAgent	指定HTTP User-Agent头。 类型：字符串
acs:CurrentTime	请求到达OSS服务端的时间。 格式：ISO8601
acs:SecureTransport	请求的协议类型。如果请求是HTTP协议，则为HTTP，如果请求是HTTPS协议，则为HTTPS。
oss:Prefix	用于ListObjects请求时，列举指定前缀的Object。
oss:Delimiter	用于ListObjects请求时，对Object名字进行分组的字符。
acs:AccessId	请求中携带的AccessId。
oss:BucketTag	存储空间标签（BucketTag）。 单个BucketTag可以作为一个Condition。当设置多个BucketTag时，需在每个BucketTag前加上`oss:BucketTag/`，组成多个Condition。
acs:MFAPresent	是否启用了多因素认证MFA（Multi-factor authentication）。 取值： • true：已启用多因素认证。 • false：未启用多因素认证。
oss:ExistingObjectTag	请求的Object已存在标签。 单个ObjectTag可以作为一个Condition。当设置多个ObjectTag时，需在每个ObjectTag前加上`oss:ExistingObjectTag/`，组成多个Condition。 主要针对GetObject、HeadObject等读取文件接口以及PutObjectTagging、GetObjectTagging等ObjectTagging接口。
oss:RequestObjectTag	请求中携带的对象标签。 单个ObjectTag可以作为一个Condition。当设置多个ObjectTag时，需在每个ObjectTag前加上`oss:RequestObjectTag/`，组成多个Condition。 主要针对PutObject、PostObject等写入文件接口以及PutObjectTagging、GetObjectTagging等ObjectTagging接口。

常见示例

您可以使用RAM Policy实现不同场景下的用户权限策略。更多信息，请参见RAM Policy常见示例。

7.1.6.2. RAM Policy常见示例

通过RAM Policy，您可以集中管理您的用户（例如员工、系统或应用程序），以及控制用户可以访问您名下哪些资源的权限，例如授权RAM用户列举并读取某个存储空间（Bucket）的资源。

为RAM用户授权自定义的权限策略

1. 创建自定义权限策略。

您可以结合实际使用场景，选用下文列举的常见授权示例，然后通过脚本配置方式创建自定义权限策略。具体操作，请参见创建自定义权限策略。

有关权限策略中包含版本号（Version）和授权语句（Statement），以及授权语句中包含的授权效力（Effect）、操作（Action）、资源（Resource）以及限制条件（Condition，可选项）等信息，请参见RAM Policy概述。

> **注意** 在OSS中，Resource支持使用通配符星号（*）来指代某类具体的资源。Resource的格式为 `acs:oss:{region}:{bucket_owner}:{bucket_name}/{object_name}` 。例如，当Resource为 `acs:oss:*:*:mybucket/*` ，表示指代mybucket下的所有资源。当Resource为 `acs:oss:*:*:mybucket/abc*.txt` ，表示指代mybucket下前缀为abc且格式为.txt的所有文件。

2. 为RAM用户授权自定义权限策略。

   为RAM用户授权步骤1中创建好的RAM Policy。具体操作，请参见为RAM用户授权。

示例一：授予RAM用户对某个Bucket的完全控制权限

以下示例为授权RAM用户对名为 `mybucket` 的Bucket拥有完全控制的权限。

> **警告** 对于移动应用来说，授予用户对Bucket的完全控制权限有极高风险，应尽量避免。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "oss:*",
            "Resource": [
                "acs:oss:*:*:mybucket",
                "acs:oss:*:*:mybucket/*"
            ]
        }
    ]
}
```

示例二：拒绝RAM用户删除某个bucket下指定的多个文件的权限

以下示例为拒绝RAM用户删除名为 `mybucket` 的Bucket下前缀为abc且格式为.txt的所有文件：

```
{
  "Version": "1",
  "Statement": [
        {
         "Effect": "Deny",
         "Action": [
           "oss:DeleteObject"
         ],
         "Resource": [
           "acs:oss:*:*:mybucket/abc*.txt"
         ]
      }
    ]
}
```

示例三：授予RAM用户列举并读取某个Bucket下所有资源的权限

- 授予RAM用户通过OSS SDK或OSS命令行工具列举并读取某个Bucket资源的权限

  以下示例为授予RAM用户通过OSS SDK或OSS命令行工具列举并读取名为 `mybucket` 的Bucket下所有资源的权限：

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "oss:ListObjects",
            "Resource": "acs:oss:*:*:mybucket"
        },
        {
            "Effect": "Allow",
            "Action": "oss:GetObject",
            "Resource": "acs:oss:*:*:mybucket/*"
        }
    ]
}
```

- 授予RAM用户通过OSS控制台列举并读取某个Bucket的资源

  以下示例为授予RAM用户通过OSS控制台列举并读取名为 `mybucket` 的Bucket下所有资源的权限：

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "oss:ListBuckets",
                      "oss:GetBucketStat",
                      "oss:GetBucketInfo",
                      "oss:GetBucketTagging",
                      "oss:GetBucketLifecycle",
                      "oss:GetBucketWorm",
                      "oss:GetBucketVersioning",
                      "oss:GetBucketAcl"
                      ],
            "Resource": "acs:oss:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:ListObjects",
                "oss:GetBucketAcl"
            ],
            "Resource": "acs:oss:*:*:mybucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:GetObject",
                "oss:GetObjectAcl"
            ],
            "Resource": "acs:oss:*:*:mybucket/*"
        }
    ]
}
```

示例四：拒绝RAM用户删除某个Bucket的权限

以下示例用于拒绝RAM用户删除名为 `mybucket` 的Bucket的权限：

```
{
  "Version": "1",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": "oss:*",
          "Resource": [
              "acs:oss:*:*:mybucket",
              "acs:oss:*:*:mybucket/*"
          ]
      },
        {
         "Effect": "Deny",
         "Action": [
           "oss:DeleteBucket"
         ],
         "Resource": [
           "acs:oss:*:*:mybucket"
         ]
     }
   ]
}
```

示例五：授予RAM用户访问某个Bucket下多个目录的权限

假设用于存放照片的Bucket为 `mybucket` ，该Bucket下有一些目录，代表照片的拍摄地，每个拍摄地目录下还包含了年份子目录。

```
mybucket[Bucket]
  ├── beijing
  │   ├── 2014
  │   └── 2015
  ├── hangzhou
  │   ├── 2013
  │   ├── 2014
  │   └── 2015
  └── qingdao
      ├── 2014
      └── 2015
```

您希望授予RAM用户访问 `mybucket/hangzhou/2014/` 和 `mybucket/hangzhou/2015/` 目录的只读权限。目录级别的授权属于授权的高级功能，根据使用场景不同，授权策略的复杂程度也不同，以下几种场景可供参考。

- 授予RAM用户仅拥有读取目录 `mybucket/hangzhou/2014/` 和 `mybucket/hangzhou/2015/` 中文件内容的权限

  由于RAM用户知道文件的完整路径，建议直接使用完整的文件路径来读取目录下的文件内容。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "oss:GetObject"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket/hangzhou/2014/*",
                "acs:oss:*:*:mybucket/hangzhou/2015/*"
            ]
        }
    ]
}
```

- 授予RAM用户使用OSS命令行工具访问目录 `mybucket/hangzhou/2014/` 和 `mybucket/hangzhou/2015/` 并列举目录中文件的权限

  RAM用户不清楚目录中有哪些文件，可以使用OSS命令行工具或API直接获取目录信息，此场景下需要添加 `ListObjects` 权限。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "oss:GetObject"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket/hangzhou/2014/*",
                "acs:oss:*:*:mybucket/hangzhou/2015/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:ListObjects"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket"
            ],
            "Condition":{
                "StringLike":{
                    "oss:Prefix": [
                        "hangzhou/2014/*",
                    "hangzhou/2015/*"
                     ]
                }
            }
        }
    ]
}
```

- 授予RAM用户使用OSS控制台访问目录的权限

  使用OSS控制台访问目录 `mybucket/hangzhou/2014/` 和 `mybucket/hangzhou/2015/` 时，RAM用户可以从根目录开始，逐层进入要访问的目录。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "oss:ListBuckets",
                      "oss:GetBucketStat",
                      "oss:GetBucketInfo",
                      "oss:GetBucketTagging",
                      "oss:GetBucketLifecycle",
                      "oss:GetBucketWorm",
                      "oss:GetBucketVersioning",
                      "oss:GetBucketAcl"
                      ],
            "Resource": [
                "acs:oss:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:GetObject",
                "oss:GetObjectAcl"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket/hangzhou/2014/*",
                "acs:oss:*:*:mybucket/hangzhou/2015/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:ListObjects"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket"
            ],
            "Condition": {
                "StringLike": {
                    "oss:Delimiter": "/",
                    "oss:Prefix": [
                        "",
                        "hangzhou/",
                        "hangzhou/2014/*",
                        "hangzhou/2015/*"
                    ]
                }
            }
        }
    ]
}
```

示例六：拒绝RAM用户删除某个Bucket下任意文件的权限

以下示例用于拒绝RAM用户删除名为 `mybucket` 的存储空间下任意文件的权限：

```
{
  "Version": "1",
  "Statement": [
        {
         "Effect": "Deny",
         "Action": [
           "oss:DeleteObject"
         ],
         "Resource": [
           "acs:oss:*:*:mybucket/*"
         ]
      }
    ]
}
```

示例七：拒绝RAM用户访问指定标签Object的权限

以下为添加Deny策略，用于拒绝RAM用户访问存储空间examplebucket下对象标签为status:ok以及key1:value1的Object的权限。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "oss:GetObject"
            ],
            "Resource": [
                "acs:oss:*:1746495857602745:examplebucket/*"
            ],
            "Condition": {
                "StringEquals": {
                    "oss:ExistingObjectTag/status":"ok",
                    "oss:ExistingObjectTag/key1":"value1"
                }
            }
        }
    ]
}
```

示例八：授予RAM用户通过特定的IP地址访问OSS的权限

- 在 `Allow` 授权中增加IP地址限制

  以下示例为在 `Allow` 授权中增加IP地址限制，授予RAM用户仅允许通过 `192.168.0.0/16` 、 `172.12.0.0/16` 两个IP地址段读取名为 `mybucket` Bucket下所有资源的权限。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "oss:ListBuckets",
                      "oss:GetBucketStat",
                      "oss:GetBucketInfo",
                      "oss:GetBucketTagging",
                      "oss:GetBucketAcl"
                      ],
            "Resource": [
                "acs:oss:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:ListObjects",
                "oss:GetObject"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket",
                "acs:oss:*:*:mybucket/*"
            ],
            "Condition":{
                "IpAddress": {
                    "acs:SourceIp": ["192.168.0.0/16", "172.12.0.0/16"]
                }
            }
        }
    ]
}
```

- 在 `Deny` 授权中增加IP地址限制

  以下示例为在 `Deny` 授权中增加IP地址限制，拒绝源IP地址不在 `192.168.0.0/16` 范围内的RAM用户对OSS执行任何操作。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "oss:ListBuckets",
                      "oss:GetBucketStat",
                      "oss:GetBucketInfo",
                      "oss:GetBucketTagging",
                      "oss:GetBucketAcl"
                      ],
            "Resource": [
                "acs:oss:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:ListObjects",
                "oss:GetObject"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket",
                "acs:oss:*:*:mybucket/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": "oss:*",
            "Resource": [
                "acs:oss:*:*:*"
            ],
            "Condition":{
                "NotIpAddress": {
                    "acs:SourceIp": ["192.168.0.0/16"]
                }
            }
        }
    ]
}
```

说明 由于权限策略的鉴权规则是Deny优先，所以访问者从 `192.168.0.0/16` 以外的IP地址访问mybucket中的内容时，OSS会提示没有权限。

示例九：通过RAM或STS服务向其他用户授权

通过RAM或STS服务向其他用户授权的场景说明如下：

- 将阿里云账号名下的 `mybucket` 和 `mybucket/file*` 资源授权给相应的用户。
- 允许其他用户执行GetBucketAcl、GetBucket、PutObject、GetObject和DeleteObject操作。
- Condition中的条件表示UserAgent为java-sdk，源IP地址为 `192.168.0.1` 的鉴权被允许，此时被授权的用户拥有相关资源的访问权限。
- 仅列举Prefix为foo的Object。

符合上述场景的RAM Policy示例如下：

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": [
                "oss:GetBucketAcl",
                "oss:ListObjects"
            ],
            "Resource": [
                "acs:oss:*:177530505652XXXX:mybucket"
            ],
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "acs:UserAgent": "java-sdk",
                    "oss:Prefix": "foo"
                },
                "IpAddress": {
                    "acs:SourceIp": "192.168.0.1"
                }
            }
        },
        {
            "Action": [
                "oss:PutObject",
                "oss:GetObject",
                "oss:DeleteObject"
            ],
            "Resource": [
                "acs:oss:*:177530505652XXXX:mybucket/file*"
            ],
            "Effect": "Allow",
            "Condition": {
               "StringEquals": {
                    "acs:UserAgent": "java-sdk"
                },
                "IpAddress": {
                    "acs:SourceIp": "192.168.0.1"
                }
            }
        }
    ]
}
```

7.1.6.3. 教程示例：使用RAM Policy控制OSS的访问权限

本教程示例详细演示了如何使用RAM Policy控制用户对OSS存储空间（Bucket）、文件夹以及文件夹下文件（Object）的访问权限。

背景信息

RAM Policy是基于用户的授权策略。通过设置RAM Policy，您可以集中管理您的用户（例如员工、系统或应用程序），以及控制用户可以访问您名下哪些资源的权限，例如限制您的用户只拥有对某一个Bucket的读权限。

RAM Policy为JSON格式。各字段定义如下：

- Statement：授权语句，一个权限策略可以有多条授权语句。
- Effect：授权效力，包括允许（Allow）和拒绝（Deny）两种。

> **说明** 当权限策略中既有Allow又有Deny的授权语句时，遵循Deny优先的原则。

- Action：对具体资源的操作权限。

如果您选择使用RAM Policy，建议您通过官方工具RAM策略编辑器快速生成RAM策略。

相比于RAM Policy，Bucket Policy支持在控制台直接进行图形化配置操作，并且Bucket拥有者可以直接进行授权访问。更多信息，请参见使用Bucket Policy授权其他用户访问OSS资源。

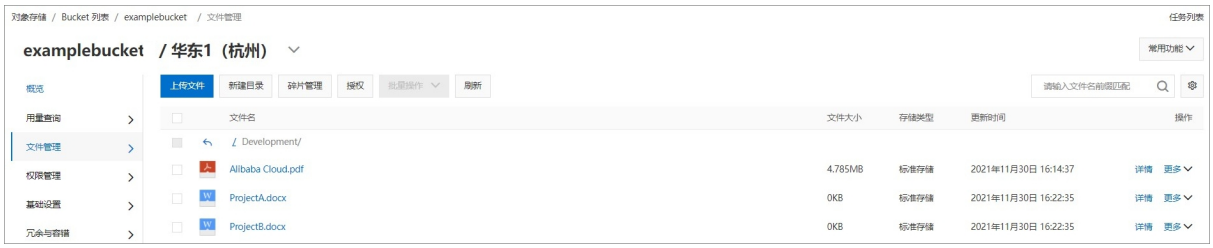
存储空间和文件夹的基本概念

阿里云OSS的数据模型为扁平型结构，所有文件都直接隶属于其对应的存储空间。因此，OSS缺少文件系统中类似于目录与子文件夹的层次结构。但是，您可以在OSS控制台上模拟文件夹层次结构。在该控制台中，您可以按文件夹对相关文件进行分组、分类和管理，如下图所示。

OSS提供使用键值（key）对格式的分布式对象存储服务。您可以根据其唯一的key（对象名）检索对象的内容。例如，名为*examplebucket*的存储空间有三个文件夹，分别为*Development*、*Marketing*和*Private*，以及一个对象*oss-dg.pdf*。

- 在创建*Development*文件夹时，控制台会创建一个key为 `Development/` 的对象，文件夹的key包括分隔符 `/` 。
- 当您将名为*ProjectA.docx*的对象上传到*Development* 文件夹中时，控制台会上传该对象并将其key设置为 `Development/ProjectA.docx` 。

  在该key中， `Development` 为前缀，而 `/` 为分隔符。您可以从存储空间中获取具有特定前缀和分隔符的所有对象的列表。在控制台中，单击*Development* 文件夹时，控制台会列出文件夹中的对象，如下图所示。



> **说明** 当控制台列举examplebucket存储空间中的 *Development*文件夹时，它会向OSS发送一个用于指定前缀 `Development` 和分隔符 `/` 的请求。因此，存储空间examplebucket有三个对象，其key分别为 `Development/Alibaba Cloud.pdf` 、 `Development/ProjectA.docx` 及 `Development/ProjectB.docx` 。

在本教程开始之前，您还需要了解根级存储空间内容的概念。假设*examplebucket*存储空间包含以下对象：

- Development/Alibaba Cloud.pdf
- Development/ProjectA.docx
- Development/ProjectB.docx
- Marketing/data2020.xlsx
- Marketing/data2021.xlsx
- Private/2017/images.zip
- Private/2017/promote.pptx
- oss-dg.pdf

这些对象的key构建了一个以*Development*、*Marketing*和*Private*作为根级文件夹并以oss-dg.pdf作为根级对象的逻辑层次结构。当您单击OSS控制台中的存储空间名时，控制台会将一级前缀和一个分隔符，例如*Development/*、*Marketing/*和*Private/*显示为根级文件夹。对象*oss-dg.pdf*没有前缀，因此显示为根级别项。

OSS的请求和响应逻辑

在授予RAM用户相关权限之前，您需要了解单击某个存储空间的名字时控制台向OSS发送请求、OSS返回响应，以及控制台如何解析该响应的逻辑。

- 请求某个存储空间

  单击*examplebucket*存储空间时，控制台会将GetBucket (ListObjects)请求发送至OSS。

  - 请求示例

```
GET /?prefix=&delimiter=/ HTTP/1.1
Host: examplebucket.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:DNrnx7xHk3sgysx7I8U9I9IY****
```

  此请求包括prefix和delimiter参数，其中prefix的值为空字符串，delimiter的值为正斜线（/）。

- 响应示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 7 Aug 2020 08:43:27 GMT
Content-Type: application/xml
Content-Length: 712
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns=¡±http://doc.oss-cn-hangzhou.aliyuncs.com¡±>
<Name>examplebucket</Name>
<Prefix></Prefix>
<Marker></Marker>
<MaxKeys>100</MaxKeys>
<Delimiter>/</Delimiter>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>oss-dg.pdf</Key>
        ...
    </Contents>
   <CommonPrefixes>
        <Prefix>Development</Prefix>
   </CommonPrefixes>
      <CommonPrefixes>
        <Prefix>Marketing</Prefix>
   </CommonPrefixes>
      <CommonPrefixes>
        <Prefix>Private</Prefix>
   </CommonPrefixes>
</ListBucketResult>
```

- 控制台解析

  控制台会解析此结果并显示如下的根级别项：

- 请求存储空间下的某个文件夹

  单击*Development/*文件夹，控制台会将GetBucket (ListObjects)请求发送至OSS。此请求包括以下参数：

  - 请求示例

```
GET /?prefix=Development/&delimiter=/ HTTP/1.1
Host: examplebucket.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:DNrnx7xHk3sgysx7I8U9I9IY****
```

  此请求包括prefix和delimiter参数，其中prefix的值为 `Development/` ，delimiter的值为正斜线（/）。

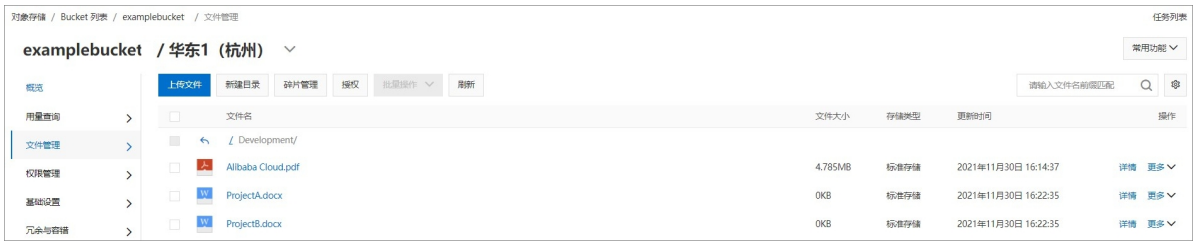
- 响应示例

  作为响应，OSS返回以指定前缀开头的key：

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 7 Aug 2020 08:43:27 GMT
Content-Type: application/xml
Content-Length: 712
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns=¡±http://doc.oss-cn-hangzhou.aliyuncs.com¡±>
<Name>examplebucket</Name>
<Prefix>Development/</Prefix>
<Marker></Marker>
<MaxKeys>100</MaxKeys>
<Delimiter>/</Delimiter>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>ProjectA.docx</Key>
        ...
    </Contents>
    <Contents>
        <Key>ProjectB.docx</Key>
        ...
    </Contents>
    <Contents>
        <Key>Alibaba Cloud.pdf</Key>
        ...
    </Contents>
</ListBucketResult>
```

- 控制台解析

  控制台会解析此结果并显示如下的key：



场景示例

假设您是目标存储空间 `examplebucket` 的Owner，且该Bucket下所有的文件或目录读写权限ACL默认为私有。现在，您希望授予RAM用户Anne访问该Bucket下文件夹 `Development` 及其子文件夹和文件的读写权限，RAM用户Leo访问文件夹 `Marketing` 及其子文件夹和文件的只读权限，以及当前阿里云账号下的所有RAM用户均无权访问文件夹 `Private` 的权限。

步骤一：创建存储空间并上传文件

1. 创建存储空间*examplebucket*。
   i. 使用阿里云账号登录OSS控制台。
   ii. 创建名为*examplebucket*的存储空间。具体操作，请参见创建存储空间。
2. 创建目录*Development*、*Marketing*和*Private*。具体操作，请参见创建目录。
3. 按如下要求将文件上传至指定路径。
   - 将oss-dg.pdf文件上传至examplebucket的根目录。
   - 将文件Alibaba Cloud.pdf、ProjectA.docx以及ProjectB.docx上传至Development目录。
   - 将文件data2020.xlsx和data2021.xlsx上传至Marketing目录。
   - 将文件images.zip和promote.pptx上传至Private目录。

   具体操作，请参见上传文件。

步骤二：创建RAM用户Anne和Leo

通过RAM控制台创建RAM用户Anne和Leo为例。有关创建RAM用户的详情，请参见创建RAM用户。

步骤三：授予RAM用户Anne拥有文件夹Development的读写权限

1. 创建自定义权限策略*AllowAnneToReadAndWriteFolderDevelopment*，并授予RAM用户Anne拥有文件夹Development及文件夹下所有文件的读写权限。
   i. 在左侧导航栏的**权限管理**菜单下，单击**权限策略**。
   ii. 单击**创建权限策略**。

iii. 在**新建自定义权限策略**页面，**策略名称**填写为*AllowAnneToReadAndWriteFolderDevelopment*，配置模式选择**脚本配置**，**策略内容**配置如下：

```
{
    "Version":"1",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":[
                "oss:ListObjects"
            ],
            "Resource":[
                "acs:oss:*:*:examplebucket"
            ],
            "Condition":{
                "StringEquals":{
                    "oss:Prefix":[
                        "Development",
                        "Development/*"
                    ]
                }
            }
        },
        {
            "Effect":"Allow",
            "Action":[
                "oss:GetObject",
                "oss:PutObject",
                "oss:GetObjectAcl"
            ],
            "Resource":[
                "acs:oss:*:*:examplebucket/Development/*"
            ]
        }
    ]
}
```

iv. 单击**确定**。

2. 为RAM用户Anne添加自定义权限策略*AllowAnneToReadAndWriteFolderDevelopment*。具体操作，请参见为RAM用户授权。

步骤四：授予RAM用户Leo拥有文件夹Marketing的只读权限

参见步骤三创建自定义权限策略*AllowLeoToReadAndWriteFolderMarketing*，并授予RAM用户Leo只读访问文件夹Marketing及文件夹下所有文件的权限。其策略内容配置如下：

```
{
    "Version":"1",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":[
                "oss:ListObjects"
            ],
            "Resource":[
                "acs:oss:*:*:examplebucket"
            ],
            "Condition":{
                "StringEquals":{
                    "oss:Prefix":[
                        "Marketing",
                        "Marketing/*"
                    ]
                }
            }
        },
        {
            "Effect":"Allow",
            "Action":[
                "oss:GetObject",
                "oss:GetObjectAcl"
            ],
            "Resource":[
                "acs:oss:*:*:examplebucket/Marketing/*"
            ]
        }
    ]
}
```

步骤5：拒绝当前阿里云账号下的所有RAM用户访问Private文件夹

1. 创建用户组并添加用户组成员。

   创建用户组的具体操作，请参见创建用户组。用户组创建完成后，将当前阿里云账号下的所有RAM用户添加到该用户组。具体操作，请参见为用户组添加RAM用户。

2. 创建自定义权限策略*DenyAllRamToAccessFolderPrivate*，并授予当前阿里云账号下的所有RAM用户拒绝访问Private文件夹的权限。

   i. 在左侧导航栏的**权限管理**菜单下，单击**权限策略**。

   ii. 单击**创建权限策略**。

iii. 在**新建自定义权限策略**页面，**策略名称**填写为*DenyAllRamToAccessFolderPrivate*，配置模式选择**脚本配置**，**策略内容**配置如下：

```
{
    "Version":"1",
    "Statement":[
        {
            "Effect":"Deny",
            "Action":[
                "oss:*"
            ],
            "Resource":[
                "acs:oss:*:*:examplebucket/Private/*"
            ],
            "Condition":{
            }
        },
        {
            "Effect":"Deny",
            "Action":[
                "oss:ListObjects"
            ],
            "Resource":[
                "acs:oss:*:*:*"
            ],
            "Condition":{
                "StringEquals":{
                    "oss:Prefix":[
                        "Private/",
                        "Private/*"
                    ]
                }
            }
        }
    ]
}
```

iv. 单击**确定**。

3. 为用户组添加自定义权限策略*DenyAllRamToAccessFolderPrivate*。具体操作，请参见为用户组授权。

   添加权限策略后，用户组中的任何RAM用户都不能访问您存储空间 `examplebucket` 中的文件夹 `Private` ，且当RAM用户请求列举 `Private` 文件夹下的 `Private/2017/images.zip` 、 `Private/2017/promote.pptx` 文件时，OSS也将返回错误响应。

7.1.6.4. 教程示例：基于RAM角色实现跨账号访问OSS

阿里云OSS的资源默认都是私有的，若您希望您的合作伙伴可以访问您的OSS资源，可以通过RAM角色实现跨账号访问。

背景信息

某公司A希望其合作公司B可以访问自己名下OSS内的数据，但又不方便开放RAM账号给公司B。此时，公司A可创建一个RAM角色，并授权RAM角色OSS的访问权限。公司B可用RAM用户扮演这个RAM角色，实现跨账号访问的目的。

步骤1：公司A创建RAM角色并授予OSS访问权限

公司A需要先创建一个拥有OSS访问权限的RAM角色，提供给公司B的RAM用户扮演。

1. 公司A登录RAM控制台。
2. 选择**身份管理 > 角色**，单击**创建角色**。
3. 在**创建角色**面板的**选择类型**步骤，选择可信实体类型为**阿里云账号**，之后单击**下一步**。
4. 在**配置角色**步骤，配置如下参数：
   - **角色名称**：填写角色名称。本示例设为： `admin-oss` 。
   - **备注**：角色的备注信息，选填。本示例置空此项。
   - **选择云账号**：选择**其他云账号**，并配置公司B的阿里云账号UID。本示例设为： `17464958576******` 。
5. 单击**完成**。
6. 在**创建完成**步骤，单击**为角色授权**。
7. 在添加权限面板选择策略为**系统策略**，找到并单击AliyunOSSReadOnlyAccess（只读访问对象存储服务（OSS）的权限）策略。在右侧**已选择**列表看到该策略后，单击**确定**。

   AliyunOSSReadOnlyAccess为访问OSS下所有Bucket的策略，若您仅希望您的客户访问部分Bucket或部分目录，可创建自定义策略。更多信息，请参见RAM Policy概述。

如果您需要指定该RAM角色只能被指定的RAM用户扮演，可修改RAM角色的信任策略。操作步骤，请参见修改RAM角色的信任策略。

步骤2：公司B创建RAM用户并授予允许扮演RAM角色的权限

公司B需要创建一个可以扮演角色的RAM用户，用以扮演公司A创建的RAM角色。

1. 公司B登录RAM控制台。
2. 选择**身份管理 > 用户**，单击**创建用户**。
3. 在**创建用户**页面填写**登录名称**和**显示名称**，并选中**控制台登录**，根据您的需要配置控制台的登录信息。
4. 单击**确定**。
5. 在跳转的**用户信息**列表选中刚创建的用户，单击**添加权限**。

   注意保存RAM用户信息，防止丢失。
6. 在添加权限面板选择策略为**系统策略**，找到并单击AliyunSTSAssumeRoleAccess（调用STS服务AssumeRole接口的权限）策略。在右侧**已选择**列表看到该策略后，单击**确定**。

步骤3：公司B的RAM用户登录控制台并扮演公司A的RAM角色

公司B需使用刚创建的RAM用户登录阿里云控制台，并切换身份为公司A创建的RAM角色。

1. 公司B使用RAM用户登录阿里云控制台，步骤请参见RAM用户登录阿里云控制台。
2. 鼠标移至右上角的头像上，单击**切换身份**。
3. 在**角色切换**页面输入RAM角色信息后，单击**切换**。

   RAM角色信息如下：
   - **企业别名/默认域名**：填写公司A的企业别名或默认域名，详情请参见基本概念。

     本示例以默认域名为例，输入默认域名为 `1178810717******.onaliyun.com` ， `1178810717******` 为公司A阿里云账号的UID。
   - **角色名**：输入公司A创建的RAM角色 `admin-oss` 。
4. 打开OSS管理控制台即可管理公司A的OSS资源。

更多参考

您也可以通过Bucket Policy实现以上需求。操作步骤，请参见教程示例：基于Bucket Policy实现跨账号访问OSS。

7.2. 数据容灾

7.2.1. 同城冗余存储

OSS采用多可用区（AZ）机制，将用户的数据分散存放在同一地域（Region）的3个可用区。当某个可用区不可用时，仍然能够保障数据的正常访问。OSS同城冗余存储提供99.9999999999%（12个9）的数据设计持久性以及99.995%的服务可用性。

使用场景

同城冗余存储能够提供机房级容灾能力。当发生断网、断电或者灾难事件导致某个机房不可用时，OSS仍能继续提供强一致性的服务。整个故障切换过程用户无感知、业务不中断、数据不丢失，满足关键业务系统对于恢复时间目标（RTO）以及恢复点目标（RPO）等于0的强需求。

注意事项

- 支持地域

  仅华南1（深圳）、华北2（北京）、华东1（杭州）、华东2（上海）、中国（香港）、新加坡以及印度尼西亚（雅加达）地域支持同城冗余存储。
- 计费说明

  相对于本地冗余存储，同城冗余存储会产生更高的存储费用。更多信息，请参见OSS产品定价。
- 创建方式

  目前仅支持在创建Bucket时开启同城冗余存储。对于已创建的Bucket，您可以利用迁移工具（例如ossimport、在线迁移服务等），将现有Bucket中的数据迁移到已开启同城冗余存储的Bucket中。

支持的存储类型

OSS的同城冗余存储目前支持标准存储类型、低频访问存储类型。这两种存储类型的各项对比指标如下：

对比指标	标准存储类型	低频访问存储类型
数据设计持久性	99.9999999999%（12个9）	99.9999999999%（12个9）
服务可用性	99.995%	99.50%
对象最小计量单位	无	64 KB
最短存储时间	无	30天
数据取回费用	无	按实际获取的数据量收取，单位GB
数据访问	实时访问，毫秒延迟	实时访问，毫秒延迟
图片处理	支持	支持

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击**创建Bucket**。
3. 在**创建Bucket**面板，按如下说明配置各项参数。

参数	是否必选	描述
Bucket名称	是	填写Bucket名称。命名规则如下： ○ 所选定的存储空间名称在阿里云OSS的所有现有存储空间名称中必须具有唯一性。 ○ 只能包括小写字母、数字和短划线（-）。 ○ 必须以小写字母或者数字开头和结尾。 ○ 长度必须在3~63字符之间。
地域	是	Bucket的数据中心。 如需通过ECS内网访问OSS，请选择与您ECS相同的地域。更多信息，请参见 OSS访问域名使用规则。 **说明** 如果您创建的存储空间位于中国内地，您需要在实名登记页面提交您的相关信息完成实名登记。

参数	是否必选	描述
存储类型	是	Bucket的存储类型。 ○ **标准存储**：提供高可靠、高可用、高性能的对象存储服务，能够支持频繁的数据访问。适用于各种社交、分享类的图片、音视频应用、大型网站、大数据分析等业务场景。 ○ **低频访问存储**：提供高持久性、较低存储成本的对象存储服务。有最低存储时间（30天）和最小计量单位（64 KB）要求。支持数据实时访问，访问数据时会产生数据取回费用，适用于较低访问频率（平均每月访问频率1到2次）的业务场景。 ○ **归档存储**：提供高持久性、极低存储成本的对象存储服务。有最低存储时间（60天）和最小计量单位（64 KB）要求。数据需解冻（约1分钟）后访问，解冻会产生数据取回费用。适用于数据长期保存的业务场景，例如档案数据、医疗影像、科学资料、影视素材等。 ○ **冷归档存储**：提供高持久性的对象存储服务，费用在四种存储类型最低。有最低存储时间（180天）和最小计量单位（64 KB）要求。数据需解冻后访问，解冻时间根据数据大小和选择的解冻模式决定，解冻会产生数据取回费用。适用于需要超长时间存放的极冷数据，例如因合规要求需要长期留存的数据、大数据及人工智能领域长期积累的原始数据、影视行业长期留存的媒体资源、在线教育行业的归档视频等业务场景。 **说明** 冷归档存储类型支持以下地域：华北1（青岛）、华北2（北京）、华北3（张家口）、华东1（杭州）、华东2（上海）、华南1（深圳）、西南1（成都）、华北6（乌兰察布）、中国香港、澳大利亚（悉尼）、新加坡、美国（硅谷）、德国（法兰克福）、马来西亚（吉隆坡）、印度尼西亚（雅加达）、印度（孟买）、阿联酋（迪拜）。请联系技术支持申请使用。 有关存储类型的更多信息，请参见存储类型介绍。
HDFS服务	否	如果您希望通过JindoSDK访问OSS实现数据湖场景，请先开启HDFS服务。 **注意** ○ 仅华东1（杭州）、华东2（上海）、华南1（深圳）、华北2（北京）和华北 3（张家口）地域支持开启HDFS服务，请联系技术支持申请试用。HDFS服务开启后不支持关闭，请谨慎操作。 ○ 归档以及冷归档存储类型Bucket不支持开启HDFS服务。
同城冗余存储	否	Bucket的数据容灾类型。 ○ 启用：开启后，将以同城冗余ZRS的方式存储您的OSS数据。同城冗余采用多可用区（AZ）机制，将您的数据冗余存储在同一地域（Region）的3个可用区。可支持单个可用区（机房）整体故障时（如断电、火灾等），仍然能够保障数据的正常访问。 **注意** 仅华南1（深圳）、华北2（北京）、华东1（杭州）、华东2（上海）、中国（香港）、新加坡以及印度尼西亚（雅加达）地域支持开启同城冗余存储。此外，同城冗余存储的费用较高，且开启后不支持关闭，请谨慎操作。 有关同城冗余存储的更多信息，请参见同城冗余存储。 ○ 关闭：以本地冗余LRS的方式存储您的OSS数据。本地冗余将您的数据冗余存储在同一个可用区的不同存储设备上，可支持两个存储设备并发损坏时，仍维持数据不丢失，可正常访问。
版本控制	否	选择是否开通版本控制功能。 ○ **开通**：开通Bucket版本控制功能后，针对数据的覆盖和删除操作将会以历史版本的形式保存下来。当您在错误覆盖或者删除Object后，能够将Bucket中存储的Object恢复至任意时刻的历史版本。更多详情请参见版本控制介绍。 ○ **不开通**：不开通版本控制功能，则不保存覆盖或删除的数据。
读写权限	是	选择Bucket的读写权限。 ○ **私有（private）**：只有该存储空间的拥有者可以对该存储空间内的文件进行读写操作，其他人无法访问该存储空间内的文件。 ○ **公共读（public-read）**：只有该存储空间的拥有者可以对该存储空间内的文件进行写操作，任何人（包括匿名访问者）可以对该存储空间中的文件进行读操作。 **警告** 互联网上任何用户都可以对该Bucket内文件进行访问，这有可能造成您数据的外泄以及费用激增，请谨慎操作。 ○ **公共读写（public-read-write）**：任何人（包括匿名访问者）都可以对该存储空间内文件进行读写操作。 **警告** 互联网上任何用户都可以对该Bucket内的文件进行访问，并且向该Bucket写入数据。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益。除特殊场景外，不建议您配置公共读写权限。

参数	是否必选	描述
服务端加密	否	选择是否开启服务器端加密。 ○ **服务端加密方式**：选择Object的加密方式。 ■ **无**：不启用服务器端加密。 ■ **OSS完全托管**：使用OSS托管的密钥进行加密。OSS会为每个Object使用不同的密钥进行加密，作为额外的保护，OSS会使用定期轮转的主密钥对加密密钥本身进行加密。 ■ **KMS**：使用KMS默认托管的CMK或指定CMK ID进行加解密操作。 使用KMS加密方式前，需要开通KMS服务。具体步骤，请参见开通KMS服务。 ○ **加密算法**：目前仅支持AES256。 ○ **加密密钥**：**服务端加密方式**选择**KMS**时，可配置此项。参数说明如下： ■ **alias/acs/oss**：使用默认托管的CMK生成不同的密钥来加密不同的Object，并且在Object被下载时自动解密。 ■ CMK ID：使用指定的CMK生成不同的密钥来加密不同的Object，并将加密Object的CMK ID记录到Object的元信息中，具有解密权限的用户下载Object时会自动解密。选择指定的CMK ID前，您需在KMS管理控制台创建一个与Bucket处于相同地域的普通密钥或外部密钥。详情请参见导入密钥材料。
实时日志查询	否	如果您希望在不付费的情况下实时查询最近7天的OSS访问日志，请选择**开通**。 有关实时日志查询的更多信息，请参见实时日志查询。 如果您不需要进行实时日志查询，请保持**不开通**的默认配置。
定时备份	否	如果您希望定时备份您的OSS数据，请选择**开通**。此时，OSS将自动创建备份计划，并由混合云备份HBR执行备份频率为每天备份一次OSS数据，备份文件保存一周的任务。 **注意** 如果HBR未开通或未授权HBR访问OSS，则无法创建定时备份计划。更多信息，请参见设置定时备份。 如果您不需要定时备份您的OSS数据，请保持**不开通**的默认配置。
分层命名空间	否	如果您希望快速地对目录或文件进行重命名，请开启分层命名空间。 **注意** 仅允许在创建Bucket时开启分层命名空间功能。注意开启后不支持关闭，且不支持部分功能。请参见分层命名空间查看分层命名空间不支持的功能列表。

4. 单击**确定**。

使用阿里云SDK

以下仅列举常见SDK在创建Bucket时开启同城冗余存储的代码示例。关于其他SDK在创建Bucket时开启同城冗余存储的代码示例，请参见SDK简介。

Node.js

使用命令行工具ossutil

关于使用ossutil在创建Bucket时开启同城冗余存储的具体步骤， 请参见mb（创建存储空间）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucket。

7.2.2. 跨区域复制

跨区域复制（Cross-Region Replication）是跨不同OSS数据中心（地域）的存储空间（Bucket）自动、异步（近实时）复制文件（Object），它会将Object的创建、更新和删除等操作从源存储空间复制到不同区域的目标存储空间。

使用场景

跨区域复制功能满足Bucket跨区域容灾或用户数据复制的需求。目标Bucket中的Object是源Bucket中Object的精确副本，它们具有相同的Object名、版本信息、元数据以及内容，例如创建时间、拥有者、用户定义的元数据、Object ACL、Object内容等。您可以通过配置跨区域复制规则实现以下场景需求。

- 合规性要求

  虽然OSS默认对每个存储的Object在物理盘上有多份副本，但合规性要求所规定的数据需要跨一定距离保存一份副本。通过跨区域复制，可以在远距离的OSS数据中心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Object时的延迟，可以在地理位置与用户较近的OSS数据中心维护Object副本。

- 数据备份与容灾

  您对数据的安全性和可用性有极高的要求，对所有写入的数据，都希望在另一个数据中心显式地维护一份副本，以备发生特大灾难（如地震、海啸等）导致一个OSS数据中心损毁时，还能启用另一个OSS数据中心的备份数据。

- 数据迁移

  由于业务原因，需要将数据从OSS的一个数据中心迁移到另一个数据中心。

- 操作原因

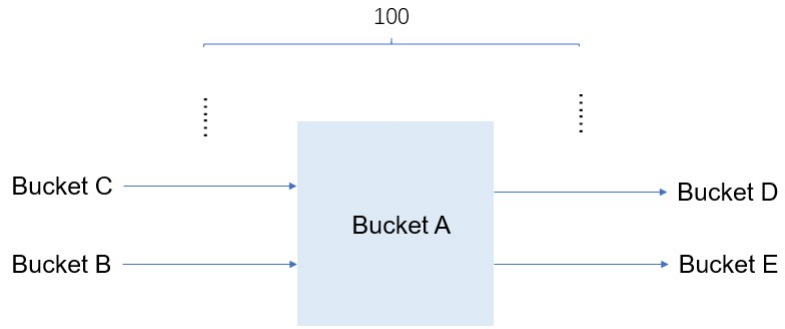
  您在两个不同数据中心中拥有分析同一组Object的计算集群。您可以选择在两个不同区域中维护Object副本。

功能特性

跨区域复制支持特性如下：

- 不同地域Bucket之间的数据同步

  源Bucket中的数据可以同步到多个目标Bucket。单个Bucket关联的复制规则数量不能超过100条。这些规则中，该Bucket既可以作为源Bucket，也可以作为目标Bucket。



  如果您的业务场景涉及更大数量的复制规则，请联系技术支持。

- 实时同步数据

  对于数据的增加、删除、修改能够实时监控并同步到目标地域Bucket。对于2 MB以下文件，能够做到分钟级别信息同步，保证两边数据的最终一致。

- 历史数据迁移

  迁移历史数据，让源Bucket中历史数据也能进行同步，形成相同的两份数据。

- 实时获取同步进度

  能够针对实时同步数据展示最近同步的时间节点，针对历史数据的迁移展示迁移的百分比。

- 版本控制

  对同时处于开启版本控制状态的源Bucket和目标Bucket，保证其数据版本的最终一致性。如果数据同步方式为写（增、改）同步，则源Bucket指定版本删除的操作不会同步到目标Bucket，源Bucket创建的删除标记会同步到目标Bucket。

- 传输加速

  支持通过传输加速功能提高中国内地各地域与非中国内地各地域之间进行跨区域复制时的数据传输速度。传输加速功能详情请参见传输加速。

- 复制加密数据

  支持复制未加密的Object和使用SSE-KMS、SSE-OSS方式进行服务器端加密的Object。详情请参见跨区域复制结合服务器端加密。

- 配置事件通知以及实时日志查询

  您可以通过以下两种方式准确获取跨区域复制过程中源Bucket以及目标Bucket内Object的新增、更新、删除、覆盖等变化情况。

  - 在设置事件通知规则中将事件类型配置为 `ObjectReplication:ObjectCreated` 、 `ObjectReplication:ObjectRemoved` 以及 `ObjectReplication:ObjectModified` 。详情请参见事件通知概述。
  - 在OSS管理控制台开启实时日志查询，获取Object操作的统计信息。详情请参见查询实时日志。

注意事项

- 相关费用
  - 进行跨区域复制时，OSS会根据复制文件产生的流量收取跨区域复制流量费用。计费方式，请参见跨区域复制流量费用。
  - 每同步1个Object，OSS会计算请求次数并收取请求费用。计费方式，请参见请求费用。
  - 若开启传输加速功能，会额外产生传输加速费用。计费方式，请参见传输加速费用。
- 复制时间

  跨区域复制采用异步（近实时）复制，数据复制到目标Bucket需要一定的时间，通常几分钟到几小时不等，取决于数据的大小。

使用限制

- 地域限制
  - 菲律宾（马尼拉）地域暂不支持跨区域复制。关于OSS数据中心所在的地域，请参见访问域名和数据中心。
  - 中国内地各地域与非中国内地各地域之间进行跨区域复制时，必须开启传输加速功能。
  - 目前仅在以下地域进行跨区域复制时支持设置标签规则：
    - 源地域为华东1（杭州），目标地域为除华东1（杭州）以外的任意地域。
    - 源地域为澳大利亚（悉尼），目标地域为除中国内地和澳大利亚（悉尼）以外的任意地域。
- 操作限制
  - 仅允许同时处于非版本化或启用版本控制状态的两个Bucket开启数据同步。
  - 处于同步状态下的两个Bucket不允许改变其版本控制状态。
  - 对于处于同步状态的两个Bucket，由于您可以同时操作这两个Bucket，源Bucket复制过去的Object可能存在覆盖目标Bucket中同名Object的风险。
  - 源Bucket中的数据可以同步到多个目标Bucket。单个Bucket关联的复制规则数量不能超过100条。这些规则中，该Bucket既可以作为源Bucket，也可以作为目标Bucket。如果您的业务场景涉及更大数量的复制规则，请联系技术支持。

使用OSS控制台

1. 登录OSS管理控制台。
2. 在左侧导航栏，单击**Bucket列表**，然后单击任意待开启跨区域复制的Bucket。
3. 在左侧导航栏，选择**冗余与容错 > 跨区域复制**。在**跨区域复制**区域，单击**设置**。
4. 单击**跨区域复制**，然后在**跨区域复制**面板配置以下参数。

参数	说明
源Bucket地域	显示您当前Bucket所在地域。
源Bucket	显示您当前Bucket名称。
目标地域	选择目标Bucket所在地域。
目标Bucket	选择开启数据同步的目标Bucket。
加速类型	加速类型仅支持**传输加速**。传输加速可用于提升在中国内地与非中国内地之间跨区域复制时的传输速度。开启传输加速功能，OSS还会额外收取传输加速费用。计费方式，请参见传输加速费用。
数据同步对象	选择需要同步的源数据。 ○ **全部文件进行同步**：将该Bucket内所有的Object同步到目标存储空间。 ○ **指定文件名前缀进行同步**：将该Bucket内指定前缀的Object同步到目标Bucket。最多可以添加10个前缀。
Object标签	同步拥有指定标签的Object到目标Bucket。设置方法为选中**设置规则**后添加标签（键-值对），最多可添加10个标签。 要设置该参数，必须满足以下条件： ○ 已设置Object标签。具体操作，请参见设置对象标签。 ○ 源Bucket和目标Bucket均已开启版本控制。 ○ 数据同步策略为**增/改 同步**。 ○ 源地域为华东1（杭州），目标地域为除华东1（杭州）以外的任何一个地域；或者源地域为澳大利亚（悉尼），目标地域为除中国内地和澳大利亚（悉尼）以外的任何一个地域。
数据同步策略	选择数据同步的方式。 ○ **增/改 同步**：将该Bucket内Object新增和更新操作同步到目标Bucket。 ○ **增/删/改 同步**：将该Bucket内Object的新增、更新、删除操作同步到目标Bucket。 如果某个Object是通过分片上传的方式上传至源Bucket，则每个分片的上传操作都会同步至目标Bucket。最终，对所有分片执行CompleteMultipartUpload后生成的Object，也会被同步到目标Bucket。 有关跨区域复制结合版本控制的同步行为说明，请参见跨区域复制结合版本控制。
同步历史数据	选择是否同步跨区域复制规则生效前源Bucket中已有的历史数据。 ○ **同步**：将历史数据同步至目标Bucket。 **注意** 同步历史数据时，从源Bucket复制的Object可能会覆盖目标Bucket中同名的Object。为避免这部分文件丢失，建议您对源Bucket和目标Bucket开启版本控制。 ○ **不同步**：仅同步跨区域复制规则生效后上传或更新的Object。
KMS加密目标对象	在源Object或者目标Bucket使用了KMS托管密钥加密方式（即SSE-KMS，指定CMK ID）的情况下，要将Object复制到目标Bucket，则必须选中**KMS加密目标对象**，并配置以下参数： ○ **使用的KMS密钥**：为目标Object指定加密的KMS密钥。 您需要提前在KMS平台创建一个与目标Bucket相同地域的KMS密钥。具体操作，请参见创建密钥。 ○ **授权角色**：授权一个RAM角色对目标Object执行KMS加密操作。 ■ **新建角色**：新建RAM角色对目标Object执行KMS加密，角色名称格式为 `kms-replication-源Bucket名称-目标Bucket名称` 。 ■ **AliyunOSSRole**：使用AliyunOSSRole角色对目标Object执行KMS加密。若您之前未创建AliyunOSSRole角色，当您选择此项时，OSS将自动创建AliyunOSSRole角色。 **说明** ○ 您可以通过HeadObject和GetBucketEncryption分别查询源Object和目标Bucket的加密状态。 ○ 有关跨区域复制结合服务器端加密详情，请参见跨区域复制结合服务器端加密。

5. 单击**确定**。
   - 当跨区域复制规则创建完成后，不允许对此规则进行编辑或删除。
   - 同步任务会在跨区域复制规则配置完成的3~5分钟后启动。您可以在源Bucket管理页面选择**冗余与容错 > 跨区域复制**查看同步进度。
   - 由于Bucket间的跨区域复制采用异步（近实时）复制，数据复制到目标Bucket需要的时间取决于数据的大小，通常几分钟到几小时不等。

使用阿里云SDK

以下仅列举常见SDK的开启跨区域复制示例。关于其他SDK的开启跨区域复制示例，请参见SDK简介。

Python Go

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.AddBucketReplicationRequest;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 指定数据要复制到的目标Bucket。
        String targetBucketName = "yourTargetBucketName";
        // 指定目标Bucket对应的Endpoint。例如https://oss-cn-beijing.aliyuncs.com
        String targetBucketLocation = "https://oss-cn-beijing.aliyuncs.com";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            AddBucketReplicationRequest request = new AddBucketReplicationRequest(bucketName);
            request.setTargetBucketName(targetBucketName);
            request.setTargetBucketLocation(targetBucketLocation);
            // 默认同步历史数据。此处设置为false，表示禁止同步历史数据。
            request.setEnableHistoricalObjectReplication(false);
            // 指定授权OSS进行数据复制的角色名称。如果指定使用SSE-KMS加密目标对象，则必须指定该元素。
            //request.setSyncRole("yourRole");
            // 指定OSS是否复制通过SSE-KMS加密创建的对象。
            //request.setSseKmsEncryptedObjectsStatus("Enabled");
            // 指定SSE-KMS密钥ID。如果指定Status为Enabled，则必须指定该元素。
            //request.setReplicaKmsKeyID("3542abdd-5821-4fb5-a425-90adca***");
            //List prefixes = new ArrayList();
            //prefixes.add("image/");
            //prefixes.add("video");
            //prefixes.add("a");
            //prefixes.add("A");
            // 指定待复制Object的前缀Prefix。指定Prefix后，只有匹配该Prefix的Object才会复制到目标Bucket。
            //request.setObjectPrefixList(prefixes);
            //List actions = new ArrayList();
            //actions.add(AddBucketReplicationRequest.ReplicationAction.ALL);
            // 指定可以被复制到目标Bucket的操作。默认值为ALL，表示源Bucket的所有操作都会复制到目标Bucket。
            //request.setReplicationActionList(actions);
            ossClient.addBucketReplication(request);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil开启跨区域复制的具体步骤，请参见replication（跨区域复制）。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketReplication。

7.2.3. 同区域复制

同区域复制（Same-Region Replication）是指将源存储空间（Bucket）中的文件（Object）的创建、更新和删除等操作自动、异步（近实时）地复制到相同地域下的目标Bucket。

使用场景

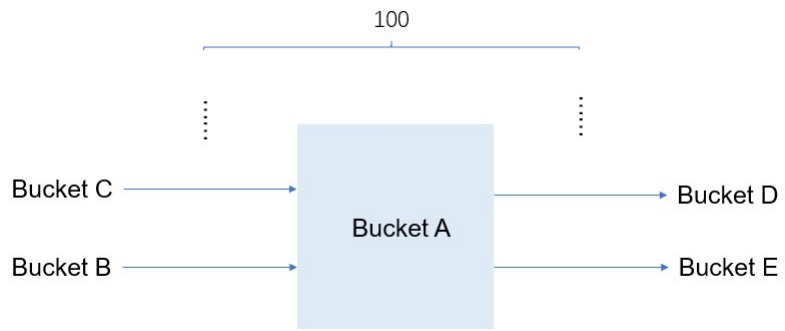
当地法规遵从性规定不允许数据离开您的国家或地区时，您可以通过配置同区域复制规则将源Bucket的数据以副本的形式存储在相同地域内的多个目标Bucket。目标Bucket中的Object是源Bucket中Object的精确副本，它们具有相同的Object名、版本信息、元数据以及内容，例如创建时间、拥有者、用户定义的元数据、Object ACL、Object内容等。

功能特性

同区域复制支持特性如下：

- 相同地域Bucket之间的数据同步

  源Bucket中的数据可以同步到相同地域的多个目标Bucket。单个Bucket关联的复制规则数量不能超过100条。这些规则中，该Bucket既可以作为源Bucket，也可以作为目标Bucket。



  如果您的业务场景涉及更大数量的复制规则，请联系技术支持。

- 实时同步数据

  对于数据的增加、删除、修改能够实时监控并同步到目标地域Bucket。对于2 MB以下文件，能够做到分钟级别信息同步，保证两边数据的最终一致。

- 历史数据迁移

  迁移历史数据，让源Bucket中历史数据也能进行同步，形成相同的两份数据。

- 实时获取同步进度

  能够针对实时同步数据展示最近同步的时间节点，针对历史数据的迁移展示迁移的百分比。

- 版本控制

  对同时处于开启版本控制状态的源Bucket和目标Bucket，保证其数据版本的最终一致性。如果数据同步方式为写（增、改）同步，则源Bucket指定版本删除的操作不会同步到目标Bucket，源Bucket创建的删除标记会同步到目标Bucket。

注意事项

- 相关费用

  开启同区域复制后，同区域的两个Bucket之间复制文件时会产生数据流量，但暂不收取数据流量费用。此外，每同步一个Object，OSS都会累计请求次数，但暂不收取请求费用。

- 复制时间

  同区域复制采用异步（近实时）复制，数据复制到目标Bucket需要一定的时间，通常几分钟到几小时不等，取决于数据的大小。

使用限制

- 仅允许同时处于非版本化或启用版本控制状态的两个Bucket开启数据同步。
- 处于同步状态下的两个Bucket不允许改变其版本控制状态。
- 对于处于同步状态的两个Bucket，由于您可以同时操作这两个Bucket，源Bucket复制过去的Object可能存在覆盖目标Bucket中同名Object的风险。

使用OSS控制台

1. 登录OSS管理控制台。
2. 在左侧导航栏，单击**Bucket列表**，然后单击任意待开启同区域复制的Bucket。
3. 在左侧导航栏，选择**冗余与容错 > 同区域复制**。
4. 在**同区域复制**区域，单击**设置**。
5. 单击**同区域复制**。
6. 在**同区域复制**面板，按如下说明配置各项参数。

参数	说明
源Bucket地域	您当前Bucket所在地域。
源Bucket	您当前Bucket名称。
目标Bucket	选择将源Bucket数据同步到相同地域下哪个目标Bucket。
数据同步对象	选择需要同步的源数据。 ○ **全部文件进行同步**：将该Bucket内所有的Object同步到目标存储空间。 ○ **指定文件名前缀进行同步**：将该Bucket内指定前缀的Object同步到目标Bucket。最多可以添加10个前缀。
Object标签	同步拥有指定标签的Object到目标Bucket。设置方法为选中**设置规则**后添加标签（键-值对），最多可添加10个标签。 要设置该参数，必须满足以下条件： ○ 已设置Object标签。具体操作，请参见设置对象标签。 ○ 源Bucket和目标Bucket均已开启版本控制。 ○ 数据同步策略为**增/改 同步**。

参数	说明
数据同步策略	选择数据同步的方式。 ○ **增/改 同步**：仅将该Bucket内Object新增和更新操作同步到目标Bucket。 ○ **增/删/改 同步**：将该Bucket内Object的新增、更新、删除操作同步到目标Bucket。
同步历史数据	选择是否同步复制规则生效前源Bucket中已有的历史数据。 ○ **同步**：将历史数据同步至目标Bucket。 **注意** 同步历史数据时，从源Bucket复制的Object可能会覆盖目标Bucket中同名的Object。为避免这部分文件丢失，建议您对源Bucket和目标Bucket开启版本控制。 ○ **不同步**：仅同步复制规则生效后上传或更新的Object。
KMS加密目标对象	在源Object或者目标Bucket使用了KMS托管密钥加密方式（即SSE-KMS，指定CMK ID）的情况下，要将Object复制到目标Bucket，则必须选中**KMS加密目标对象**，并配置以下参数： ○ **使用的KMS密钥**：为目标Object指定加密的KMS密钥。 您需要提前在KMS平台创建一个与目标Bucket相同地域的KMS密钥。具体操作，请参见创建密钥。 ○ **授权角色**：授权一个RAM角色对目标Object执行KMS加密操作。 ■ **新建角色**：新建RAM角色对目标Object执行KMS加密，角色名称格式为 `kms-replication-源Bucket名称-目标Bucket名称` 。 ■ **AliyunOSSRole**：使用AliyunOSSRole角色对目标Object执行KMS加密。若您之前未创建AliyunOSSRole角色，当您选择此项时，OSS将自动创建AliyunOSSRole角色。 **说明** 您可以通过HeadObject和GetBucketEncryption分别查询源Object和目标Bucket的加密状态。

7. 单击**确定**。
   - 当同区域复制规则创建完成后，不允许对此规则进行编辑或删除。
   - 同步任务会在同区域复制规则配置完成后即刻启动，您可以在**同区域复制**页面查看同步进度。
   - 数据复制到目标Bucket需要的时间取决于数据的大小，通常几分钟到几小时不等。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketReplication。

7.2.4. 特殊场景下的复制行为

本文介绍跨区域复制结合版本控制、生命周期、服务器端加密、合规保留策略等特殊场景的复制行为。

跨区域复制结合版本控制

跨区域复制结合版本控制的使用场景中，有如下限制：

- 仅允许同时处于非版本控制或启用版本控制状态的两个存储空间（Bucket）开启跨区域复制。处于数据同步状态下的两个Bucket不允许改变其版本控制状态。
- 数据同步过程中不允许暂停源或目标Bucket的版本控制，如有需要，您可以先删除跨区域复制规则再暂停版本控制。

从已开启版本控制的源Bucket中删除对象（Object）时，会出现以下几种情况：

请求方式	数据同步策略	结果
发出Delete请求但未指定Object版本ID	写同步（增/改）	OSS将在源Bucket中创建删除标记（Delete Marker），且源Bucket创建的删除标记会同步到目标Bucket。
	增/删/改同步	OSS将在源Bucket中创建删除标记（Delete Marker），且源Bucket创建的删除标记会同步到目标Bucket。
发出Delete请求且指定了Object版本ID	写同步（增/改）	源Bucket删除的操作不会同步到目标Bucket。
	增/删/改同步	源Bucket删除的操作将同步到目标Bucket。

有关如何在启用版本控制的Bucket中配置数据同步策略的详情，请参见设置跨区域复制。

跨区域复制结合生命周期

跨区域复制结合版本控制的使用场景，使得目标Bucket中存在Object的多个历史版本，产生较多的存储消耗。如果您希望减少目标Bucket由于跨区域复制和版本控制带来的存储成本，建议通过生命周期规则实现存储成本控制和自定义数据保留策略。

跨区域复制结合生命周期的使用场景中，有如下注意事项：

- 跨区域复制时仅将源Bucket生命周期规则作用的结果同步至目标Bucket，而不会将源Bucket的生命周期规则配置同步到目标Bucket。如果您希望目标Bucket中的Object副本能够遵循和源Bucket一样的生命周期规则，请在目标Bucket添加与源Bucket相同的生命周期规则。
- 如果您对目标Bucket设置了生命周期规则，需要注意跨区域复制后的对象副本的创建时间为对象在源Bucket中的创建时间，而非出现在目标Bucket中的时间。
- 如果您在源Bucket中设置了生命周期规则，某个Object正在进行跨区域复制的同时被生命周期规则删除，则Object的跨区域复制行为可能仍会继续，目标Bucket中的Object副本仍然保留。

跨区域复制结合服务器端加密

跨区域复制支持复制未加密的对象和使用KMS托管密钥加密、OSS完全托管加密（SSE-OSS）进行服务器端加密的对象，详情请参见服务器端加密。

跨区域复制结合服务器端加密的使用场景中，会出现以下几种情况：

源Object的加密情况	目标Bucket的加密方式	是否使用KMS加密目标对象	目标Object的加密方式
未加密	未加密	不影响	保留未加密状态
	SSE-OSS	不影响	SSE-OSS
	SSE-KMS，不指定CMK ID	不影响	SSE-KMS，不指定CMK ID
	SSE-KMS，指定CMK ID	是 配置SyncRole、CMKID	SSE-KMS，指定CMK ID
		否	不涉及（源Object无法复制到目标Bucket）
OSS完全托管加密（SSE-OSS）	无限制	不影响	SSE-OSS
KMS托管密钥加密（SSE-KMS，不指定CMK ID）	无限制	是 配置SyncRole、CMKID	SSE-KMS，指定CMK ID
		否	SSE-KMS，不指定CMK ID
KMS托管密钥加密（SSE-KMS，指定CMK ID）	无限制	是 配置SyncRole、CMKID	SSE-KMS，指定CMK ID
		否	不涉及（源Object无法复制到目标Bucket）

有关如何在配置跨区域复制规则时使用KMS加密目标Object的详情，请参见设置跨区域复制。

跨区域复制结合合规保留策略

当Bucket的合规保留策略（WORM）被锁定后，您可以在Bucket中上传和读取Object，但是在Object的保留时间到期之前，无法修改（覆写）或删除Object。

有关合规保留策略的更多详情，请参见合规保留策略。

跨区域复制结合合规保留策略的使用场景中，会出现以下几种情况：

源Object是否处于WORM保护期	源Bucket中允许的操作	目标Object是否处于WORM保护期	是否同步到目标Bucket
否	新增Object	是	否
	覆写Object	是	否
	删除Object	是	否
否	新增Object	否	是
	覆写Object	否	是
	删除Object	否	是
是	新增Object	不影响	是

7.2.5. 数据复制问题排查

在源存储空间（Bucket）配置了跨区域复制规则后，如果对象副本未出现在目标Bucket中，请参考以下几种可能原因排查并修复问题。

- 时间限制

  数据复制采用异步（近实时）复制的机制，将数据复制到目标Bucket需要一定的时间，通常几分钟到几小时不等，取决于数据的大小。如果要复制的对象较大，请稍等片刻，再检查对象副本是否出现在目标Bucket中。

- 源Bucket配置问题
  - 数据复制状态是否为已开启（Enabled）。

- 前缀（Prefix）是否正确。
  - 同步指定对象：如果需要同步源Bucket中的指定对象到目标Bucket，请将Prefix设置为指定对象名称。例如，Prefix设置为log，则仅复制log/date1.txt、log/date2.txt等以log开头的对象。与指定Prefix不匹配的对象不会复制到目标Bucket，例如date3.txt。
  - 同步所有对象：如果需要将源Bucket中的全部对象复制到目标Bucket时，请将Prefix置空。

- 复制机制限制

  如果Bucket中的某个对象是另一个复制配置创建的副本，则OSS不会复制该对象。例如，您配置了Bucket A同步到Bucket B，Bucket B再同步到Bucket C，则OSS不会将从Bucket A同步到Bucket B的对象副本复制到Bucket C。

- 版本控制状态不一致

  开启数据同步的源Bucket和目标Bucket的版本控制状态必须一致，即这两个Bucket同时处于非版本控制状态，或者都已开启版本控制。

7.3. 数据加密

7.3.1. 服务器端加密

OSS支持服务器端加密功能。上传文件（Object）时，OSS对收到的文件进行加密，再将得到的加密文件持久化保存；下载文件时，OSS自动将加密文件解密后返回给用户，并在返回的HTTP请求Header中，声明该文件进行了服务器端加密。

使用场景

OSS通过服务器端加密机制，提供静态数据保护。适合于对于文件存储有高安全性或者合规性要求的应用场景。例如，深度学习样本文件的存储、在线协作类文档数据的存储。

注意事项

- 无法使用Bucket的默认加密方式自动加密镜像回源的文件。
- 同一个Object在同一时间内仅可以使用一种服务器端加密方式。
- 如果配置了Bucket加密，仍然可以在上传或拷贝Object时单独对Object配置加密方式，且以Object配置的加密方式为准。详情请参见Put Object。

加密方式

OSS针对不同使用场景提供了两种服务器端加密方式，您可以根据实际使用场景选用。

加密方式	功能描述	使用场景	注意事项	费用说明
使用KMS托管密钥进行加解密（SSE-KMS）	使用KMS托管的默认CMK（Customer Master Key）或指定CMK进行加解密操作。数据无需通过网络发送到KMS服务端进行加解密。	因安全合规的要求，需要使用自管理、可指定的密钥。	• 用于加密Object的密钥也会被加密，并写入Object的元信息中。 • KMS托管密钥的服务器端加密方式仅加密Object数据，不加密Object的元数据。	在KMS服务侧产生少量的KMS密钥请求费用。费用详情，请参见KMS计费标准。
使用OSS完全托管密钥进行加解密（SSE-OSS）	使用OSS完全托管的密钥加密每个Object。为了提升安全性，OSS还会使用定期轮转的主密钥对加密密钥本身进行加密。	仅需要基础的加密能力，对密钥无自管理需求。	无	免费

使用OSS控制台

方式一：创建Bucket时开启服务器端加密功能

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，之后单击**创建Bucket**。
3. 在**创建Bucket**页面填写各项参数。

   其中，**服务器端加密**区域配置参数说明如下：

   - **服务端加密方式**：选择Object的加密方式。
     - **无**：不启用服务器端加密。
     - **OSS完全托管**：使用OSS托管的密钥进行加密。OSS会为每个Object使用不同的密钥进行加密，作为额外的保护，OSS会使用定期轮转的主密钥对加密密钥本身进行加密。
     - **KMS**：使用KMS默认托管的CMK或指定CMK ID进行加解密操作。

       使用KMS加密方式前，需要开通KMS服务。具体步骤，请参见开通KMS服务。
   - **加密算法**：目前仅支持AES256。
   - **加密密钥**：**服务端加密方式**选择**KMS**时，可配置此项。参数说明如下：
     - **alias/acs/oss**：使用默认托管的CMK生成不同的密钥来加密不同的Object，并且在Object被下载时自动解密。
     - CMK ID：使用指定的CMK生成不同的密钥来加密不同的Object，并将加密Object的CMK ID记录到Object的元信息中，具有解密权限的用户下载Object时会自动解密。选择指定的CMK ID前，您需在KMS管理控制台创建一个与Bucket处于相同地域的普通密钥或外部密钥。详情请参见导入密钥材料。

   其他参数请参见创建存储空间。
4. 单击**确定**。

方式二：为已创建的Bucket开启服务器端加密

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，之后单击目标Bucket名称。
3. 单击**基础设置 > 服务器端加密**。
4. 在**服务器端加密**区域，单击**设置**。

   其中，**服务器端加密**区域配置参数说明如下：

- **服务端加密方式**：选择Object的加密方式。
  - **无**：不启用服务器端加密。
  - **OSS完全托管**：使用OSS托管的密钥进行加密。OSS会为每个Object使用不同的密钥进行加密，作为额外的保护，OSS会使用定期轮转的主密钥对加密密钥本身进行加密。
  - **KMS**：使用KMS默认托管的CMK或指定CMK ID进行加解密操作。

    使用KMS加密方式前，需要开通KMS服务。具体步骤，请参见开通KMS服务。
- **加密算法**：目前仅支持AES256。
- **加密密钥**：**服务端加密方式**选择**KMS**时，可配置此项。参数说明如下：
  - **alias/acs/oss**：使用默认托管的CMK生成不同的密钥来加密不同的Object，并且在Object被下载时自动解密。
  - CMK ID：使用指定的CMK生成不同的密钥来加密不同的Object，并将加密Object的CMK ID记录到Object的元信息中，具有解密权限的用户下载Object时会自动解密。选择指定的CMK ID前，您需在KMS管理控制台创建一个与Bucket处于相同地域的普通密钥或外部密钥。详情请参见导入密钥材料。

5. 单击**保存**。

> **注意** 开启或修改Bucket默认加密方式不会影响Bucket中已有文件的加密配置。

使用阿里云SDK

以下仅列举常见SDK的服务器端加密的代码示例。关于其他SDK的服务器端加密的代码示例，请参见SDK简介。

Nodejs

```
import com.aliyun.oss.*;
import com.aliyun.oss.model.*;
public class Demo {
    public static void main(String[] args) throws Throwable {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写KMS用户主密钥ID，例如e1935511-cf88-1123-a0f8-1be8d2511***。
        String kmsId = "e1935511-cf88-1123-a0f8-1be8d2511***";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 设置Bucket加密。
            ServerSideEncryptionByDefault applyServerSideEncryptionByDefault = new ServerSideEncryptionByDefault(SSEAlgorithm.KMS);
            applyServerSideEncryptionByDefault.setKMSMasterKeyID(kmsId);
            ServerSideEncryptionConfiguration sseConfig = new ServerSideEncryptionConfiguration();
            sseConfig.setApplyServerSideEncryptionByDefault(applyServerSideEncryptionByDefault);
            SetBucketEncryptionRequest request = new SetBucketEncryptionRequest(bucketName, sseConfig);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

方式一：创建Bucket时开启服务器端加密

关于使用ossutil在创建Bucket时开启服务器端加密的具体操作，请参见bucket-encryption（服务器端加密）。

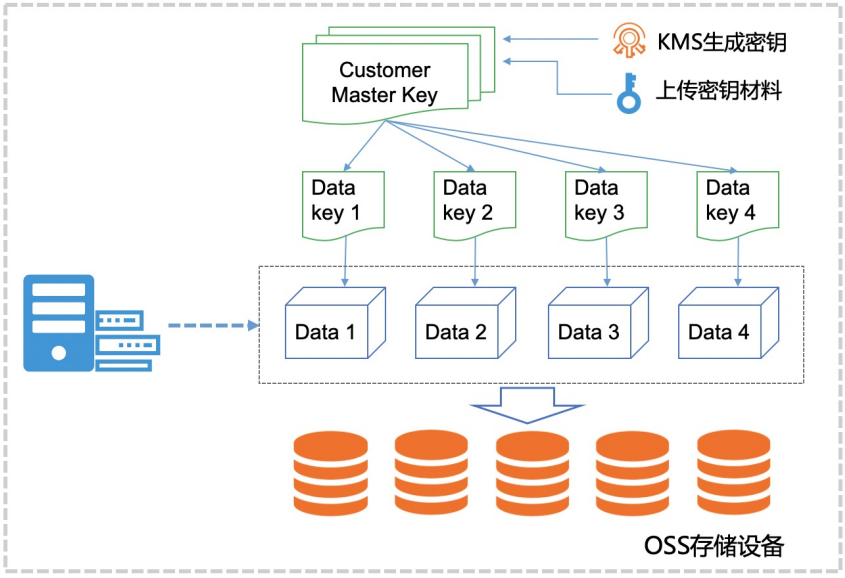
方式二：上传文件并指定加密方式

关于使用ossutil上传文件并指定加密方式的具体操作，请参见上传并指定加密方式。

使用KMS托管密钥进行加解密

使用KMS托管的用户主密钥CMK生成加密密钥加密数据，通过信封加密机制，可进一步防止未经授权的数据访问。借助KMS，您可以专注于数据加解密、电子签名验签等业务功能，无需花费大量成本来保障密钥的保密性、完整性和可用性。

SSE-KMS加密方式的逻辑示意图如下。



- 多层密钥信封加密
- 支持使用BYOK进行加解密
- 密钥由KMS服务统一管理
- 每个对象都有1个随机的加密密钥

使用SSE-KMS加密方式时，可使用如下密钥：

- 使用OSS默认托管的KMS密钥

  OSS使用默认托管的KMS CMK生成不同的密钥来加密不同的Object，并且在下载时自动解密。首次使用时，OSS会在KMS平台创建一个OSS托管的CMK。

  配置方式如下：

  - 配置Bucket默认加密方式

    配置Bucket默认加密方式为KMS，但不指定具体的CMK ID。此后，所有上传至此Bucket的Object都会被加密。

  - 为指定Object配置加密方式

    上传Object或修改Object的meta信息时，在请求中携带 `x-oss-server-side-encryption` 参数，并设置参数值为 `KMS` 。此时，OSS将使用默认托管的KMS CMK，并通过AES256加密算法加密Object。更多详情，请参见PutObject。

- 使用自带密钥BYOK（Bring Your Own Key）

  您在KMS控制台使用BYOK材料生成CMK后，OSS可使用指定的KMS CMK生成不同的密钥来加密不同的Object，并将加密Object的CMK ID记录到Object的元数据中，只有具有解密权限的用户下载Object时才会自动解密。

  BYOK材料来源有两种：

  - 由阿里云提供的BYOK材料：在KMS平台创建密钥时，选择密钥材料来源为**阿里云KMS**。
  - 使用用户自有的BYOK材料：在KMS平台创建密钥时，选择密钥材料来源为**外部**，并按照要求导入外部密钥材料。导入外部密钥可参考文档导入密钥材料。

  配置方式如下：

  - 配置Bucket默认加密方式

    配置Bucket默认加密方式为KMS，并指定具体的CMK ID。此后，所有上传至此Bucket的Object都会被加密。

  - 为目标Object配置加密方式

    上传Object或修改Object的meta信息时，在请求中携带 `x-oss-server-side-encryption` 参数，并设置参数值为 `KMS` ；携带 `x-oss-server-side-encryption-key-id` 参数，并设置参数值为指定CMK ID。此时，OSS将使用指定的KMS CMK，并通过AES256加密算法加密Object。更多详情，请参见PutObject。

使用OSS完全托管密钥进行加解密

OSS负责生成和管理数据加密密钥，并采用高强度、多因素的安全措施进行保护。数据加密的算法采用行业标准的AES256（即256位高级加密标准）算法。

配置方式如下：

- 配置Bucket默认加密方式

  配置Bucket默认加密方式为OSS完全托管，并指定加密算法为AES256。此后，所有上传至此Bucket的Object都会默认被加密。

- 为目标Object配置加密方式

  上传Object或修改Object的meta信息时，在请求中携带 `x-oss-server-side-encryption` 参数，并设置参数值为 `AES256` 。此时，目标Object将使用OSS完全托管的密钥进行加密。更多详情，请参见PutObject。

权限说明

RAM用户在如下场景中使用服务端加解密时，需要具有以下权限：

- 设置Bucket默认加密方式
  - 具有对目标Bucket的管理权限。
  - 具有 `PutBucketEncryption` 和 `GetBucketEncryption` 权限。

- 若设置加密方式为SSE-KMS，且指定了CMK ID，还需要 `ListKeys` 、 `Listalias` 、 `ListAliasesByKeyId` 以及 `DescribeKeys` 权限。此场景下的RAM Policy授权策略如下：

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:List*",
        "kms:DescribeKey"
      ],
      "Resource": [
        "acs:kms:*:141661496593****:*" //示例表示允许调用该阿里云账号ID下所有的KMS密钥，如果仅允许使用某个CMK，此处可输入对应的CMK ID。
      ]
    }
  ]
}
```

- 上传文件至设置了默认加密方式的Bucket
  - 具有目标Bucket的上传文件权限。
  - 若设置加密方式为KMS，并指定了CMK ID，还需要 `ListKeys` 、 `Listalias` 、 `ListAliasesByKeyId` 、 `DescribeKeys` 、 `GenerateDataKey` 以及 `kms:Decrypt` 权限。此场景下的RAM Policy授权策略如下：

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:List*",
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "acs:kms:*:141661496593****:*"//示例表示允许调用该阿里云账号ID下所有的KMS密钥，如果仅允许使用某个CMK，此处可输入对应的CMK ID。
      ]
    }
  ]
}
```

- 从设置了默认加密方式的Bucket中下载文件
  - 具有目标Bucket的文件访问权限。
  - 若设置加密方式为KMS，并指定了CMK ID，还需要 `Decrypt` 权限。此场景下的RAM Policy授权策略如下：

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
    "kms:Decrypt"
      ],
      "Resource": [
        "acs:kms:*:141661496593****:*"//示例表示具有该阿里云账号ID下所有KMS的解密权限。若要针对某个KMS密钥进行解密，此处可输入对应的CMK ID。
      ]
    }
  ]
}
```

常见问题

配置Bucket默认加密方式后，OSS会对历史文件进行加密吗？

OSS只对服务器端加密配置生效后上传的Object进行加密，不会加密历史文件。若您需要加密历史文件，可通过CopyObject覆写历史文件。

7.3.2. 客户端加密

客户端加密是指将文件（Object）发送到对象存储OSS之前在本地进行加密，本文介绍客户端加密的方式与流程。

免责声明

- 使用客户端加密功能时，您需要对主密钥的完整性和正确性负责。因您维护不当导致主密钥用错或丢失，从而导致加密数据无法解密所引起的一切损失和后果均由您自行承担。
- 在对加密数据进行复制或者迁移时，您需要对加密元信息的完整性和正确性负责。因您维护不当导致加密元信息出错或丢失，从而导致加密数据无法解密所引起的一切损失和后果均由您自行承担。

操作方式

您可以通过如下SDK进行客户端加密：

- Java SDK
- Python SDK
- Go SDK

- C++ SDK

背景信息

使用客户端加密时，会为每个Object生成一个随机数据加密密钥，用该随机数据加密密钥明文对Object的数据进行对称加密。主密钥用于生成随机的数据加密密钥，加密后的内容会当作Object的meta信息保存在服务端。解密时先用主密钥将加密后的随机密钥解密出来，再用解密出来的随机数据加密密钥明文解密Object的数据。主密钥只参与客户端本地计算，不会在网络上进行传输或保存在服务端，以保证主密钥的数据安全。

> **注意**
> - 客户端加密支持分片上传超过5 GB的文件。在使用分片方式上传文件时，需要指定上传文件的总大小和分片大小， 除了最后一个分片外，每个分片的大小要一致，且分片大小目前必须是16的整数倍。
> - 调用客户端加密上传文件后，加密元数据会被保护，无法通过CopyObject修改Object meta信息。

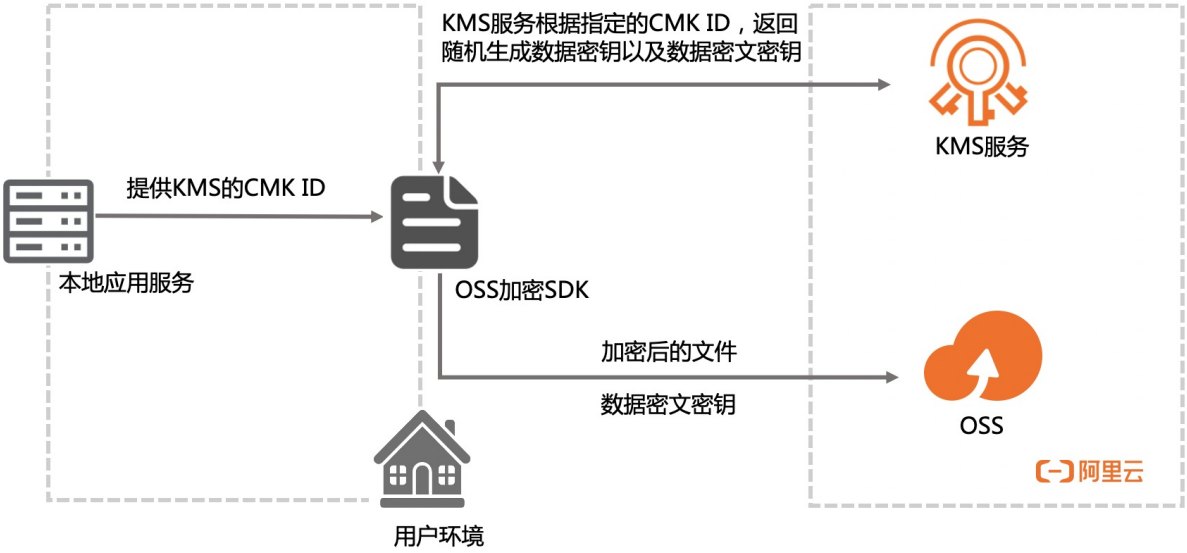
对于主密钥的使用，目前支持如下两种方式：

- 使用KMS托管用户主密钥
- 使用用户自主管理密钥

完整的示例代码请参见GitHub。

使用KMS托管用户主密钥

当使用KMS托管用户主密钥用于客户端数据加密时，无需向OSS加密客户端提供任何加密密钥，只需要在上传对象时指定KMS用户主密钥ID（也就是CMK ID）。具体工作原理如下图所示。



- 加密并上传Object
  i. 获取加密密钥。

     通过使用CMK ID，客户端首先向KMS发送一个请求，申请1个用于加密Object的数据密钥（Data Key）。作为响应，KMS会返回一个随机生成的数据明文密钥（Data Key）以及一个数据密文密钥（Encrypted Data Key）。

  ii. 加密数据并上传至OSS。

     本地客户端接收到KMS返回的数据明文密钥以及数据密文密钥后，将使用数据明文密钥进行本地加密，并且将加密后的对象以及数据密文密钥上传至OSS。
- 下载并解密Object
  i. 下载Object。

     客户端从OSS服务端下载加密的Object以及作为对象元数据存储的数据密文密钥。

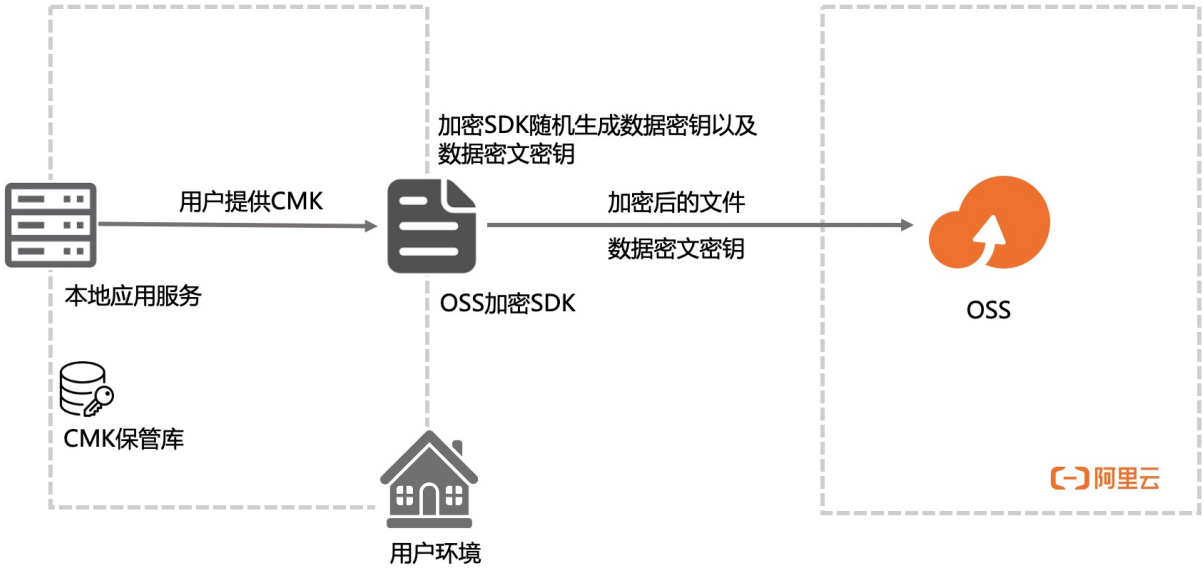
  ii. 解密Object。

     客户端将数据密文密钥以及CMK ID发送至KMS服务器。作为响应，KMS将使用指定的CMK解密，并且将数据明文密钥返回给本地加密客户端。

> **说明**
> - 本地加密客户端为每一个上传的对象获取一个唯一的数据加密密钥。
> - 为了保证数据的安全性，建议CMK定期轮换或者更新。
> - 您需要维护CMK ID与Object之间的映射关系。

使用用户自主管理密钥

使用用户自主管理密钥时，需要您自主生成并保管加密密钥。当本地客户端加密Object时，由用户自主上传加密密钥（对称加密密钥或者非对称加密密钥）至本地加密客户端。其具体加密过程如下图所示。



- 加密并上传Object
  i. 用户向本地加密客户端提供1个用户主密钥（对称密钥或者非对称密钥）。
  ii. 本地加密客户端在本地生成一个一次性的对称密钥，即数据密钥（Data Key）。它将用于加密单个对象（针对每个对象，客户端都会随机生成1个数据密钥）。
  iii. 客户端使用数据密钥加密对象，并使用用户提供的主密钥来加密数据密钥。
  iv. 客户端将加密的数据密钥（Encrypted Data Key）作为对象元数据的一部分上传至OSS。
- 下载并解密Object
  i. 客户端从OSS下载加密的对象以及元数据。
  ii. 通过使用元数据中的材料，客户端将授权确定使用对应主密钥来解密数据密钥，之后使用解密后的数据密钥来解密对象。

> **注意**
> - OSS本地加密客户端不会将用户主密钥以及未加密的数据发送至OSS。所以，请务必妥善保管加密密钥，如果密钥丢失，将无法解密数据。
> - 数据密钥由本地加密客户端随机生成。

7.4. 版本控制

7.4.1. 版本控制介绍

版本控制是针对存储空间（Bucket）级别的数据保护功能。开启版本控制后，针对数据的覆盖和删除操作将会以历史版本的形式保存下来。您在错误覆盖或者删除对象（Object）后，能够将Bucket中存储的Object恢复至任意时刻的历史版本。

使用场景

建议您在以下场景中使用版本控制，为您的数据安全提供更好的保障。

- 数据误删除

  当前OSS不提供回收站功能。您删除OSS数据后想要找回时，可使用版本控制功能，恢复已删除的数据。
- 文件被覆盖

  对于网盘、在线协作类产品，文件会被频繁修改，针对文件的编辑会产生大量的临时版本。您可以使用版本控制功能找回某个时间点的版本。

注意事项

使用版本控制时，有如下注意事项：

- 费用说明

  版本控制功能本身不收取任何费用，但对当前版本和所有历史版本的文件都会收取存储费用。为避免不必要的存储费用，请及时删除不需要的历史版本文件；此外，若您对历史版本文件进行下载或恢复等操作，还会产生相应的请求费用、流量费用等。计费详情，请参见计量项与计费项。
- 权限说明

  只有Bucket的拥有者及授予了PutBucketVersioning权限的RAM用户才能配置版本控制。
- 功能互斥
  - 同一Bucket中，版本控制与合规保留策略无法同时配置。
  - 如果Bucket已开启版本控制，上传文件时附加的覆盖同名文件请求头 `x-oss-forbid-overwrite` 将不生效。更多信息，请参见请求头。

版本控制状态

Bucket包含三种版本控制状态，分别为未开启、开启或者暂停。

- 默认情况下，Bucket版本控制状态为“未开启”。一旦Bucket处于“开启”版本状态，将无法返回至“未开启”状态。但是，您可以暂停Bucket的版本控制状态。
- 当Bucket版本控制处于“开启”状态时，OSS将为新上传的Object生成全局唯一的随机字符串版本ID。有关启用版本控制状态下Object的相关操作详情，请参见开启版本控制下Object的操作。
- 当Bucket版本控制处于“暂停”状态时，OSS将为新上传的Object生成特殊字符串为“null”的版本ID。有关暂停版本控制状态下Object的相关操作详情，请参见暂停版本控制下Object的操作。

> **说明** 当Bucket版本控制处于“开启”状态时，由于Object的每个版本都被保存下来，每个版本都会占用存储空间，OSS会对Object的所有版本收取存储费用。建议结合您的使用场景通过生命周期规则，将当前版本或历史版本Object转换为低频或归档存储类型或删除不再需要的历史版本，以降低您的存储费用，详情请参见使用生命周期管理文件版本。

数据保护

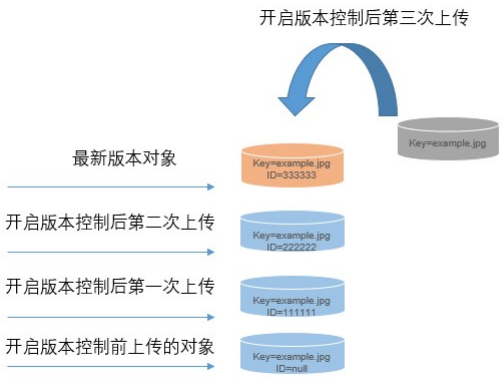
以下表格详细阐述了不同版本控制状态下，OSS对覆盖和删除Object的处理逻辑，帮助您了解版本控制状态下的数据保护机制。

版本控制状态	覆盖Object	删除Object
未开启	已有Object被直接覆盖，且无法恢复，只能获取最新版本Object。	直接删除，无法再获取此Object。
开启	为此Object添加新的版本ID，历史版本不受影响。	为此Object添加删除标记（Delete Marker），删除标记将携带一个全局唯一的版本ID，历史版本不受影响。
暂停	为此Object产生版本ID为“null”的新版本。 历史版本里若已存在版本号为“null”的Object或删除标记，则将会被新的“null”版本Object覆盖，其他非“null”版本的Object或删除标记不受影响。	为此Object产生版本ID为“null”的删除标记。 历史版本里若已存在版本号为“null”的Object或删除标记，则将会被新的删除标记覆盖，其他非“null”版本的Object或删除标记不受影响。

以下以图示的方法说明在Bucket版本控制状态处于“开启”和“暂停”时，上传同名Object或删除Object时OSS的处理行为。图示中的版本号均以简短版本号代替。

- 开启版本控制下的Object覆盖操作

  在开启版本控制的Bucket中连续执行上传Object操作，Object虽然被多次覆盖，但每次覆盖操作均会产生一个独立的版本ID。

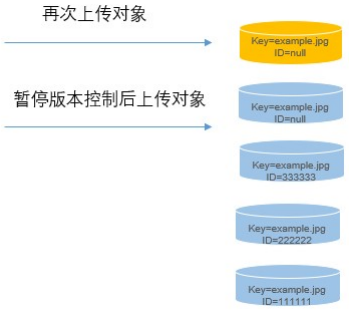


- 开启版本控制下的Object删除操作

  在开启版本控制下的Bucket中删除Object时，历史版本Object不会被真正删除，而是产生一个删除标记来标识Object的当前版本是删除状态。如果再重复上传同名Object，将产生新的版本ID。

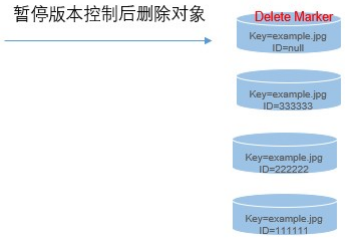
- 暂停版本控制下的Object覆盖操作

  在暂停版本控制状态Bucket中上传Object时，历史版本数据继续保留，新上传的Object版本号为“null”。如果再重复上传同名Object，将产生新的“null”版本，并自动把前一次的“null”版本覆盖。



- 暂停版本控制下的Object删除操作

  在暂停版本控制下的Bucket中删除Object时，历史版本Object不会被真正删除，而是产生一个删除标记来标识Object的当前版本是删除状态。



从上述信息得知，当您的Bucket处于版本控制状态时，针对数据的覆盖和删除操作将会以历史版本的形式保存下来。您在错误覆盖或者删除Object后，能够将Bucket中存储的Object恢复至任意时刻的历史版本。

开启版本控制

使用OSS控制台

开启版本控制后，OSS会为Bucket中所有Object的每个版本指定唯一的versionId。

- 新建Bucket时开启版本控制。
  i. 登录OSS管理控制台。
  ii. 单击**Bucket列表**，然后单击**创建Bucket**。
  iii. 在**创建Bucket**页面配置各项参数。

     其中，**版本控制**区域选择**开通**。其他参数的配置详情，请参见创建存储空间。
  iv. 单击**确定**。
- 对已创建的Bucket开启版本控制。
  i. 单击**Bucket列表**，然后单击目标Bucket名称。
  ii. 在左侧导航栏，选择**冗余与容错 > 版本控制**。
  iii. 单击**设置**，然后版本控制状态选择**开通**。
  iv. 单击**保存**。

开启版本控制后，您可以在**文件管理**页面查看所有版本的文件。如果仅需查看文件的当前版本，请将**历史版本**状态设置为**隐藏**。隐藏历史版本并不能提升列举文件的性能，如果列举文件时页面响应过慢，请参见响应速度下降排查并解决。

使用阿里云SDK

以下仅列举常见SDK的开启版本控制的代码示例。关于其他SDK的开启版本控制的代码示例，请参见SDK简介。

Python

```
Failed to resolve content from t220171.dita#concept_265070/codeblock_6gw_mzv_jbe
```

使用命令行工具ossutil

关于使用ossutil开启版本控制的具体步骤，请参见设置版本控制状态。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketVersioning。

暂停版本控制

使用OSS控制台

开启版本控制后，您还可以随时暂停版本控制以停止在Bucket中继续累积同一Object的新版本。暂停版本控制后，OSS将为新生成的Object添加versionId为null的版本，已有的历史版本Object将继续保留。

暂停Bucket版本控制的操作步骤如下：

1. 单击**Bucket列表**，然后单击要暂停版本控制的目标Bucket名称。
2. 在左侧导航栏，选择**冗余与容错 > 版本控制**。
3. 单击**设置**，版本控制状态选择**暂停**。

4. 单击**保存**。

使用阿里云SDK

以下仅列举常见SDK的暂停版本控制的代码示例。关于其他SDK的暂停版本控制的代码示例，请参见SDK简介。

Python

```
// Endpoint以杭州为例，其它Region请按实际情况填写。
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// 阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 设置存储空间版本控制状态为Suspended。
BucketVersioningConfiguration configuration = new BucketVersioningConfiguration();
configuration.setStatus(BucketVersioningConfiguration.SUSPENDED);
SetBucketVersioningRequest request = new SetBucketVersioningRequest(bucketName, configuration);
ossClient.setBucketVersioning(request);
// 关闭OSSClient。
ossClient.shutdown();
```

使用命令行工具ossutil

关于使用ossutil暂停版本控制的具体步骤，请参见设置版本控制状态。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketVersioning。

7.4.2. 开启版本控制下Object的操作

存储空间（Bucket）开启版本控制后，OSS会为Bucket中所有文件（Object）的每个版本指定唯一的ID值，且Bucket中现有Object的内容、权限保持不变。开启版本控制后，还能够防止意外覆盖或者删除Object，并允许查询、恢复Object的历史版本。

注意事项

当您在开启了版本控制的Bucket中进行上传文件、列举文件、下载文件、删除文件、恢复文件等操作时，有如下注意事项：

- 开启版本控制的Bucket会维护一个当前版本Object，以及零个或零个以上历史版本Object。
- 如果在Bucket开启版本控制前上传了Object，则OSS将Object的版本ID值置为null。
- 以下图示中的版本ID均以简短版本ID代替。

有关版本控制的详细说明，请参见版本控制介绍。

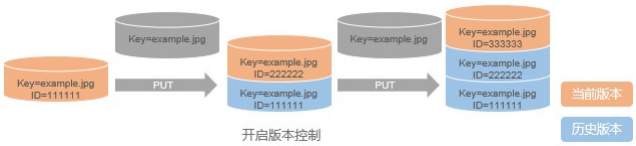
上传文件

在开启了版本控制的Bucket上传Object时，OSS会为这些上传的Object自动添加唯一的版本ID。

> **说明** PutObject、PostObject、CopyObject、MultipartUpload等操作都会为新生成的Object自动添加唯一的版本ID。

通过PUT操作上传Object（key=example.jpg）时，OSS为该Object指定了唯一的版本号（ID=111111），如下图所示。

通过PUT操作第一次上传同名Object（key=example.jpg）时，原始Object版本（ID=111111）作为历史版本，生成的新版本（ID=222222）将作为当前版本保存在Bucket中。当再次上传同名Object时，原始Object版本（包括ID=111111以及ID=222222）将作为历史版本，而生成的新版本（ID=333333）则作为当前版本保存在Bucket中，如下图所示。



您可以通过cp命令、Java SDK、PHP SDK、Node.js SDK、Python SDK、.NET SDK、Go SDK、C++ SDK的方式在已开启版本控制的Bucket中上传文件。

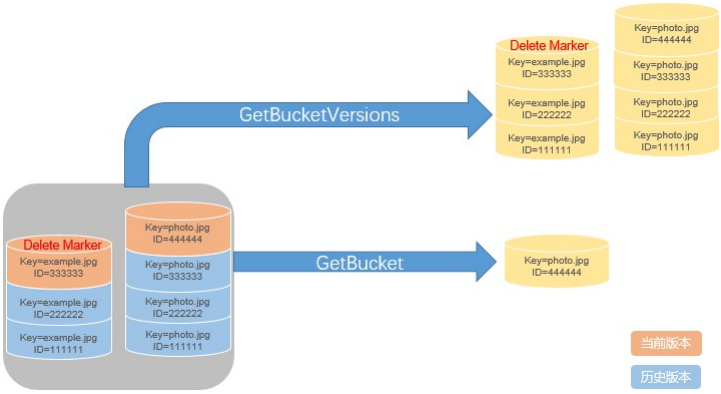
列举文件

在开启了版本控制的Bucket中，您可以使用GetBucketVersions(ListObjectVersions)接口获取Object的所有版本信息，包括删除标记（Delete Marker）。

- 与GetBucketVersions(ListObjectVersions)不同的是，GetBucket(ListObject)接口仅返回Object的当前版本，且当前版本不为删除标记。
- 单个GetBucketVersions(ListObjectVersions)请求最多返回1000个版本Object。您可以通过发送多次请求来获取Object的所有版本。

  例如，如果Bucket中包含两个Key（如example.jpg和photo.jpg），且第一个Key（example.jpg）有900个版本，第二个Key（photo.jpg）有500个版本，则单个请求将先按照Key的字母序，再按照版本的新旧顺序依次列举example.jpg的所有900个版本，另加photo.jpg的100个版本。

如下图所示，在开启了版本控制的Bucket中，调用GetBucketVersions(ListObjectVersions)接口时，返回了Bucket中所有Object的所有版本，包含当前版本为删除标记的Object；调用GetBucket(ListObject)接口时，则仅返回Object的当前版本，且当前版本不能为删除标记，因此仅返回当前版本ID为444444的Object。

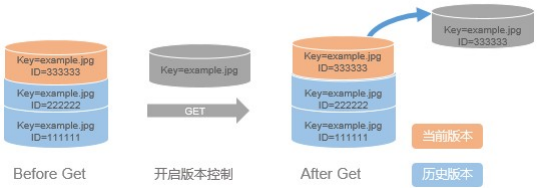


您可以通过ls命令、Java SDK、Node.js SDK、Python SDK、Go SDK的方式在开启版本控制的Bucket中列举文件。

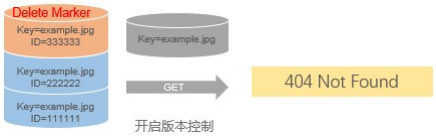
下载文件

您可以在开启了版本控制Bucket下载当前版本或指定版本的Object。

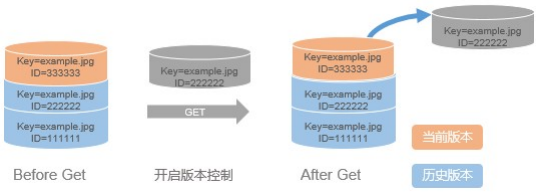
通过GET请求下载Object时，如果没有指定Object的版本ID，默认情况下返回Object的当前版本。如下图所示返回Object的当前版本（ID=333333）。



在当前版本为删除标记（Delete Marker）时执行GET操作时返回404 Not Found。



如果要下载指定的Object版本，则通过GET请求下载Object时需要指定其版本ID，如下图所示获取指定版本ID为222222的Object。



您可以通过cp命令、Java SDK、PHP SDK、Node.js SDK、Python SDK、.NET SDK、Go SDK、C++ SDK的方式在已开启版本控制的Bucket中下载文件。

删除文件

开启版本控制后，您可以通过指定Object版本ID或者配置Lifecycle将Object永久删除。如果删除Object时未指定版本ID，则Bucket中将插入一个删除标记（Delete Marker）作为当前版本。

> **说明** 开启版本控制后，若删除Object时未指定版本ID，默认不会删除Object的当前版本以及历史版本。

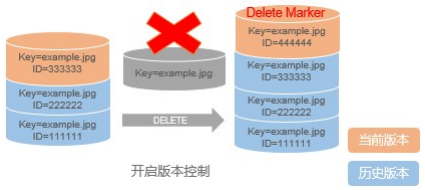
此外，您还可以在开启版本控制的Bucket中通过生命周期规则的Expiration元素指定对象当前版本过期，还可以通过NoncurrentVersionExpiration元素永久删除非当前版本对象，对于这两种元素的详细说明如下：

- Expiration元素应用于当前对象版本，OSS通过添加删除标记将当前版本作为非当前版本保留，而不是删除当前版本对象，然后删除标记将成为对象的当前版本。
- NoncurrentVersionExpiration元素适用于非当前版本对象，OSS会永久删除这些对象版本，且无法恢复永久删除的对象。

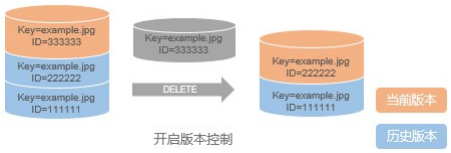
有关版本控制结合生命周期的更多信息，请参见生命周期配置元素。

以下分别说明在未指定版本ID以及指定版本ID的情况下，执行DELETE操作时Object的删除行为。

- 如果未指定Object的版本ID，则OSS会插入一个删除标记作为当前版本，该删除标记也会有相应的唯一版本ID，但没有相关数据和ACL等，如下图所示（当前版本为删除标记，且版本ID=444444）。



- 如果指定Object的版本ID，则永久删除该指定版本的Object，如下图所示（即删除版本ID=333333的Object）。



您可以通过rm命令、Java SDK、PHP SDK、Node.js SDK、Python SDK、.NET SDK、Go SDK、C++ SDK的方式在已开启版本控制的Bucket中删除文件。

恢复文件

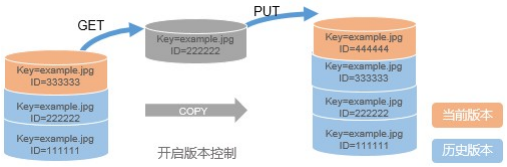
开启版本控制后，Bucket中Object的所有版本都将得以保留。您可以通过恢复指定历史版本的方式，使得任意Object的历史版本成为当前版本。

您可以通过以下两种方式将Object的历史版本恢复至当前版本：

- 通过CopyObject来恢复Object的历史版本

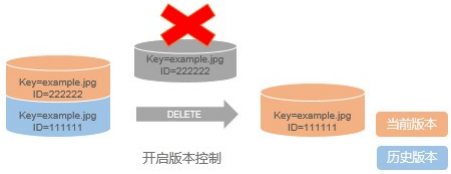
  复制的Object将成为该Object的当前版本，且所有Object版本都将保留。

  如下图所示，将原Object的历史版本（ID=222222）复制到同一个Bucket中，OSS将为该Object生成新的版本（ID=444444），并将其置为该Object的当前版本。因此，该Object同时具有历史版本（ID=222222）以及当前版本（ID=444444）。



- 通过删除Object的当前版本来恢复Object的历史版本

  如下图所示，当您通过DELETE versionId的方式永久删除当前Object版本（ID=222222）后， 下一个历史版本（ID=111111）成为了该Object的当前版本。



> **注意** 由于Object的当前版本删除后无法恢复，建议您通过CopyObject的方式来恢复Object的历史版本。

您可以通过cp命令、Java SDK、PHP SDK、Node.js SDK、Python SDK、.NET SDK、Go SDK、C++ SDK的方式在已开启版本控制的Bucket中恢复历史版本文件。

7.4.3. 暂停版本控制下Object的操作

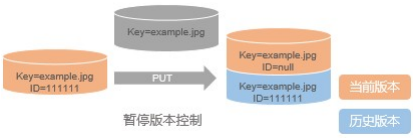
您可以暂停版本控制以停止在存储空间（Bucket ） 中继续累积同一文件（Object）的新版本。暂停版本控制后，您可以上传文件，并通过指定版本ID（versionId）的方式对历史版本Object进行下载和删除操作。

上传文件

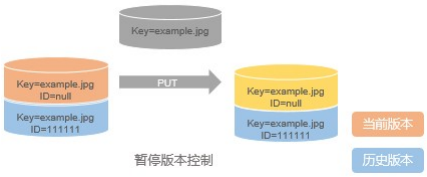
向暂停版本控制的存储空间（Bucket）上传文件（Object）时，OSS将为新生成的Object添加versionId为null的版本，且每个Object只会保留一个versionId为null的版本。

- 如下图所示，向暂停版本控制的Bucket中通过PUT操作上传Object时，OSS会为上传的Object自动添加null的版本ID。

- 如下图所示，如果暂停版本控制的Bucket中存在开启版本控制时生成的Object版本（ID=111111），通过PUT操作向该Bucket上传同名Object时，OSS会为新版本Object分配null的版本ID，且该版本作为当前版本，同时开启版本控制时生成的Object版本（ID=111111）将作为历史版本保存下来。



- 如果暂停版本控制的Bucket中已存在版本ID为null的Object，通过PUT操作向该Bucket上传同名Object时，原版本ID为null的版本将被覆盖。



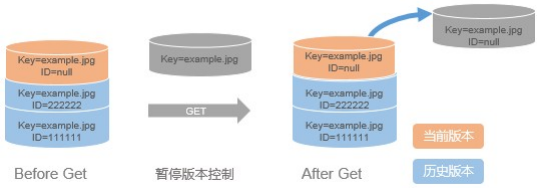
您可以通过cp命令、Java SDK、PHP SDK、Node.js SDK、Python SDK、.NET SDK、Go SDK、C++ SDK的方式在已暂停版本控制的Bucket中上传文件。

下载文件

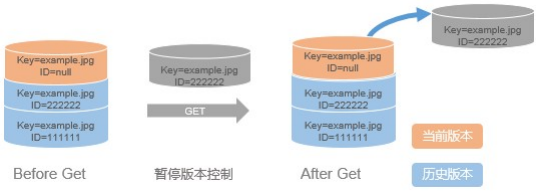
您可以在暂停版本控制的存储空间（Bucket）中下载当前版本或指定版本的文件（Object）。

通过GET请求下载Object时：

- 如果没有指定Object的版本ID，默认情况下返回Object的当前版本。如下图所示返回版本ID为null的当前版本。



- 如果要下载指定的版本，则通过GET请求下载Object时需要指定其版本ID，如下图所示获取的指定版本为（ID=222222）。



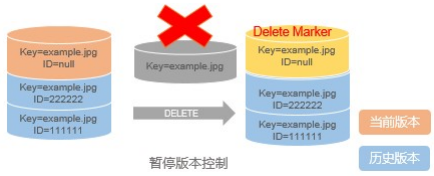
您可以通过cp命令、Java SDK、PHP SDK、Node.js SDK、Python SDK、.NET SDK、Go SDK、Go SDK的方式在已暂停版本控制的Bucket中下载文件。

删除文件

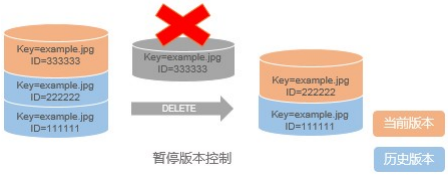
在暂停版本控制的Bucket中执行DELETE操作时，分以下三种情形：

- 如果对Bucket中当前版本ID不为null的Object执行DELETE操作时，则OSS会插入版本ID为null的删除标记（Delete Marker）作为当前版本。

- 如果对Bucket中当前版本ID为null的Object执行DELETE操作时，则OSS会插入版本ID为null的删除标记（Delete Marker）作为当前版本。由于OSS保证同一个Object只允许存在一个null的版本，因此原版本ID为null的版本将被覆盖。



- 如果通过DELETE+versionId的方式删除Object，则该指定版本的Object将被永久删除，如下图所示（即删除版本ID=333333的Object）。



您可以通过rm命令、Java SDK、PHP SDK、Node.js SDK、Python SDK、.NET SDK、Go SDK、C++ SDK的方式在已暂停版本控制的Bucket中删除文件。

7.4.4. 删除标记

删除标记（Delete Marker）是用于受版本控制的对象（Object）的占位符，即DELETE请求中命名的标记符。

删除标记与Object的异同

与其他任何Object一样，删除标记同样有文件名称（Key）和版本ID，但删除标记在以下方面与其他Object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。当前版本为删除标记的Object时，GET请求会引发404错误。
- 仅拥有 `oss:DeleteObjectVersion` 权限的用户只能对删除标记执行DELETE操作。

对已开启版本控制或已暂停版本控制的Bucket发送DeleteObject请求时，OSS将创建删除标记。在DeleteObject请求中如果未指定Object的版本ID，则不会删除Object，而是创建删除标记作为Object的当前版本。

> **说明** 无法直接删除已启用版本控制的Bucket中的Object，但删除标记可以将Object视为已删除。

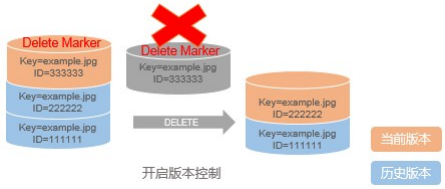
如何删除“删除标记”

以下内容介绍如何在开启版本控制的Bucket中删除“删除标记”。

如果在DELETE操作时未指定删除标记的版本ID，则OSS不会删除“删除标记”，而是插入删除标记作为Object的当前版本。删除标记可以进行累积，如下图所示。

说明 在已开启版本控制的Bucket中，相同的Object可能有多个删除标记，且删除标记将对应唯一的版本ID。

如果在DELETE请求中指定版本ID，则该指定版本的Object将被永久删除，如下图所示（即删除versionId=333333的删除标记，versionId=222222的版本成为Object的当前版本）。



您可以通过Java SDK、Python SDK、PHP SDK、Node.js SDK、.NET SDK、Go SDK、C++ SDK的方式删除指定版本Object及其删除标记。

7.4.5. 常见问题

本文介绍您在使用版本控制过程中可能遇到的问题，并提供相应的排查方法与解决方案。

存储费用

版本控制功能本身不收取任何费用，但对当前版本和所有历史版本的文件（Object）都会收取存储费用。以下列场景为例，说明使用版本控制时的存储费用（假定当月有30天）：

- 当月第1天：通过PutObject操作向某一存储空间（Bucket）上传了4 GB大小的Object，存储类型为标准存储（本地冗余）。
- 当月第16天：通过PutObject操作对同一个Bucket中的同一个Object写入5 GB的数据。

分析上述Object当月的存储费用时，请注意在第16天对Object写入5 GB数据时，第1天上传的4 GB的Object并未从Bucket中删除。相反，4 GB作为Object的历史版本在Bucket中存储了30天，而5 GB作为Object的最新版本在Bucket中存储了15天。

按照存储费用的按量付费计算规则得知，该Object当月的存储费用为：4 GB×0.12元/GB/月+5 GB×0.12元/GB/月÷2=0.78元。

有关不同类型的存储费用说明，请参见存储费用。

响应速度下降

问题描述：启用版本控制后，调用GetBucket (ListObjects)接口列举当前版本Object时，为什么响应速度会显著下降？

问题原因：您的Bucket中有一个或多个Object包含大量的非当前版本Object或过期删除标记。

问题排查：

- 通过GetBucketVersions(ListObjectVersions)查看Object是否存在较多版本。详情请参见GetBucketVersions(ListObjectVersions)。
- 通过Bucket清单功能查看Bucket中Object的信息，包括版本信息、是否包含删除标记等。详情请参见存储空间清单。

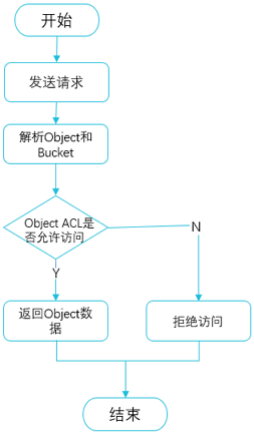
解决方法：启用生命周期管理中的非当前版本过期操作（NonCurrentVersionExpiration）以及移除过期删除标记策略（ExpiredObjectDeleteMarker），以便使早期版本的对象过期，并删除在Bucket中的过期删除标记。详情请参见生命周期配置元素。

7.5. 签名

7.5.1. OSS请求流程

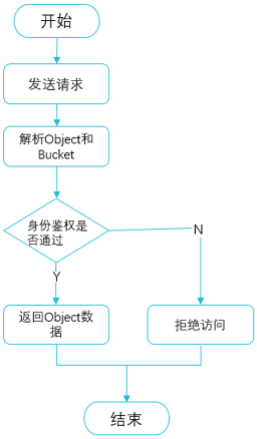
对OSS的HTTP请求可以根据是否携带身份验证信息分为匿名请求和带身份验证的请求。匿名请求指的是请求中没有携带任何和身份相关的信息；带身份验证的请求指的是按照OSS API文档中规定的在请求头部或者在请求URL中携带签名的相关信息。

匿名请求流程



1. 用户的请求被发送到OSS的HTTP服务器上。
2. OSS根据URL解析出Bucket和Object。
3. OSS检查Object ACL是否允许匿名访问。
   - 如果允许匿名访问，则返回Object的内容给用户。
   - 如果不允许匿名访问，则拒绝访问。

带身份验证的请求流程



1. 用户的请求被发送到OSS的HTTP服务器上。
2. OSS根据URL解析出Bucket和Object。
3. OSS根据请求的OSS的AccessKeyId获取请求者的相关身份信息，进行身份鉴权。
   - 如果未获取成功，则拒绝访问，请求结束。
   - 如果获取成功，但请求者不被允许访问此资源，则拒绝访问，请求结束。
   - 如果获取成功，但OSS端根据请求的HTTP参数计算的签名和请求发送的签名字符串不匹配，则返回，请求结束。
   - 如果身份鉴权成功，则返回Object的内容给用户。

AccessKey类型

目前访问OSS使用的 AccessKey（AK）有三种类型。

- 阿里云账号AK

  阿里云账号AK特指阿里云主账号的AK，每个阿里云账号提供的AccessKey对拥有的资源有完全控制的权限。每个阿里云账号能够同时拥有不超过5个active或者inactive的AK对（AccessKeyId和AccessKeySecret）。

  用户可以登录AccessKey管理控制台，申请新增或删除AK对。

  每个AK对都有active和inactive两种状态。

  - Active：表明用户的AK处于激活状态，可以在身份验证的时候使用。
  - Inactive：表明用户的AK处于非激活状态，不能在身份验证的时候使用。

  > **注意** 为了您的数据安全，不建议直接使用阿里云账号AccessKey，您可以创建RAM子账号之后并授权后，使用子账号的AccessKey管理您的资源。

- RAM子账号AK

  RAM（Resource Access Management）是阿里云提供的资源访问控制服务。RAM账号AK指的是通过RAM被授权的AK。这组AK只能按照RAM定义的规则去访问Bucket里的资源。通过RAM，您可以集中管理您的用户（比如员工、系统或应用程序），以及控制用户可以访问您名下哪些资源的权限。比如，能够限制您的用户只拥有对某一个Bucket的读权限。子账号是从属于主账号的，并且这些账号下不能拥有实际的任何资源，所有资源都属于主账号。

- STS账号AK

  STS（Security Token Service）是阿里云提供的临时访问凭证服务。STS账号AK指的是通过STS颁发的AK。这组AK只能按照STS定义的规则去访问Bucket里的资源。

身份验证具体实现

目前主要有三种身份验证方式：

- AK验证
- RAM验证
- STS验证

当用户以个人身份向OSS发送请求时，其身份验证的实现如下：

1. 用户将发送的请求按照OSS指定的格式生成签名字符串。
2. 用户使用AccessKeySecret对签名字符串进行加密产生验证码。
3. OSS收到请求以后，通过AccessKeyId找到对应的AccessKeySecret，以同样的方法提取签名字符串和验证码。
   - 如果计算出来的验证码和提供的一样即认为该请求是有效的。
   - 否则，OSS将拒绝处理这次请求，并返回HTTP 403错误。

带身份验证访问OSS的三种方法

- 使用控制台访问OSS：控制台中对用户隐藏了身份验证的细节，使用控制台访问OSS的用户无需关注细节。更多信息请参见下载文件。
- 使用SDK访问OSS：OSS提供了多种开发语言的SDK，SDK中实现了签名算法，只需要将AccessKey信息作为参数输入即可。SDK示例请参见：
  - Java SDK
  - Python SDK
  - Go SDK
  - C++ SDK
  - PHP SDK
  - C SDK
  - .NET SDK
  - Android SDK
  - iOS SDK
  - Node.js
  - Browser.js
- 使用API访问OSS：如果您想用自己喜欢的语言来封装调用RESTful API接口，您需要实现签名算法来计算签名。具体请参见API手册中的在Header中包含签名和在URL中包含签名。

7.5.2. 在Header中包含签名

您可以在HTTP请求中增加 `Authorization` 的Header来包含签名（Signature）信息，表明该消息已被授权。

SDK签名实现

OSS SDK已实现签名，您使用OSS SDK时无需关注签名问题。如果您想了解具体语言的签名实现，请参考OSS SDK的代码。OSS SDK签名实现的文件请参见下表。

SDK	签名实现
Java SDK	OSSRequestSigner.java
Python SDK	auth.py
.Net SDK	OssRequestSigner.cs
PHP SDK	OssClient.php
C SDK	oss_auth.c
JavaScript SDK	client.js
Go SDK	auth.go
Ruby SDK	util.rb
iOS SDK	OSSModel.m
Android SDK	OSSUtils.java

Authorization字段计算的方法

- 计算方法

```
Authorization = "OSS " + AccessKeyId + ":" + Signature
Signature = base64(hmac-sha1(AccessKeySecret,
            VERB + "\n"
            + Content-MD5 + "\n"
            + Content-Type + "\n"
            + Date + "\n"
            + CanonicalizedOSSHeaders
            + CanonicalizedResource))
```

- 参数说明

参数	类型	是否必选	示例值	说明
AccessKeyId	字符串	是	LTAI4FixJvEPgvZ6g5cC****	密钥中的AccessKey ID。
AccessKeySecret	字符串	是	Q0YehC6ZyugWfjod5y8Rcqrc1y****	密钥中的AccessKey Secret。
VERB	枚举值	是	PUT	HTTP请求的Method，包含PUT、GET、POST、HEAD、DELETE等。
\n	字符串	否	\n	换行符。
Content-MD5	字符串	否	eB5eJF1ptWaXm4bijSPyxw==	请求内容数据的MD5值，对消息内容（不包括头部）计算MD5值获得128比特位数字，对该数字进行base64编码得出。更多信息，请参见RFC2616 Content-MD5。 该请求头可用于消息合法性的检查（消息内容是否与发送时一致），也可以为空。 关于Content-MD5的计算方法，请参见Content-MD5的计算方法。
Content-Type	字符串	否	application/octet-stream	请求内容的类型，也可以为空。
Date	字符串	是	Sun, 22 Nov 2015 08:16:38 GMT	此次操作的时间，Date必须为GMT格式且不能为空。 注意 如果请求中的Date时间和OSS服务器的当前时间差15分钟以上，OSS服务器将拒绝该请求，并返回HTTP 403错误。
CanonicalizedOSSHeaders	字符串	否	x-oss-meta-a:a\nx-oss-meta-b:b\nx-oss-meta-c:c\n	以 `x-oss-` 为前缀的HTTP Header的字典序排列，可以为空。 ○ 如果设置CanonicalizedOSSHeaders为空，则无需在最后添加分隔符 `\n` 。 ○ 如果只有一个CanonicalizedOSSHeaders，则需要在最后添加分隔符 `\n` ，例如 `x-oss-meta-a\n` 。 ○ 如果有多个CanonicalizedOSSHeaders，则需要在每一个CanonicalizedOSSHeaders之后添加分隔符 `\n` ，例如 `x-oss-meta-a:a\nx-oss-meta-b:b\nx-oss-meta-c:c\n` 。 关于CanonicalizedOSSHeaders的构建方法，请参见构建CanonicalizedOSSHeaders的方法。
CanonicalizedResource	字符串	是	/examplebucket/	用户想要访问的OSS资源，不能为空。 关于CanonicalizedResource的构建方法，请参见构建CanonicalizedResource的方法。

- 签名示例

请求	签名字符串计算公式	签名字符串
PUT /nelson HTTP/1.0 Content-MD5: eB5eJF1ptWaXm4bijSPyxw== Content-Type: text/html Date: Thu, 17 Nov 2005 18:49:58 GMT Host: examplebucket.oss-cn-hangzhou.aliyuncs.com x-oss-meta-author: foo@example.com x-oss-meta-magic: abracadabra	Signature = base64(hmac-sha1(AccessKeySecret,VERB + “\n” + Content-MD5 + “\n” + Content-Type + “\n” + Date + “\n” + CanonicalizedOSSHeaders+ CanonicalizedResource))	“PUT\n eB5eJF1ptWaXm4bijSPyxw==\n text/html\n Thu, 17 Nov 2005 18:49:58 GMT\n x-oss-meta-magic:abracadabra\nx-oss-meta-author:foo@example.com\n/examplebucket/nelson

假设AccessKey ID为*LTAI4FixJvEPgvZ6g5cC****，AccessKey Secret为*Q0YehC6ZyugWfjod5y8Rcqrc1y****，此处以Python为例介绍计算签名（Signature）的方法。

```
import hmac
import hashlib
import base64
h = hmac.new("Q0YehC6ZyugWfjod5y8Rcqrc1y****".encode('utf-8'),
             "PUT\nODBGOERFMDMzQTczRUY3NUE3NzA5QzdFNUYzMDQxNEM\ntext/html\nThu, 17 Nov 2005 18:49:58 GMT\nx-oss-meta-magic:abracadabra\nx-oss-meta-author:foo@example.com\n/oss-example/nelson".encode('utf-8'), hashlib.sha1)
signature =  base64.encodestring(h.digest())
print(signature)
```

签名（Signature）计算结果为 `al2vHOlTWQdQtM6oqJAEZh7d****` ，结合Authorization头的结构组成，需要发送的消息体如下。

```
PUT /nelson HTTP/1.0
Authorization:OSS LTAI4FixJvEPgvZ6g5cC****:al2vHOlTWQdQtM6oqJAEZh7d****
Content-Md5: eB5eJF1ptWaXm4bijSPyxw==
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
x-oss-meta-author: foo@example.com
x-oss-meta-magic: abracadabra
```

- 相关说明

- 如果传入的AccessKey ID不存在或未激活，则返回403 Forbidden错误，错误码为InvalidAccessKeyId；如果AccessKey ID已激活，但OSS判断用户的请求发生签名错误，则返回403 Forbidden错误，并在返回的response中显示正确的用于验证加密的签名字符串。用户可以根据OSS的response来检查自己的签名字符串是否正确。

  返回示例如下：

```
<?xml version="1.0" ?>
<Error>
 <Code>
     SignatureDoesNotMatch
 </Code>
 <Message>
     The request signature we calculated does not match the signature you provided. Check your key and signing method.
 </Message>
 <StringToSignBytes>
     47 45 54 0a 0a 0a 57 65 64 2c 20 31 31 20 4d 61 79 20 32 30 31 31 20 30 37 3a 35 39 3a 32 35 20 47 4d 54 0a 2f 75 73 72 65 61 6c 74 65 73 74
3f 61 63 6c
 </StringToSignBytes>
 <RequestId>
     1E446260FF9B****
 </RequestId>
 <HostId>
     oss-cn-hangzhou.aliyuncs.***
 </HostId>
 <SignatureProvided>
     y5H7yzPsA/tP4+0tH1HHvPEwUv8=
 </SignatureProvided>
 <StringToSign>
     GET
Wed, 11 May 2011 07:59:25 GMT
/examplebucket?acl
 </StringToSign>
 <OSSAccessKeyId>
     AKIAIVAKMSMOY7VO****
 </OSSAccessKeyId>
</Error>
```

- 如果用户请求头中Authorization值的格式不对，则返回400 Bad Request错误，错误码为InvalidArgument。
- OSS所有的请求都必须使用HTTP 1.1协议规定的GMT时间格式。其中，日期的格式为：

```
date1 = 2DIGIT SP month SP 4DIGIT; day month year (e.g., 02 Jun 1982)
```

> **说明** 上述日期格式中 `day` 所占位数为2DIGIT，因此 `Jun 2` 、 `2 Jun 1982` 以及 `2-Jun-1982` 均为非法日期格式。

  - 如果签名验证时，Authorization头中没有传入Date或者Date，或者传入的格式不正确，则返回403 Forbidden错误，错误码为AccessDenied。
  - 传入请求的时间必须在OSS服务器当前时间之后的15分钟以内，否则返回403 Forbidden错误，错误码为RequestTimeTooSkewed。

构建CanonicalizedOSSHeaders的方法

所有以 `x-oss-` 为前缀的HTTP Header被称为CanonicalizedOSSHeaders，构建方法如下：

1. 将所有以 `x-oss-` 为前缀的HTTP请求头的名称转换为小写的形式，例如 `X-OSS-Meta-Name: TaoBao` 转换为 `x-oss-meta-name: TaoBao` 。
2. 如果以从STS服务获取的临时访问凭证发送请求时，您还需要将获得的security-token值以 `x-oss-security-token:security-token` 的形式加入到签名字符串中。

> **说明** 关于搭建STS服务的具体操作，请参见使用STS临时访问凭证访问OSS。您可以通过调用STS服务的AssumeRole接口或者使用各语言STS SDK来获取临时访问凭证。临时访问凭证包括临时访问密钥（AccessKey ID和AccessKey Secret）和安全令牌（SecurityToken）。

3. 将步骤1中获取的所有HTTP请求头按照名称的字典序进行升序排列。
4. 删除请求头和内容之间分隔符两端出现的任何空格。例如 `x-oss-meta-name: TaoBao` 转换为 `x-oss-meta-name:TaoBao` 。
5. 将每一个请求头和内容使用分隔符 `\n` 分隔拼成CanonicalizedOSSHeaders。

构建CanonicalizedResource的方法

用户发送请求中想访问的OSS目标资源被称为CanonicalizedResource，构建方法如下：

1. 将CanonicalizedResource置为空字符串 `""` 。
2. 设置要访问的OSS资源，格式为 `/BucketName/ObjectName` 。
   - 如果仅有BucketName而没有ObjectName，则CanonicalizedResource格式为*/BucketName/*。
   - 如果既没有BucketName也没有ObjectName，则CanonicalizedResource为正斜线（/）。
   - 如果请求的资源包括子资源（SubResource），则所有的子资源需按照字典序升序排列，并以 `&` 为分隔符生成子资源字符串。在CanonicalizedResource字符串尾添加 `?` 和子资源字符串。此时的CanonicalizedResource为 `/BucketName/ObjectName?acl&uploadId=UploadId` 。

     OSS支持以下四种类型的子资源：

     - 资源标识，例如acl、uploads、location、cors、logging、website、referer、lifecycle、delete、append、tagging、objectMeta、uploadId、partNumber、security-token、position、img、style、styleName、replication、replicationProgress、replicationLocation、cname、bucketInfo、comp、qos、live、status、vod、startTime、endTime、symlink、x-oss-process等。更多信息，请参见关于Bucket的操作和关于Object的操作。
     - 指定返回Header字段，例如response-content-type、response-content-language、response-expires、response-cache-control、response-content-disposition、response-content-encoding等。更多信息，请参见Get Object。
     - 图片处理操作方式，例如 `x-oss-process` 。更多信息，请参见图片处理。

- 以 `x-oss-ac-*` 开头的访问控制字段，例如x-oss-ac-source-ip、x-oss-ac-subnet-mask、x-oss-ac-vpc-id、x-oss-ac-forward-allow。更多信息，请参见在URL中包含签名。

> **说明** 使用包含x-oss-ac-source-ip参数的CanonicalizedResource生成签名后，请从需要发送的消息query参数中移除x-oss-ac-source-ip，以保护IP地址信息安全。

计算签名头规则

- 签名的字符串必须为 `UTF-8` 格式。含有中文字符的签名字符串必须先进行 `UTF-8` 编码，再与 `AccessKeySecret` 计算最终签名。
- 签名的方法用RFC 2104中定义的HMAC-SHA1方法，其中Key指的是AccessKey Secret。
- `Content-Type` 和 `Content-MD5` 在请求中不是必须的，如果请求需要签名验证，空值请以换行符 `\n` 代替。
- 在所有非HTTP标准定义的Header中，只有以 `x-oss-` 开头的Header需要加入签名字符串（例如签名示例中的x-oss-meta-magic则需要加入签名字符串）；其他非HTTP标准Header将被OSS忽略。

> **说明** 以 `x-oss-` 开头的Header在签名验证前需要符合以下规范：
> - Header的名称需要转为小写。
> - Header按字典序升序排序。
> - 分割Header的name和value之间的冒号前后不能有空格。
> - 每个Header之后都要有一个换行符“\n”，如果没有Header，则设置CanonicalizedOSSHeaders为空。

Content-MD5的计算方法

以消息内容“123456789”为例，以下详细说明正确及错误计算该字符串的Content-MD5的方法。

- 正确计算示例
  i. 先计算MD5加密的二进制数组（128位）。
  ii. 对该二进制数组进行base64编码（而不是对32位字符串编码）。

  以Python为例：

```
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")   //在Python 3中此处需要改为hash.update(b"0123456789")。
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
```

  hash.digest()，计算出二进制数组（128位）。

```
>>> hash.digest()
'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
```

- 错误计算示例

> **说明** 常见错误是直接对计算出的32位字符串进行base64编码。

```
# hash.hexdigest()，计算得到可见的32位字符串编码。
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
# 错误的MD5值进行base64编码后的结果。
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

7.5.3. 在URL中包含签名

除了使用Authorization Header，您还可以在URL中加入签名信息，以便将该URL转给第三方实现授权访问。

注意事项

- 使用在URL中签名的方式，会将授权的数据在过期时间内曝露在互联网上，请预先评估使用风险。
- OSS不支持同时在URL和Header中包含签名。
- PUT和GET请求都支持在URL中签名。
- 您可以为PUT操作生成一个预签名的URL，该URL检查用户是否上传了正确的内容。SDK对请求进行预签名时，将计算请求正文的校验和，并生成包含在预签名URL中的MD5校验和。用户必须上传与SDK生成的MD5校验和相同的内容，否则操作失败。如果要验证MD5，只需在请求中增加Content-MD5头即可。

签名实现

- 签名示例

```
https://examplebucket.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?OSSAccessKeyId=nz2pc56s936****&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272Y
EAiv****
```

  如果需要使用STS用户构造URL签名，则必须携带 `security-token` 。

```
https://examplebucket.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?OSSAccessKeyId=nz2pc56s936****&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272Y
EAiv****&security-token=CAISowJ1q6Ft5B2yfSjIr5bgIOz31blR9oWmWBfCs3kDR/xm3Imc1zz2IHxMdHJsCeAcs/Q0lGFR5/sflqJIRoReREvCUcZr8szfWcsZos2T1fau5Jko1be0ewH
KeQKZsebWZ+LmNpy/Ht6md1HDkAJq3LL+bk/Mdle5MJqP+/kFC9MMRVuAcCZhDtVbLRcYgq18D3bKMuu3ORPHm3fZCFES2jBxkmRi86+ysIP+phPVlw/90fRH5dazcJW0Zsx0OJo6Wcq+3+FqM6
DQlTNM6hwNtoUO1fYUommb54nDXwQIvUjfbtC5qIM/cFVLAYEhALNBofTGkvl1h/fejYyfyWwWYbkFCHiPFNr9kJCUSbr4a4sjF6zyPnPWycyCLYXleLzhxPWd/2kagAGaXG69BqwYNvrKKI3W8
weP3bNc1wQDMXQfiHpFCRG6lYhh3iXFtpwH90A3sTlxzRGvi8+9p63JwrluOHWs+Fj6S6s0cOhKvKRWYE8UuWeXIvv4l6DAGwHDE8BLjLC11f5prUJgI2wb+3hwuBod32Jx+us/1p996Glao725
orcb****
```

  您可以在签名URL中添加想要授权的IP地址、IP地址段或VPC ID，避免未授权的终端访问OSS资源。

```
https://examplebucket.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?&OSSAccessKeyId=44CF9590006BF252F707&Expires=1475462111&Signature=OjxkTUbT6rXCZcW8Qh
vlrXQr8vsP80EFdo6oG5qsBx****&x-oss-ac-subnet-mask=32
```

- 参数说明

名称	类型	是否必选	描述
OSSAccessKeyId	字符串	是	指定URL签名中使用的AccessKey ID。
Expires	数字	是	Unix时间戳（自UTC时间1970年01月01号开始的秒数），用于标识该URL的超时时间。如果OSS接收到该URL请求的时间晚于签名中包含的Expires参数时，则返回请求超时的错误码。例如，当前时间是1141889060，开发者希望创建一个60秒后自动失效的URL，则可以设置Expires时间为1141889120。 说明 出于安全考虑，OSS控制台中默认URL的有效时间为3600秒，最大值为32400秒。关于修改URL超时时间的具体操作，请参见分享文件。
Signature	字符串	是	签名信息。格式如下： `Signature = urlencode(base64(hmac-sha1(AccessKeySecret, VERB + "\n" + CONTENT-MD5 + "\n" + CONTENT-TYPE + "\n" + EXPIRES + "\n" + CanonicalizedOSSHeaders + CanonicalizedResource)))` ○ 所有OSS支持的请求和各种Header参数，在URL中进行签名的算法和在Header中包含签名的算法类似。 ○ 生成URL中的签名字符串时，除了将Date参数替换为Expires参数外，仍然包含 `CONTENT-TYPE` 、 `CONTENT-MD5` 、 `CanonicalizedOSSHeaders` 等在Header中包含签名中定义的Header（请求中虽然仍有Date请求Header，但无需将Date加入签名字符串中）。 ○ 在URL中包含签名时必须对URL进行编码。如果在URL中多次传入Signature、Expires或OSSAccessKeyId，则以第一次传入的值为准。 ○ 使用URL签名时，OSS会先验证请求时间是否晚于Expires时间，然后再验证签名。
security-token	字符串	否	安全令牌。只有当使用STS用户构造URL签名时，才需要设置此参数。 说明 关于搭建STS服务的具体操作，请参见使用STS临时访问凭证访问OSS。您可以通过调用STS服务的AssumeRole接口或者使用各语言STS SDK来获取临时访问凭证。临时访问凭证包括临时访问密钥（AccessKey ID和AccessKey Secret）和安全令牌（SecurityToken）。
x-oss-ac-source-ip	字符串	否	指定IP地址或者IP地址段。如果在生成签名时添加了IP地址或IP地址段，则需要添加参数x-oss-ac-subnet-mask，用于标记子网掩码。
x-oss-ac-subnet-mask	数字	否	子网掩码中1的个数。如果请求中带有该参数，OSS会将实际请求IP地址与子网掩码进行求与，然后用于计算签名是否正确。如果该参数被恶意篡改，将导致签名无法校验通过。
x-oss-ac-vpc-id	字符串	否	指定VPC ID。指定该参数后，OSS会判断是否为对应VPC ID来源的请求。如果请求是从该VPC ID发起且该参数已赋值，则同时校验VPC ID和来源IP地址或IP地址段。
x-oss-ac-forward-allow	布尔型	否	指定是否允许转发请求。OSS如果检测到该字段并且请求中带有 `X-Forwarded-For` （可能为多个IP地址），则将 `X-Forwarded-For` 的值用于计算签名校验。 取值如下： ○ true：表示允许转发请求。 注意 设置为true存在请求头被篡改劫持的风险。 ○ false（默认值）：不允许转发请求。

- 生成签名的Python示例代码
  - 示例一（只涉及必选参数）

```
import base64
import hmac
import hashlib
import urllib
h = hmac.new("accesskey",
             "GET\n\n\n1141889120\n/examplebucket/oss-api.pdf",
             hashlib.sha1)
urllib.quote (base64.encodestring(h.digest()).strip())
```

- 示例二（包含可选参数）
  - 签名时，需按照字典序升序排列可选的签名参数。如果参数未配置则为空。
  - 当您需要指定具体IP地址访问时，只需在生成签名时添加IP地址（例如 `x-oss-ac-source-ip=127.0.0.1` ），并在query参数中提供子网掩码用于计算签名（例如 `x-oss-ac-subnet-mask=32` ）。

```
import base64
import hmac
import hashlib
import urllib
h = hmac.new(accesskey,
             "GET\n\n\n1141889120\n%2Fexamplebucket%2Foss-api.pdf?\
             &x-oss-ac-forward-allow=true\
             &x-oss-ac-source-ip=127.0.0.1\
             &x-oss-ac-subnet-mask=32\
             &x-oss-signature-version=OSS2",
             hashlib.sha256)
Signature = base64.encodestring(h.digest()).strip()
```

各语言SDK的URL签名实现

SDK	URL签名方法	实现文件
Java SDK	OSSClient.generatePresignedUrl	OSSClient.java
Python SDK	Bucket.sign_url	api.py
PHP SDK	OssClient.signUrl	OssClient.php
Node.js SDK	signatureUrl	Object.js
Browser.js SDK	signatureUrl	signatureUrl
Android SDK	OSSClient.presignConstrainedObjectURL	OSSClient.java
iOS SDK	presignConstrainURLWithBucketName:withObjectKey:httpMethod:withExpirationInterval:withParameters	OSSClient.h
Go SDK	signURL	signURL
C SDK	oss_gen_signed_url	oss_object.c
C++ SDK	OssClient::GeneratePresignedUrl	OssClient.cc
.Net SDK	OssClient.GeneratePresignedUri	OssClient.cs

各语言SDK签名URL的使用示例

通过签名URL上传或下载文件的各语言SDK示例如下：

- Java
- Python
- PHP
- Go
- C
- C++
- .NET
- Node.js
- Browser.js
- Android
- iOS

错误码

错误码	返回消息	描述
AccessDenied	403 Forbidden	在URL中添加签名时，Signature、Expires和OSSAccessKeyId顺序可以调换，但缺少Signature、Expires或OSSAccessKeyId中的一个或者多个。
AccessDenied	403 Forbidden	访问的当前时间晚于请求中设定的Expires时间或时间格式错误。
InvalidArgument	400 Bad Request	URL中包含Signature、Expires、OSSAccessKeyId中的一个或者多个，并且Header中也包含签名消息。

7.5.4. 签名常见问题

本文档主要介绍 OSS 签名过程中的常见问题及解决方法。

计算签名失败，报错“The request signature we calculated does not match the signature you provided”

OSS 允许在 Header 中包含签名或在 URL 中包含签名，这两种签名方式的区别如下：

Header	URL
不支持设置 expires	支持设置 expires
常用 Method 包括： GET、POST、PUT、DELETE	常用 Method 包括：GET、PUT
date 时间是 GMT 格式	date 替换成 expires 变成时间戳
signature 不需要 URL 编码	signature 需要 URL 编码

通过在 Header 或 URL 中自签名计算 signature 时， 经常遇到签名失败，报错 “The request signature we calculated does not match the signature you provided” 。以下 demo 演示了如何调用 API 自签名时上传 Object 到 OSS。

```
#! /us/bin/envy python
#Author: hanli
#Update: 2018-09-29
from optparse import OptionParser
import urllib, urllib2
import datetime
import base64
import hmac
import sha
import os
import sys
import time
class Main():
  # Initial input parse
  def __init__(self,options):
    self.ak = options.ak
    self.sk = options.sk
    self.ed = options.ed
    self.bk = options.bk
    self.fi = options.fi
    self.oj = options.objects
    self.left = '\033[1;31;40m'
    self.right = '\033[0m'
    self.types = "application/x-www-form-urlencoded"
    self.url = 'http://{0}.{1}/{2}'.format(self.bk,self.ed,self.oj)
  # Check client input parse
  def CheckParse(self):
    if (self.ak and self.sk and self.ed and self.bk and self.oj and self.fi) != None:
      if str(self.ak and self.sk and self.ed and self.bk and self.oj and self.fi):
        self.PutObject()
    else:
      self.ConsoleLog("error","Input parameters cannot be empty")
  # GET local GMT time
  def GetGMT(self):
    SRM = datetime.datetime.utcnow()
    GMT = SRM.strftime('%a, %d %b %Y %H:%M:%S GMT')
    return GMT
  # GET Signature
  def GetSignature(self):
    mac = hmac.new("{0}".format(self.sk),"PUT\n\n{0}\n{1}\n/{2}/{3}".format(self.types,self.GetGMT(),self.bk,self.oj), sha)
    Signature = base64.b64encode(mac.digest())
    return Signature
  # PutObject
  def PutObject(self):
    try:
      with open(self.fi) as fd:
        files = fd.read()
    except Exception as e:
      self.ConsoleLog("error",e)
    try:
      request = urllib2.Request(self.url, files)
      request.add_header('Host','{0}.{1}'.format(self.bk,self.ed))
      request.add_header('Date','{0}'.format(self.GetGMT()))
      request.add_header('Authorization','OSS {0}:{1}'.format(self.ak,self.GetSignature()))
      request.get_method = lambda:'PUT'
      response = urllib2.urlopen(request,timeout=10)
      fd.close()
      self.ConsoleLog(response.code,response.headers)
    except Exception,e:
      self.ConsoleLog("error",e)
  # output error log
  def ConsoleLog(self,level=None,mess=None):
    if level == "error":
      sys.exit('{0}[ERROR:]{1}{2}'.format(self.left,self.right,mess))
    else:
      sys.exit('\nHTTP/1.1 {0} OK\n{1}'.format(level,mess))
if __name__ == "__main__":
  parser = OptionParser()
  parser.add_option("-i",dest="ak",help="Must fill in Accesskey")
  parser.add_option("-k",dest="sk",help="Must fill in AccessKeySecrety")
  parser.add_option("-e",dest="ed",help="Must fill in endpoint")
  parser.add_option("-b",dest="bk",help="Must fill in bucket")
  parser.add_option("-o",dest="objects",help="File name uploaded to oss")
  parser.add_option("-f",dest="fi",help="Must fill localfile path")
  (options, args) = parser.parse_args()
  handler = Main(options)
  handler.CheckParse()
```

请求头：

```
PUT /yuntest HTTP/1.1
Accept-Encoding: identity
Content-Length: 147
Connection: close
User-Agent: Python-urllib/2.7
Date: Sat, 22 Sep 2018 04:36:52 GMT
Host: yourBucket.oss-cn-shanghai.aliyuncs.com
Content-Type: application/x-www-form-urlencoded
Authorization: OSS B0g3mdt:lNCA4L0P43Ax
```

响应头：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Sat, 22 Sep 2018 04:36:52 GMT
Content-Length: 0
Connection: close
x-oss-request-id: 5BA5C6E4059A3C2F
ETag: "D0CAA153941AAA1CBDA38AF"
x-oss-hash-crc64ecma: 8478734191999037841
Content-MD5: 0MqhU5QbIp3Ujqqhy9o4rw==
x-oss-server-time: 15
```

说明

- Signature 中所有加入计算的参数都要放在 Header 中，Header 和 Signature 需保持一致。有关要签名的 Header，请参见在 Header 中包含签名。
- 通过 PUT 上传时，signature 计算的 Content-type 可以为 application/x-www-form-urlencoded 。
- 通过 Header 方式进行签名认证时无法设置 expires。只有通过 SDK 和控制台设置签名 URL 时可以设置 expires。

通过微信小程序请求 OSS 返回签名失败，通过浏览器请求 OSS 返回签名正常

通过浏览器访问时的 HTTP 抓包数据如下图所示。

```
GET /1530903286 OSSAccessKeyId=E                    xpires=1540915205&Signature=XDdcDn8%2r                    kfB0%3D HTTP/
1.1
Accept: */*
C-Seq: 123
Cache-Control: no-cache
Connection: Keep-Alive
Content-Type: application/octet-stream
Host:
Range: bytes=0-262143
User-Agent: MicroMessenger Client

HTTP/1.1 403 Forbidden
Server: Tengine
Content-Type: application/xml
Content-Length: 785
Connection: keep-alive
Date: Fri, 12 Oct 2018 09:47:19 GMT
x-oss-request-id: 5BC06DA78E
x-oss-server-time: 0
Via: cache2.l2cm10-1[27,403-1280,M], cache27.l2cm10-1[29,0], cache2.cn1453[239,403-1280,M], cache8.cn1453[243,0]
X-Swift-Error: orig response 4XX error
Ali-Swift-Global-Savetime: 1539337639
X-Cache: MISS TCP_MISS dirn:-2:-2
X-Swift-SaveTime: Fri, 12 Oct 2018 09:47:19 GMT
X-Swift-CacheTime: 0
X-Swift-Error: orig response 4XX error
Timing-Allow-Origin: *
EagleId: 7488861c15393376395951725e
```

- 通过反复对比 403 和 200 的抓包数据发现通过微信小程序发出的 HTTP 请求和浏览器发起的 HTTP 请求的 URL 、signature、expires 相同，区别在于微信小程序携带了 Content-type ，而通过浏览器的请求没有携带 Content-type。
- signature 计算时没有包含 Content-tpye ，而微信小程序发起的请求携带了 Content-type 。OSS 收到请求后会按照携带了 Content-type 的方式来计算 signature ，导致计算结果不一致。

遇到类似问题，建议抓包排查。如果 OSS 请求 Header 中携带了 Content-type，则 signature 计算也要加上 Content-type。

通过 OSS 多个语言版本 SDK 测试发现，结合 CDN 使用 OSS 时，客户端使用 CDN 域名计算 signature，并发起 Head 请求时，OSS 收到请求后返回 403 ；

```
Loaded assembly: Microsoft.GeneratedCode [External]
System.Net.WebException: The remote server returned an error: (403) Forbidden.
  at Aliyun.OSS.Common.Communication.ServiceResponse.EnsureSuccessful () [0x00021] in <d8db38433d62487181c
  at Aliyun.OSS.Common.Handlers.ErrorResponseHandler.ErrorHandle (Aliyun.OSS.Common.Communication.ServiceR
  at Aliyun.OSS.Common.Handlers.ErrorResponseHandler.Handle (Aliyun.OSS.Common.Communication.ServiceRespon
  at Aliyun.OSS.Common.Communication.ServiceClient.HandleResponse (Aliyun.OSS.Common.Communication.Service
  at Aliyun.OSS.Common.Communication.ServiceClient.Send (Aliyun.OSS.Common.Communication.ServiceRequest re
  at Aliyun.OSS.Common.Communication.RetryableServiceClient.SendImpl (Aliyun.OSS.Common.Communication.Serv
  at Aliyun.OSS.Common.Communication.RetryableServiceClient.Send (Aliyun.OSS.Common.Communication.ServiceR
  at Aliyun.OSS.Commands.OssCommand.Execute () [0x00014] in <d8db38433d62487181cd2ba5c760defd>:0
  at Aliyun.OSS.Commands.OssCommand`1[T].Execute () [0x00000] in <d8db38433d62487181cd2ba5c760defd>:0
  at Aliyun.OSS.OssClient.GetObjectMetadata (System.String bucketName, System.String key) [0x0001f] in <d8
```

通过 tcpdump 抓包或者 Wireshark 对比得知，由于客户端发起的 Head 请求在通过 CDN 回源到 OSS 时，CDN 回源用的是 GET 请求，OSS 接收到该请求时用 GET 请求方式来计算 signature，得到的结果与客户端计算不一致，该问题可以升级阿里云 CDN 处理。

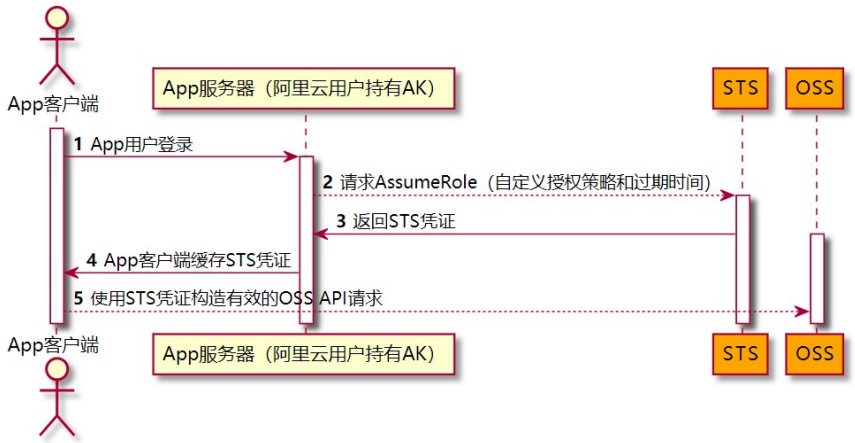
7.6. 使用STS临时访问凭证访问OSS

您可以通过STS服务给其他用户颁发一个临时访问凭证。该用户可使用临时访问凭证在规定时间内访问您的OSS资源。临时访问凭证无需透露您的长期密钥，使您的OSS资源访问更加安全。

适用场景

假设您是一个移动App开发者，希望使用阿里云OSS服务来保存App的终端用户数据，并且要保证每个App用户之间的数据隔离。此时，您可以使用STS授权用户直接访问OSS。

使用STS授权用户直接访问OSS的流程如下：



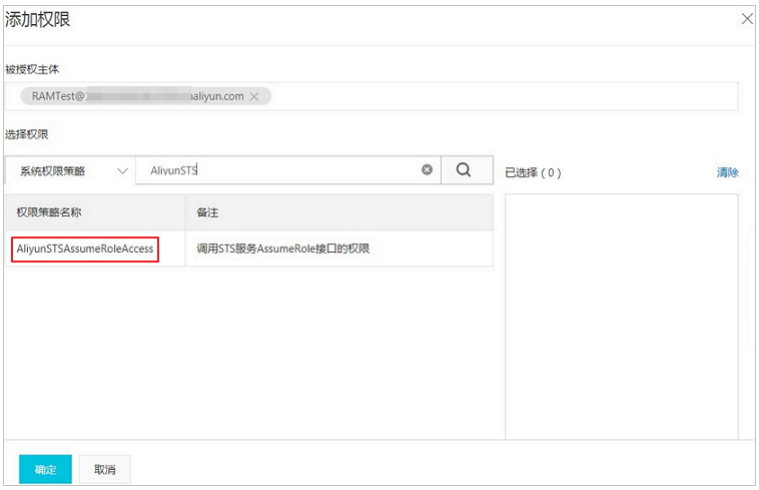
1. App用户登录。App用户和云账号无关，它是App的终端用户，App服务器支持App用户登录。对于每个有效的App用户来说，需要App服务器能定义出每个App用户的最小访问权限。
2. App服务器请求STS服务获取一个安全令牌（SecurityToken）。在调用STS之前，App服务器需要确定App用户的最小访问权限（用RAM Policy来自定义授权策略）以及凭证的过期时间。然后通过扮演角色（AssumeRole）来获取一个代表角色身份的安全令牌（SecurityToken）。
3. STS返回给App服务器一个临时访问凭证，包括一个安全令牌（SecurityToken）、临时访问密钥（AccessKeyId和AccessKeySecret）以及过期时间。
4. App服务器将临时访问凭证返回给App客户端，App客户端可以缓存这个凭证。当凭证失效时，App客户端需要向App服务器申请新的临时访问凭证。例如，临时访问凭证有效期为1小时，那么App客户端可以每30分钟向App服务器请求更新临时访问凭证。
5. App客户端使用本地缓存的临时访问凭证去请求OSS API。OSS收到访问请求后，会通过STS服务来验证访问凭证，正确响应用户请求。

步骤一：创建RAM用户

1. 登录RAM控制台。
2. 在左侧导航栏，选择**身份管理 > 用户**。
3. 单击**创建用户**。
4. 输入**登录名称**和**显示名称**。
5. 在**访问方式**区域下，选择**Open API 调用访问**，然后单击**确定**。
6. 单击**复制**，保存访问密钥（AccessKey ID 和 AccessKey Secret）。

步骤二：为RAM用户授予请求AssumeRole的权限

1. 单击已创建RAM用户右侧对应的**添加权限**。
2. 在**添加权限**页面，选择**AliyunSTSAssumeRoleAccess**系统策略。



3. 单击**确定**。

步骤三：创建用于获取临时访问凭证的角色

1. 在左侧导航栏，选择**身份管理 > 角色**。
2. 单击**创建角色**，选择可信实体类型为**阿里云账号**，单击**下一步**。

3. **角色名称**填写为*RamOssTest*，**选择云账号**为**当前云账号**。
4. 单击**完成**。角色创建完成后，单击**关闭**。
5. 在**RAM角色管理**页面，搜索框输入角色名称*RamOssTest*。
6. 单击**复制**，保存角色的ARN。

步骤四：为角色授予上传文件的权限

1. 创建上传文件的自定义权限策略。
   i. 在左侧导航栏，选择**权限管理 > 权限策略**。
   ii. 单击**创建权限策略**。
   iii. 在**创建权限策略**页面，单击**脚本编辑**，然后在策略文档输入框中赋予角色向目标存储空间examplebucket下的目录exampledir上传文件的权限。具体配置示例如下。

   > **警告** 以下示例仅供参考。您需要根据实际需求配置更细粒度的授权策略，防止出现权限过大的风险。关于更细粒度的授权策略配置详情，请参见通过RAM或STS服务向其他用户授权。

   ```
   {
       "Version": "1",
       "Statement": [
        {
              "Effect": "Allow",
              "Action": [
                "oss:PutObject"
              ],
              "Resource": [
                "acs:oss:*:*:examplebucket/exampledir",
                "acs:oss:*:*:examplebucket/exampledir/*"
              ]
        }
       ]
   }
   ```

   iv. 策略配置完成后，单击**下一步**。
   v. 在**基本信息**区域，填写策略**名称**为*RamTestPolicy*，然后单击**确定**。
2. 为RAM角色*RamOssTest*授予自定义权限策略。
   i. 在左侧导航栏，选择**身份管理 > 角色**。
   ii. 在**角色**页面，找到目标RAM角色*RamOssTest*。
   iii. 单击RAM角色*RamOssTest*右侧的**添加权限**。
   iv. 在**添加权限**页面下的**自定义策略**页签，选择已创建的自定义权限策略*RamTestPolicy*。
   v. 单击**确定**。

步骤五：获取临时访问凭证

您可以通过调用STS服务接口AssumeRole 或者使用各语言STS SDK来获取临时访问凭证。

以下代码用于获取临时访问凭证：

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.profile.IClientProfile;
import com.aliyuncs.sts.model.v20150401.AssumeRoleRequest;
import com.aliyuncs.sts.model.v20150401.AssumeRoleResponse;
public class StsServiceSample {
    public static void main(String[] args) {
        // STS接入地址，例如sts.cn-hangzhou.aliyuncs.com。
        String endpoint = "<sts-endpoint>";
        // 填写步骤1生成的访问密钥AccessKey ID和AccessKey Secret。
        String AccessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        // 填写步骤3获取的角色ARN。
        String roleArn = "<yourRoleArn>";
        // 自定义角色会话名称，用来区分不同的令牌，例如可填写为SessionTest。
        String roleSessionName = "<yourRoleSessionName>";
        // 以下Policy用于限制仅允许使用临时访问凭证向目标存储空间examplebucket上传文件。
        // 临时访问凭证最后获得的权限是步骤4设置的角色权限和该Policy设置权限的交集，即仅允许将文件上传至目标存储空间examplebucket下的exampledir目录。
        String policy = "{\n" +
                "    \"Version\": \"1\", \n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Action\": [\n" +
                "                \"oss:PutObject\"\n" +
                "            ], \n" +
                "            \"Resource\": [\n" +
                "                \"acs:oss:*:*:examplebucket/*\" \n" +
                "            ], \n" +
                "            \"Effect\": \"Allow\"\n" +
                "        }\n" +
                "    ]\n" +
                "}";
        try {
            // regionId表示RAM的地域ID。以华东1（杭州）地域为例，regionID填写为cn-hangzhou。也可以保留默认值，默认值为空字符串（""）。
            String regionId = "";
            // 添加endpoint。适用于Java SDK 3.12.0及以上版本。
            DefaultProfile.addEndpoint(regionId, "Sts", endpoint);
            // 添加endpoint。适用于Java SDK 3.12.0以下版本。
            // DefaultProfile.addEndpoint("",regionId, "Sts", endpoint);
            // 构造default profile。
            IClientProfile profile = DefaultProfile.getProfile(regionId, AccessKeyId, accessKeySecret);
            // 构造client。
            DefaultAcsClient client = new DefaultAcsClient(profile);
            final AssumeRoleRequest request = new AssumeRoleRequest();
            // 适用于Java SDK 3.12.0及以上版本。
            request.setSysMethod(MethodType.POST);
            // 适用于Java SDK 3.12.0以下版本。
            //request.setMethod(MethodType.POST);
            request.setRoleArn(roleArn);
            request.setRoleSessionName(roleSessionName);
            request.setPolicy(policy); // 如果policy为空，则用户将获得该角色下所有权限。
            request.setDurationSeconds(3600L); // 设置临时访问凭证的有效时间为3600秒。
            final AssumeRoleResponse response = client.getAcsResponse(request);
            System.out.println("Expiration: " + response.getCredentials().getExpiration());
            System.out.println("Access Key Id: " + response.getCredentials().getAccessKeyId());
            System.out.println("Access Key Secret: " + response.getCredentials().getAccessKeySecret());
            System.out.println("Security Token: " + response.getCredentials().getSecurityToken());
            System.out.println("RequestId: " + response.getRequestId());
        } catch (ClientException e) {
            System.out.println("Failed: ");
            System.out.println("Error code: " + e.getErrCode());
            System.out.println("Error message: " + e.getErrMsg());
            System.out.println("RequestId: " + e.getRequestId());
        }
    }
}
```

说明

- 临时访问凭证有效时间单位为秒，最小值为900，最大值以当前角色设定的最大会话时间为准。详情请参见设置角色最大会话时间。
- 有关角色会话名称 `roleSessionName` 的命名规范，请参见AssumeRole。

步骤六：使用临时访问凭证上传文件至OSS

以Java SDK 3.12.0版本为例，将本地 `D:\\localpath` 路径下的examplet est.txt文件上传至存储空间examplebucket下的exampledir目录的示例代码如下：

```
import com.aliyun.oss.OSSClient;
import com.aliyun.oss.model.PutObjectRequest;
import java.io.File;
public class Upload {
    public static void main(String[] args) {
// Endpoint以杭州为例，其它Region请按实际情况填写。
String endpoint = "oss-cn-hangzhou.aliyuncs.com";
// 填写步骤五获取的临时访问密钥（AccessKey ID和AccessKey Secret）。
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// 填写步骤五获取的安全令牌SecurityToken。
String securityToken = "<yourSecurityToken>";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
// 将本地文件exampletest.txt上传至目标存储空间examplebucket下的目录exampledir。
PutObjectRequest putObjectRequest = new PutObjectRequest("examplebucket", "exampledir/exampletest.txt", new File("D:\\localpath\\exampletest.txt"));
// 如果需要上传时设置存储类型与访问权限，请参考以下示例代码。
// ObjectMetadata metadata = new ObjectMetadata();
// metadata.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard.toString());
// metadata.setObjectAcl(CannedAccessControlList.Private);
// putObjectRequest.setMetadata(metadata);
// 上传文件。
ossClient.putObject(putObjectRequest);
// 关闭OSSClient。
ossClient.shutdown();
    }
}
```

更多语言的SDK示例请参见：

- Java SDK
- Python SDK
- Go SDK
- C++ SDK
- PHP SDK
- C SDK
- .NET SDK
- Node.js SDK
- Browser.js SDK
- Android SDK
- iOS SDK

Object上传成功后，关于如何获取访问URL的具体操作，请参见上传Object后如何获取访问URL？。

常见问题

- 报错 `The security token you provided is invalid.` 如何处理？

  请确保完整填写步骤五获取到的SecurityToken。

- 报错 `The OSS Access Key Id you provided does not exist in our records.` 如何处理？

  临时访问凭证已过期，过期后自动失效。请使用临时访问密钥（AccessKeyId和AccessKeySecret）向App服务器申请新的临时访问凭证。具体操作，请参见步骤五。

- 获取STS时报错 `NoSuchBucket` 如何处理？

  出现这种报错通常是STS的 `endpoint` 填写错误。请根据您的地域，填写正确的STS接入地址。各地域的STS接入地址请参见接入地址。

7.7. 防盗链

OSS支持对存储空间（Bucket）设置防盗链，即通过对访问来源设置白名单的机制，避免OSS资源被其他人盗用。

功能介绍

防盗链功能通过设置Referer白名单以及是否允许空Referer，限制仅白名单中的域名可以访问您Bucket内的资源。OSS支持基于HTTP和HTTPS header中表头字段Referer的方法设置防盗链。

是否进行防盗链验证的具体场景如下：

- 仅当通过签名URL或者匿名访问Object时，进行防盗链验证。
- 当请求的Header中包含 `Authorization` 字段，不进行防盗链验证。

防盗链通过请求Header中的Referer地址判断访问来源。当浏览器向Web服务器发送请求的时候，请求Header中将包含Referer，用于告知Web服务器该请求的页面链接来源。OSS根据浏览器附带的Referer与用户配置的Referer规则来判断允许或拒绝此请求，如果Referer一致，则OSS将允许该请求的访问；如果Referer不一致，则OSS将拒绝该请求的访问。例如，某个Bucket设置了Referer为*https://10.10.10.10*：

- 用户A在*https://10.10.10.10*嵌入test.jpg图片，当浏览器请求访问此图片时会带上*https://10.10.10.10*的Referer，此场景下OSS将允许该请求的访问。
- 用户B盗用了test.jpg的图片链接并将其嵌入*https://192.168.0.0*，当浏览器请求访问此图片时会带上*https://192.168.0.0*的Referer，此场景下OSS将拒绝该请求的访问。

Referer配置规则

- 单个Bucket支持配置多个Referer。通过控制台设置Referer时，使用回车作为换行符分隔；通过API设置Referer时，使用英文逗号（,）分隔。
- Referer支持使用通配符星号（*）和问号（?）。
  - 通配符星号（*）表示使用星号代替0个或多个字符。如果Referer白名单配置为**.aliyun.com*，且不允许空Referer的情况下，则只有HTTP或HTTPS header中包含Referer字段的请求才能访问OSS资源，例如*help.aliyun.com*、*www.aliyun.com*等；如果Referer白名单配置为**.aliyun.com*，且允许空Referer的情况下，则Referer为空的请求也允许访问OSS资源。

- 通配符问号（?）表示使用问号代替一个字符。如果设置了包含通配符问号（?）的Referer白名单，且不允许空Referer的情况下，则只有HTTP或HTTPS Header中包含Referer字段的请求才能访问OSS资源。如果设置了包含通配符问号（?）的Referer白名单，且允许空Referer的情况下，则Referer为空的请求也允许访问OSS资源。

- Referer支持使用反斜线（\）对通配符星号（*）和问号（?）进行转义。
- Referer默认截断QueryString。如有特殊需求，可设置不允许截断QueryString。

  例如，Referer设置为*http://www.example.com/?action=nop*，由于OSS匹配该Referer时默认截断QueryString，即使用*http://www.example.com/*进行匹配。如果希望OSS使用*http://www.example.com/?action=nop*进行Referer匹配，请通过设置AllowTruncateQueryString参数为*false*实现不允许截断QueryString。关于设置不允许截断QueryString的具体操作，请参见PutBucketReferer。

  设置不允许截断QueryString时，Referer匹配规则有以下注意事项：

  - 不解码QueryString

    通过*http://www.example.com/?job_id=task$01*访问OSS时，可能该访问URL会被继续保留为*http://www.example.com/?job_id=task$01*，也可能被转义为*http://www.example.com/?job_id=task%2401*。通过Referer进行匹配时，不会对访问URL中的QueryString进行解码，示例如下：

    - 当Referer设置为*http://www.example.com/?job_id=task%2401*的情况下，通过*http://www.example.com/?job_id=task$01*请求访问OSS时，OSS将拒绝该请求。
    - 当Referer设置为*http://www.example.com/?job_id=task$01*的情况下，通过*http://www.example.com/?job_id=task%2401*请求访问OSS时，OSS将拒绝该请求。

  - 忽略QueryString中的参数大小写

    通过Referer进行匹配时，会忽略QueryString中的大小写。即当Referer设置为*http://www.example.com/?action=nop*的情况下，通过*http://www.examplecom/?ACTION=NOP*请求访问OSS时，OSS将允许该请求。

  - 不解析QueryString中的参数

    默认情况下，浏览器会将*http://example.com/?a=b&b=c*与*http://example.com/?b=c&a=b*视为相同的访问URL。但是通过Referer进行匹配时，不会对QueryString中的参数进行解析，因此*http://example.com/?a=b&b=c*与*http://example.com/?b=c&a=b*会被视为不同的访问URL。即当Referer设置为*http://example.com/?a=b&b=c*的情况下，通过*http://example.com/?b=c&a=b*请求访问OSS时，OSS将拒绝该请求。

Referer配置效果

- 如果Referer白名单为空，则所有的请求都会被允许。
- 如果Referer白名单不为空，且不允许空Referer，则只有Referer属于白名单的请求被允许，其他请求（包括Referer为空的请求）会被拒绝。
- 如果Referer白名单不为空，但允许空Referer，则Referer为空的请求和符合白名单的请求会被允许，其他请求都会被拒绝。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**权限管理 > 防盗链**。
4. 在**防盗链**区域，单击**设置**。
   - 在**Referer**框中，填写域名或IP地址，支持通配符星号（*）和问号（?），多个Referer以换行分隔。示例如下：
     - 配置为 `www.aliyun.com` ，可匹配如*www.aliyun.com/123*、*www.aliyun.com.cn*等以*www.aliyun.com*为前缀的地址。
     - 通配符星号（*）表示使用星号代替0个或多个字符。例如配置为 `*www.aliyun.com/` ，可匹配如*http://www.aliyun.com/*和*https://www.aliyun.com/*地址。配置为 `*.aliyun.com` ，可匹配如*help.aliyun.com*、*www.aliyun.com*等地址。
     - 通配符问号（?）表示使用问号代替一个字符。
     - 支持带端口的域名或IP地址，例如*www.example.com:8080*、*10.10.10.10:8080*等地址。
   - 在**允许空Referer**框中，选择是否允许Referer为空。

     空Referer表示HTTP或HTTPS请求中，不带Referer字段或Referer字段为空。

     如果不允许空Referer，则只有HTTP或HTTPS Header中包含Referer字段的请求才能访问OSS资源。

     > **说明** 当您使用OSS的Bucket域名（如bucketname.oss-cn-zhangjiakou.aliyuncs.com）预览MP4文件时，由于浏览器默认会同时发出两个请求，其中一个为带Referer的请求，另一个为空Referer的请求，因此设置防盗链时必须同时满足在Referer中添加Bucket域名，且允许空Referer的条件。当您使用OSS的Bucket域名预览非MP4文件时，则仅需允许空Referer。

5. 单击**保存**。

使用阿里云SDK

以下仅列举常见SDK的设置防盗链的代码示例。关于其他SDK的设置防盗链的代码示例，请参见SDK简介。

Python

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.BucketReferer;
import java.util.ArrayList;
import java.util.List;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            List<String> refererList = new ArrayList<String>();
            // 添加Referer白名单。Referer参数支持通配符星号（*）和问号（?）。
            refererList.add("http://www.aliyun.com");
            refererList.add("http://www.*.com");
            refererList.add("http://www.?.aliyuncs.com");
            // 设置存储空间Referer列表。设为true表示Referer字段允许为空，设为false表示Referer字段不允许为空。
            BucketReferer br = new BucketReferer(true, refererList);
            ossClient.setBucketReferer(bucketName, br);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置防盗链的具体操作，请参见添加或修改防盗链配置。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketReferer。

更多参考

- 当您需要限定符合特定条件的用户能够访问您Bucket内的全部或部分资源、对资源授予相关操作权限等，建议您使用Bucket Policy。例如您可以通过配置Bucket Policy的来源IP，限制符合指定IP或IP地址段的用户才能访问某个Bucket。关于配置Bucket Policy的具体操作，请参见通过Bucket Policy授权用户访问指定资源。
- 关于使用防盗链过程中遇到的常见问题，请参见OSS防盗链（Referer）配置及错误排除。

7.8. 合规保留策略

对象存储OSS支持WORM特性，允许用户以“不可删除、不可篡改”方式保存和使用数据，符合美国证券交易委员会（SEC）和金融业监管局（FINRA）的合规要求。

背景信息

OSS提供强合规策略，用户可针对存储空间（Bucket）设置基于时间的合规保留策略。当策略锁定后，用户可以在Bucket中上传和读取文件（Object），但是在Object的保留时间到期之前，任何用户都无法删除Object和策略。Object的保留时间到期后，才可以删除Object。OSS支持的WORM特性，适用于金融、保险、医疗、证券等行业。您可以基于OSS搭建云上数据合规存储空间。

OSS是目前已通过Cohasset Associates审计认证的云服务，可满足严格的电子记录保留要求，例如SEC Rule 17a-4（f）、FINRA 4511、CFTC 1.31等合规要求。更多信息，请参见OSS Cohasset Assessment Report。

注意事项

- 合规保留策略正在公测中，如需试用此功能，请联系技术支持申请试用。
- OSS目前仅支持针对Bucket级别设置合规保留策略。
- 同一个Bucket中，版本控制和合规保留策略无法同时配置。若Bucket已开启版本控制功能，则无法再配置保留策略。版本控制功能详情请参见版本控制介绍。
- Bucket内的Object在合规保留策略生效期间，可通过设置生命周期规则进行存储类型转化，在保证合规性的前提下，降低存储成本。

规则说明

OSS允许添加一条基于时间的合规保留策略，保护周期为1天到70年。

假设您在2013年6月1日创建一个名为examplebucket的Bucket，并且在不同时间上传了file1.txt、file2.txt、file3.txt三个Object。随后，在2014年7月1日创建了保护周期为5年的合规保留策略。有关这三个Object的具体上传时间以及对应的Object到期时间如下：

Object名称	上传时间	Object到期时间
file1.txt	2013年6月1日	2018年5月31日
file2.txt	2014年7月1日	2019年6月30日
file3.txt	2018年9月30日	2023年9月29日

- 生效规则

  当基于时间的合规保留策略创建后，该策略默认处于“InProgress”状态，且该状态的有效期为 24 小时。在有效期24小时内，此策略对应的Bucket资源处于保护状态。

  - 启动合规保留策略24小时内：若该策略未提交锁定，则Bucket所有者以及授权用户可以删除该策略；若该保留策略已提交锁定，则不允许删除该策略，且无法缩短策略保护周期，仅可以延长保护周期。
  - 启动合规保留策略24小时后：若超过24小时该保留策略未提交锁定，则该策略自动失效。
  - Bucket内的数据处于被保护状态时，若您尝试删除或修改这些数据，OSS API将返回 `409 FileImmutable` 的错误信息。

- 删除规则

  - 基于时间的合规保留策略是Bucket的一种Metadata属性。当删除某个Bucket时，该Bucket对应的合规保留策略以及访问策略也会被删除。因此当Bucket为空时，Bucket的所有者可以删除该Bucket，从而间接删除该Bucket的保留策略。
  - 启动保留策略24小时内，若该保留策略未提交锁定，则Bucket所有者以及授权用户可以删除该策略。
  - 若Bucket内有文件处于保护周期内，那么您将无法删除保留策略，同时也无法删除Bucket。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**基础设置>保留策略**。在**保留策略**区域，单击**设置**。
4. 单击**创建策略**。
5. 在**创建策略**对话框，设置合规保留策略的**保留周期**。保留周期的取值范围为1天到70年。
6. 单击**确定**。

策略创建后，状态显示为IN_PROGRESS。此状态下，策略可被锁定和删除。

8. 单击**锁定**。确认合规保留策略无误后，单击**确定**。

> **注意**
>
> - 此时策略状态变为LOCKED，您仅可以单击编辑延长文件的保护周期，无法删除策略或缩短文件的保护周期。
> - 当Bucket内的数据处于被保护状态时，如果您在控制台上尝试删除或修改这些数据，控制台上会返回“该文件已被锁定，不可执行操作”的错误提示。

使用阿里云SDK

以下仅列举常见SDK的设置合规保留策略的代码示例。关于其他SDK的设置合规保留策略的代码示例，请参见SDK简介。

Python

```
// Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。关于其他Region对应的Endpoint信息，请参见访问域名和数据中心。
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
 String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
 // 填写Bucket名称，例如examplebucket。
String bucketName = "examplebucket";

// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// 创建InitiateBucketWormRequest对象。
InitiateBucketWormRequest initiateBucketWormRequest = new InitiateBucketWormRequest(bucketName);
// 指定Object保护天数为30天。
initiateBucketWormRequest.setRetentionPeriodInDays(30);

// 创建合规保留策略。
InitiateBucketWormResult initiateBucketWormResult = ossClient.initiateBucketWorm(initiateBucketWormRequest);

// 查看合规保留策略Id。
String wormId = initiateBucketWormResult.getWormId();
System.out.println(wormId);

// 关闭ossClient。
ossClient.shutdown();
```

使用命令行管理工具ossutil

关于使用ossutil设置合规保留策略的具体操作，请参见创建并锁定合规保留策略。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见InitiateBucketWorm。

7.9. OSS沙箱

当您的OSS存储空间（Bucket）遭受攻击或通过Bucket分享违法内容，OSS会自动将Bucket切入沙箱。沙箱中的Bucket仍可以正常响应请求，但服务质量将被降级，您的应用可能会有明显感知。

注意事项

- 对于被攻击的Bucket，OSS会将其切入沙箱。如果您的Bucket遭受攻击，您需要自行承担因攻击而产生的全额费用。
- 如果您的用户通过您的Bucket分享涉黄、涉政、涉恐等违法内容，也会导致您的Bucket被切入沙箱。情节严重者，将被追究法律责任。

针对攻击的预防措施

为防止您的Bucket因DDoS和CC攻击等原因被切入沙箱，您可以配置OSS高防或者配置ECS反向代理并绑定高防IP。这两种方案的优劣势如下表所示：

方案名称	说明	优势	劣势
方案一：配置OSS高防	OSS高防是OSS结合DDoS高防推出的DDoS攻击代理防护服务。为Bucket配置OSS高防后，在Bucket遭受大流量攻击时，OSS高防会将攻击流量牵引至高防集群进行清洗，并将正常访问流量回源到目标Bucket，确保业务的正常进行。	• 使用场景多样化：同时支持防护Bucket域名以及自定义域名。 • 成本较低：根据您的高防实例数量、高防防护流量和高防请求次数等收取DDoS防护费用。详情请参见DDoS防护费用。 • 配置简单：支持控制台图形化设置。	防护Bucket的数量有限：每个地域仅可以创建一个高防OSS实例，每个实例最多只能绑定同一地域下的10个Bucket。
方案2：配置ECS反向代理并配置高防IP	基于安全考虑，Bucket默认域名解析的IP地址是随机变化的。如果您期望使用固定IP地址访问，推荐使用ECS搭建反向代理的方式访问OSS。ECS上的EIP可以绑定高防IP以抵御DDoS攻击和CC攻击。	适合通过固定IP地址访问OSS的场景。	• 配置复杂：需自行搭建Nginx反向代理。 • 成本高：需额外购买ECS搭建反向代理。

两种方案实现步骤如下：

- 方案一：配置OSS高防

  通过OSS控制台为Bucket配置OSS高防的步骤如下：

  i. 创建高防OSS实例。

  ii. 绑定Bucket。

  iii. 如果需要防护自定义域名，请添加对应的自定义域名。

  有关配置OSS高防的注意事项，请参见配置OSS高防。

- 方案二：配置ECS反向代理并绑定高防IP

  请按如下步骤配置ECS反向代理并绑定高防IP：

  i. 配置ECS反向代理。

  a. 创建一个CentOS或Ubuntu的ECS实例。具体步骤，请参见创建ECS实例。

  > **注意** 如果Bucket有较大的网络流量或访问请求，请提高ECS硬件配置或者搭建ECS集群。

  b. 配置以ECS反向代理的方式访问OSS。具体步骤，请参见配置OSS反向代理。

  ii. 绑定高防IP。

  a. 结合业务情况购买合适的高防IP。购买页面，请参见高防IP。

  b. 绑定高防IP。其中，**网站**填写您ECS绑定的域名。服务器地址选择**源站IP**，并填写ECS的外网IP地址。其他参数的配置说明，请参见添加网站。

7.10. OSS高防

OSS高防是OSS结合DDoS高防推出的DDoS攻击代理防护服务。当受保护的存储空间（Bucket）遭受大流量攻击时，OSS高防会将攻击流量牵引至高防集群进行清洗，并将正常访问流量回源到目标Bucket，确保业务能正常进行。

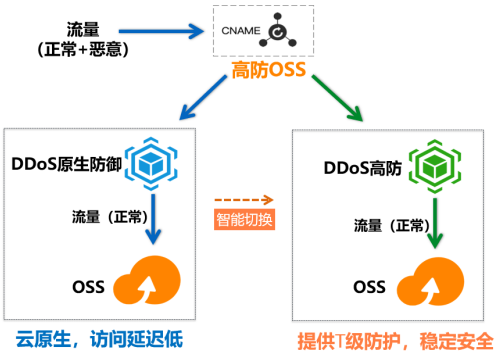
使用场景

DDoS攻击是近年来对企业业务危害最大的攻击手段之一。当企业遭受DDoS攻击时，可能会导致业务中断，进而导致企业的形象受损、客户流失、收益受损等，严重影响企业业务的正常运营。

为此，OSS深度结合DDoS高防产品，提供最高T级DDoS防护能力、百万QPS防护、秒级攻击切换能力，可有效抵御SYN Flood、ACK Flood、ICMP Flood、UDP Flood、NTP Flood 、SSDP Flood、DNS Flood、HTTP Flood等攻击。OSS高防适用于业务经常遭恶意攻击、勒索、刷单、刷流量等安全防护场景。

防护原理

OSS高防的防护原理如下图所示：



OSS默认使用DDoS原生防护保护您的Bucket。但是，当攻击频率超出DDoS原生防护的防御阈值时，原生防护将无法提供有效防护，可能会出现Bucket访问异常的情况。

您开通了OSS高防后，当攻击频率超出DDoS原生防护的防御阈值时，OSS会自动将受攻击Bucket的所有访问流量牵引至高防集群。恶意攻击流量将在高防流量清洗中心进行清洗过滤，正常访问流量通过端口协议转发的方式返回给目标Bucket，保证Bucket在受攻击时的稳定访问。

当攻击结束后，受攻击Bucket会切回至DDoS原生防护进行保护。

使用限制

- 仅支持在华东1（杭州）、华东2（上海）、华北1（青岛）、华北2（北京）、华南1（深圳）和中国（香港）地域配置OSS高防。
- 高防OSS实例创建后需至少使用7天。如果实例在7天内被删除，OSS会收取剩余时间的高防基础资源费用。计费详情，请参见
- 每个地域可以创建一个高防OSS实例，每个实例最多只能绑定同一地域下的10个Bucket。
- 在Bucket绑定高防实例后，您无法通过浏览器预览Bucket内的资源。此外，OSS默认不对Bucket关联的自定义域名进行防护，因此在Bucket遭到攻击时，无法通过自定义域名正常访问OSS。如果您希望在Bucket遭受攻击时可以通过自定义域名访问Bucket，请在OSS高防控制台添加要防护的自定义域名。每个Bucket防护的自定义域名数量上限是5个。

  如果Bucket中要防护的自定义域名（`www.example.com`）匹配了DDoS高防产品中配置了转发规则的相同域名（`www.example.com`）或泛域名（`*.example.com`），则需要前往DDoS高防控制台解绑相同域名或泛域名。否则，在该Bucket受到攻击时，无法通过该自定义域名正常访问OSS。

  关于同名或泛域名转发规则的更多信息，请参见添加网站。

使用OSS控制台

1. 创建高防OSS实例。
   i. 登录OSS管理控制台。
   ii. 在左侧导航栏，单击**OSS高防**。
   iii. 单击**创建高防OSS实例**。
   iv. 在**创建高防OSS实例**面板选择目标地域。
   v. 单击**确认**。
2. 绑定Bucket。
   i. 单击目标实例右侧的**查看和绑定Buckets**。
   ii. 在**查看和绑定Buckets**面板，单击**绑定高防Buckets**。
   iii. 在**绑定高防Buckets**面板的**高防Buckets**下拉列表，选择需要绑定的Bucket。

   已被高防OSS实例绑定的Bucket不会在**高防Buckets**下拉列表显示。
   iv. 单击**确定**。

   绑定后，Bucket处于**初始化**状态。待状态更新为**防护中**时，表明高防OSS实例已对Bucket域名开始实施保护。
3. 如果需要防护自定义域名，请添加对应的自定义域名。

   > **注意** OSS默认不对Bucket关联的自定义域名进行防护，因此在Bucket遭到攻击时，无法通过自定义域名正常访问OSS。如果您希望在Bucket遭受攻击时可以通过自定义域名访问Bucket，请添加要防护的自定义域名。每个Bucket最多可添加5个要防护的自定义域名。

   - 如果Bucket此前未绑定自定义域名，需要先绑定自定义域名。具体操作，请参见绑定自定义域名。
   - 如果Bucket已绑定自定义域名，请按如下步骤添加要防护的自定义域名：
     a. 选择已绑定高防Bucket右侧操作下**更多 > 修改自定义域名**。
     b. 选中要防护的自定义域名。
     c. 单击**确定**。

     高防OSS实例对选中的自定义域名开始实施防护。

8.日志管理

8.1. 日志转存

访问OSS的过程中会产生大量的访问日志。您可以通过日志转存功能将这些日志按照固定命名规则，以小时为单位生成日志文件写入您指定的存储空间（Bucket）。对于已存储的日志，您可以通过阿里云日志服务或搭建Spark集群等方式进行分析。

注意事项

- 生成日志的源Bucket和存储日志的目标Bucket可以相同也可以不同，但是必须属于同一账号下的相同地域。
- 日志文件以小时为单位生成，但并不表示某个时段的日志文件记录了该时段的所有请求，部分请求可能会出现在上一时段或下一时段的日志文件中。
- 在您关闭日志转存功能前，OSS的日志文件会一直生成。请及时清理不再需要的日志文件，以减少您的存储费用。

  您可以通过生命周期规则定期删除日志文件。更多信息，请参见基于最后一次修改时间的生命周期规则介绍。
- OSS会根据需求在日志的尾部添加一些字段，请您在开发日志处理工具时考虑兼容性的问题。

日志文件命名规则

转存后的日志文件命名规则如下：

```
<TargetPrefix><SourceBucket>YYYY-mm-DD-HH-MM-SS-UniqueString
```

字段	说明
TargetPrefix	日志文件的文件名前缀。
SourceBucket	产生访问日志的源Bucket名称。
YYYY-mm-DD-HH-MM-SS	日志文件被创建的时间。从左到右分别表示：年、月、日、小时、分钟和秒。
UniqueString	系统生成的字符串，是日志文件的唯一标识。

日志的格式和示例

- 日志格式

  OSS的访问日志包含请求者和被访问资源的相关信息，格式如下：

```
RemoteIP Reserved Reserved Time "RequestURL" HTTPStatus SentBytes RequestTime "Referer" "UserAgent" "HostName" "RequestID" "LoggingFlag" "Requester
AliyunID" "Operation" "BucketName" "ObjectName" ObjectSize ServerCostTime "ErrorCode RequestLength "UserID" DeltaDataSize "SyncRequest" "StorageCla
ss" "TargetStorageClass" "TransmissionAccelerationAccessPoint" "AccessKeyID"
```

字段	示例值	说明
RemoteIP	192.168.0.1	请求者的IP地址。
Reserved	-	保留字段，固定值为-。
Reserved	-	保留字段，固定值为-。
Time	03/Jan/2021:14:59:49 +0800	OSS收到请求的时间。
RequestURL	GET /example.jpg HTTP/1.0	包含query string的请求URL。 OSS会忽略以 `x-` 开头的query string参数，但这个参数会被记录在访问日志中。所以您可以使用 `x-` 开头query string参数标记一个请求，然后使用这个标记快速查找该请求对应的日志。
HTTPStatus	200	OSS返回的HTTP状态码。
SentBytes	999131	请求产生的下行流量。单位：Byte。
RequestTime	127	完成本次请求耗费的时间。单位：ms。
Referer	http://www.aliyun.com/product/oss	请求的HTTP Referer。
UserAgent	curl/7.15.5	HTTP的User-Agent头。
HostName	examplebucket.oss-cn-hangzhou.aliyuncs.com	请求访问的目标域名。
RequestID	5FF16B65F05BC932307A3C3C	请求的Request ID。
LoggingFlag	true	是否已开启日志转存。取值如下： ○ *true*表示已开启日志转存。 ○ *false*表示未开启日志转存。
RequesterAliyunID	16571836914537****	请求者的用户ID。取值-表示匿名访问。
Operation	GetObject	请求类型。
BucketName	examplebucket	请求的目标Bucket名称。
ObjectName	example.jpg	请求的目标Object名称。

字段	示例值	说明
ObjectSize	999131	目标Object大小。单位：Byte。
ServerCostTime	88	OSS处理本次请求所花的时间。单位：毫秒。
ErrorCode	-	OSS返回的错误码。取值-表示未返回错误码。
RequestLength	302	请求的长度。单位：Byte。
UserID	16571836914537****	Bucket拥有者ID。
DeltaDataSize	-	Object大小的变化量。取值-表示此次请求不涉及Object的写入操作。
SyncRequest	cdn	请求是否为CDN回源请求。取值如下： ○ *cdn*表示请求是CDN回源请求。 ○ -表示请求不是CDN回源请求。
StorageClass	Standard	目标Object的存储类型。取值如下： ○ *Standard*表示标准存储。 ○ *IA*表示低频访问存储。 ○ *Archive*表示归档存储。 ○ *Cold Archive*表示冷归档存储。 ○ -表示未获取Object存储类型。
TargetStorageClass	-	是否通过生命周期规则或CopyObject转换了Object的存储类型。取值如下： ○ *Standard*表示转换为标准存储。 ○ *IA*表示转换为低频访问存储。 ○ *Archive*表示转换为归档存储。 ○ *Cold Archive*表示转换为冷归档存储。 ○ -表示请求不涉及Object存储类型转换操作。
TransmissionAccelerationAccessPoint	-	通过传输加速域名访问目标Bucket时使用的传输加速接入点。例如请求者通过华东1（杭州）的接入点访问目标Bucket时，值为*cn-hangzhou*。 取值-表示未使用传输加速域名或传输加速接入点与目标Bucket所在地域相同。
AccessKeyID	LTAI4FrfJPUSoKm4JHb5****	请求者的AccessKey ID。取值-表示匿名请求。

- 日志示例

```
192.168.0.1 - - [03/Jan/2021:14:59:49 +0800] "GET /example.jpg HTTP/1.0" 200 999131 127 "http://www.aliyun.com/product/oss" "curl/7.15.5" "examplebucket.oss-cn-hangzhou.aliyuncs.com" "5FF16B65F05BC932307A3C3C" "true" "16571836914537****" "GetObject" "examplebucket" "example.jpg" 999131 88 "-" 302 "16571836914537****" - "cdn" "standard" "-" "-" "LTAI4FrfJPUSoKm4JHb5****"
```

日志文件转存到OSS指定Bucket后，您可以通过日志服务对日志文件进行分析。对日志文件进行分析前，您需要通过数据导入方式将OSS日志文件导入到日志服务。有关数据导入的具体操作，请参见导入OSS数据。有关日志服务分析功能的更多信息，请参见分析概述。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**日志管理 > 日志转存**。
4. 单击**设置**，设置日志存储位置及日志前缀。
   - **日志存储位置**：下拉选择存储日志记录的Bucket名称，只能选择同一账号下相同地域的Bucket。
   - **日志前缀**：日志文件存储的目录。如果指定此项，则日志文件将保存在目标Bucket的指定目录下。如果不指定此项，则日志文件将保存在目标Bucket的根目录下。例如，日志前缀指定为*log/*，则日志文件将被记录在*log/*目录下。
5. 单击**保存**。

使用阿里云SDK

以下仅列举常见SDK的设置日志转存的代码示例。关于其他SDK的设置日志转存的代码示例，请参见SDK简介。

Python

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.SetBucketLoggingRequest;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写待开启日志转存功能的Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写存放日志文件的目标Bucket。targetBucketName与bucketName可以为相同或不同的Bucket。
        String targetBucketName = "yourTargetBucketName";
        // 设置日志文件存储的目录为log/。如果指定此项，则日志文件将保存在目标Bucket的指定目录下。如果不指定此项，则日志文件将保存在目标Bucket的根目录下。
        String targetPrefix = "log/";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            SetBucketLoggingRequest request = new SetBucketLoggingRequest(bucketName);
            request.setTargetBucket(targetBucketName);
            request.setTargetPrefix(targetPrefix);
            ossClient.setBucketLogging(request);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用命令行工具ossutil

关于使用ossutil设置日志转存的具体操作， 请参见开启日志转存。

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketLogging。

常见问题

中断的请求能否在OSS访问日志中查询？

OSS访问日志目前不会记录中断的请求。如果您是通过SDK发起的请求，可以根据SDK的返回值判断请求中断的原因。

8.2. 实时日志查询

访问对象存储OSS的过程中会产生大量的访问日志。实时日志查询功能将OSS与日志服务SLS相结合， 允许您在OSS控制台直接查询OSS的访问日志，帮助您完成OSS访问的操作审计、访问统计、异常事件回溯和问题定位等工作，提升您的工作效率并更好地帮助您基于数据进行决策。

前提条件

- 已开通日志服务。

  如果您还未开通日志服务，请前往日志服务产品页完成开通操作。
- 已授权日志服务访问OSS。

  如果您还未授权日志服务访问OSS，请单击云资源访问授权，按照提示完成授权操作。

功能优势

- 3分钟内将日志实时推送到日志服务实例中，支持在OSS控制台直接查看实时日志。
- 提供日志分析服务，定制了常用的分析报表，数据查询更方便。
- 支持实时查询和分析原始日志，并按照Bucket名称、Object名称、API操作、时间等条件过滤日志。

费用说明

使用实时日志功能时，如果出现以下情况，则会产生费用：

- 实时日志查询免费提供最近7天的日志查询。若您设置的日志存储时间大于7天，则超过7天的部分，由日志服务单独收费。
- 实时日志查询免费提供900 GB/天的日志写入额度（如果一条访问日志为1 KB，约为9亿条），超过部分由日志服务单独收费。
- 当您通过外网读写日志服务时，会产生外网读写流量费用。

具体收费标准，请参见日志服务计费方式。

开通实时日志查询

您可以通过以下两种方式开通实时日志查询功能：

- i. 登录OSS管理控制台。
  ii. 在**概览**页面，单击右侧的**创建Bucket**。
  iii. 在**创建Bucket**对话框，**实时日志查询**区域选择**开通**，其他参数的配置详情，请参见创建存储空间。
  iv. 单击**确定**。

方式一：新建Bucket时开通实时日志查询

- i. 登录OSS管理控制台。
  ii. 单击**Bucket列表**，之后单击目标Bucket名称。
  iii. 在左侧导航栏，选择**日志管理 > 实时查询**。
  iv. 单击**立即开通**。

方式二：为已创建的Bucket开通实时日志查询

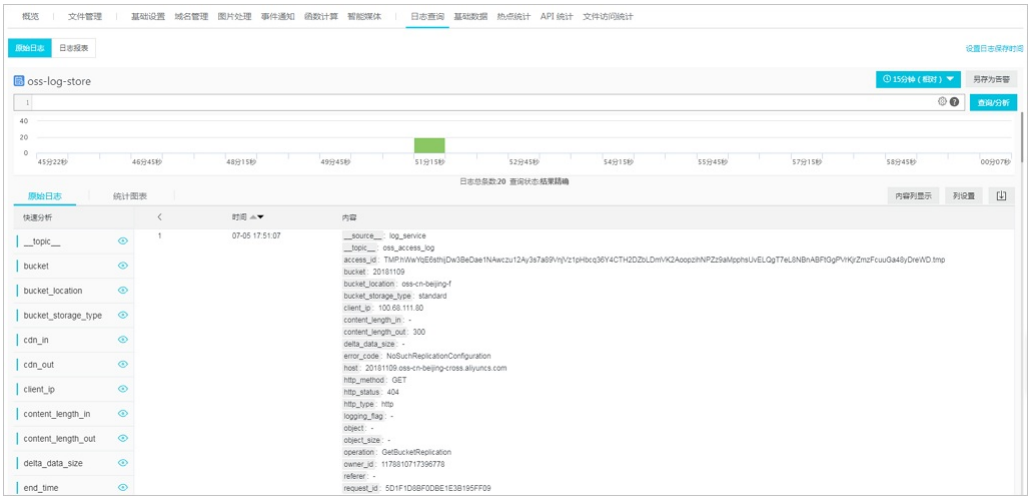
开通实时日志查询，OSS免费提供最近7天内的日志查询。您可以单击日志页面右上角的**设置日志保存时间**，修改日志的保存时间。

查询实时日志

您可以通过以下三种方式查询实时日志：

- i. 登录OSS管理控制台。
  ii. 单击**Bucket列表**，之后单击目标Bucket名称。
  iii. 在左侧导航栏，选择**日志管理 > 实时查询**。
  iv. 单击**原始日志**，对日志进行分析。

  您可以指定时间段和查询语句进行实时查询。例如可快速分析某一个字段（如API操作名称）在一段时间内的分布情况，您也可以按条件过滤或统计您希望查看的访问记录。



方式一：通过原始日志页面查询实时日志

- i. 登录OSS管理控制台。
  ii. 单击**Bucket列表**，之后单击目标Bucket名称。
  iii. 在左侧导航栏，选择**日志管理 > 实时查询**。
  iv. 单击**日志报表**，对日志进行分析。

  日志报表提供以下四类开箱即用的报表：

  - **访问中心**：展示总体运营状况信息，包括PV、UV、流量以及外网访问地图分布等。
  - **审计中心**：展示文件操作的统计信息，包括文件读、写、删等操作统计。
  - **运维中心**：展示针对访问日志的统计信息，包括请求数量、失败操作的分布统计等信息。
  - **性能中心**：展示针对性能的统计信息，包括外网下载和上传性能分布、不同网络与文件大小的传输性能、文件下载差异列表等信息。

方式二：通过原始日志页面查询实时日志

- 您可以在日志服务控制台查询OSS的实时日志。具体操作，请参见OSS访问日志。

方式三：通过日志服务控制台查询实时日志

关闭实时日志查询

当您确认不再需要保留日志数据，请按如下步骤关闭实时日志查询。

1. 登录OSS管理控制台。
2.
3. 单击**日志管理 > 实时查询**。

4. 单击右上角**关闭关联日志**。

> **注意** 开通实时日志查询时，会自动创建日志服务Project。但关闭实时日志查询时不会自动删除Project。因此，当您关闭实时日志查询后，为避免后续产生不必要的费用，请前往日志服务控制台删除开通实时日志查询时自动创建的Project。具体操作，请参见删除Project。

9.静态网站托管

9.1. 静态网站托管介绍

静态网站是指所有的网页都由静态内容构成，包括客户端执行的脚本（例如JavaScript）。您可以通过静态网站托管功能将您的静态网站托管到OSS的存储空间（Bucket），并使用Bucket的访问域名访问这个网站。

使用说明

配置静态网站托管时，您需要指定网站的默认首页和默认404页：

- 默认首页是您通过浏览器访问静态网站域名时，OSS返回的网站首页。

  您为默认首页指定的文件必须是Bucket根目录下允许被匿名访问的文件。如果您还开通了子目录首页，则子目录下也应存在此文件。
- 默认404页是您通过浏览器访问Bucket内文件出现404错误时，OSS返回的错误页面。

  您为默认404页指定的文件必须是Bucket根目录下允许被匿名访问的文件。

您可以通过将默认首页或者默认404页中指定文件的读写权限ACL设置为 `public-read` ，确保该文件允许匿名访问。有关设置文件读写权限ACL的具体步骤，请参见文件ACL。

静态网站配置完成后，如果您使用Bucket默认域名访问静态网站时，会将静态网站以文件的形式下载到本地。如需确保访问静态网站是预览行为，您必须为Bucket绑定自定义域名，并通过自定义域名访问您的静态网站。绑定自定义域名步骤，请参见绑定自定义域名。

注意事项

出于安全考虑，中国区域自2018年08月13日起，中国以外区域自2019年09月25日起，通过浏览器访问OSS上网页类型文件（mimetype为text/html，扩展名包括HTM、HTML、JSP、PLG、HTX、STM）：

- 使用OSS默认域名访问时，Response Header中会自动加上 `Content-Disposition:'attachment=filename;'` 。即从浏览器访问网页类型文件时，不会显示文件内容，而是以附件形式进行下载。
- 使用绑定的自定义域名访问OSS时，Response Header中不会加上 `Content-Disposition:'attachment=filename;'` ，只要您的浏览器支持该类型文件的预览，可以直接预览文件内容。绑定自定义域名详细步骤请参见绑定自定义域名。

配置示例

为Bucket开启静态网站托管后，您需要将与默认首页名称相同的文件（例如*index.html*）上传至目标Bucket，如果Bucket中包含了目录结构*subdir/*，则目录层级下也必须包含*index.html*文件。此外，您还需要将与默认404页名称相同的文件（例如*error.html*）上传至目标Bucket。Bucket的文件结构如下所示：

```
Bucket
 ├── index.html
 ├── error.html
 ├── example.txt
 └── subdir/
      └── index.html
```

如果该Bucket绑定了自定义域名 `example.com` ，且配置的静态网站默认首页为*index.html*，默认404页为*error.html*。则通过自定义域名访问静态网站时，根据是否开通了子目录首页，访问规则如下：

- 未开通子目录首页
  - 当您访问*https://example.com/*和*https://example.com/subdir/*时，OSS会返回*https://example.com/index.html*。
  - 当您访问*https://example.com/example.txt*时，正常获取*example.txt*文件。
  - 当您访问*https://example.com/object*时，因*object*不存在，OSS会返回*https://example.com/error.html*。
- 已开通子目录首页
  - 当您访问*https://example.com/*时，OSS会返回*https://example.com/index.html*。
  - 当您访问*https://example.com/subdir/*时，OSS会返回*https://example.com/subdir/index.html*。
  - 当您访问*https://example.com/example.txt*时，正常获取*example.txt*文件。
  - 当您访问*https://example.com/object*时，因*object*不存在，OSS会根据您设置的文件404规则返回对应信息：
    - 如果文件404规则设置为Redirect（默认值），OSS会继续检查*object/index.html*是否存在。如果文件存在则返回302，并将访问请求重定向为*https://example.com/object/index.html*；如果文件不存在则返回404，并继续检查*https://example.com/error.html*。
    - 如果文件404规则设置为NoSuchKey，则直接返回404，并继续检查*https://example.com/error.html*。
    - 如果文件404规则设置为Index，OSS会继续检查*object/index.html*是否存在。如果文件存在则返回200，并直接返回文件内容。如果文件不存在，则继续检查*https://example.com/error.html*。

使用OSS控制台

1. 设置静态网站页面。
   - 未开通子目录首页

     结合以上配置示例可知，当您希望访问子目录subdir/时，不支持跳转至子目录下的index.html页面，而是跳转至根目录下的index.html页面。此外，当访问Bucket内不存在的文件时，返回默认错误页面。具体配置步骤如下：

     a. 登录OSS管理控制台。

     b. 单击**Bucket列表**，然后单击目标Bucket名称。

     c. 在左侧导航栏，选择**基础设置 > 静态页面**。

d. 在**静态页面**区域，单击**设置**，按如下说明配置各项参数。

参数	说明
默认首页	默认首页是您通过浏览器访问静态网站域名时，OSS返回的网站首页。此处设置为*index.html*。
子目录首页	选择**不开通**，此时访问静态网站根域名或者根域名下任何一个以正斜线（/）结尾的URL都会返回根目录默认首页。
默认404页	访问Bucket内文件出现404错误时，OSS返回的错误页面。默认404页仅支持根目录下的文件。此处设置为*error.html*。
错误文档响应码	您可以配置返回错误文档时的HTTP响应码为**404**或**200**。

e. 单击**保存**。

- 已开通子目录首页

  结合以上配置示例可知，您希望访问子目录subdir/时，支持直接跳转至子目录下的index.html页面。此外，当访问Bucket内不存在的文件时，返回默认错误页面，并通过文件404规则指定访问不存在文件时的返回结果。具体配置步骤如下：

  a. 单击**Bucket列表**，然后单击目标Bucket名称。

  b. 在左侧导航栏，选择**基础设置 > 静态页面**。

c. 在**静态页面**区域，单击**设置**，按如下说明配置各项参数。

参数	说明
默认首页	默认首页是您通过浏览器访问静态网站域名时，OSS返回的网站首页。此处设置为*index.html*。
子目录首页	选择**开通**。开通子目录首页后，访问静态网站根域名时，返回根目录默认首页。访问根域名下以正斜线（/）结尾的URL时会返回对应目录的默认首页。例如，访问示例中的 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/subdir/` 时，则返回*subdir/*目录下的默认首页文件 `index.html` 。
文件404规则	开通子目录首页后，通过文件404规则决定访问不存在的Object时的返回结果。例如，访问 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/exampledir` ，因示例中不存在*exampledir*文件，则根据设置的文件404规则返回对应信息： ■ **Redirect**（默认值）：检查*exampledir/index.html*是否存在。 　■ 如果文件存在则返回302，并将访问请求重定向为 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/exampledir/index.html` 。 　■ 如果文件不存在则返回404，并继续检查 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/error.html` 。如果error.html页面也不存在该文件，则返回404状态码。 ■ **NoSuckKey**：直接返回404，并继续检查 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/error.html` 。 ■ **Index**：检查*exampledir/index.html*是否存在。 　■ 如果文件存在则返回200，并直接返回文件内容。 　■ 如果文件不存在，则继续检查 `https://examplebucket.oss-cn-hangzhou.aliyuncs.com/error.html` 。
默认404页	访问Bucket内文件出现404错误时，OSS返回的错误页面。默认404页仅支持根目录下的文件。此处设置为*error.html*。
错误文档响应码	您可以配置返回错误文档时的HTTP响应码为**404**或**200**。

d. 单击**保存**。

2. 创建并上传默认首页。

   当您为examplebucket配置静态网站托管时指定的默认首页为*index.html*，您需要将与默认首页名称相同的文件上传至examplebucket根目录下。由于examplebucket中包含了子目录*subdir/*，则子目录*subdir/*下也必须包含*index.html*文件。

   i. 创建*index.html*文件。*index.html*文件配置示例如下：

```
<html>
<head>
    <title>My Website Home Page</title>
    <meta charset="utf-8">
</head>
<body>
  <p>Now hosted on OSS.</p>
</body>
</html>
```

   ii. 将*index.html*文件保存至本地路径。

   iii. 分别将*index.html*文件上传至examplebucket根目录以及子目录subdir下。上传文件时，您需要将文件读写权限设置为公共读。

   关于上传文件的具体操作，请参见简单上传。

3. 创建并上传默认404页。

   当您为examplebucket配置静态网站托管时指定的默认404页为*error.html*，您需要将与默认404页名称相同的文件上传至examplebucket根目录下。

i. 创建*error.html*文件。*error.html*文件配置示例如下：

```
<html>
<head>
    <title>Hello OSS!</title>
    <meta charset="utf-8">
</head>
<body>
  <p>This is error 404 page.</p>
</body>
</html>
```

ii. 将*error.html*文件保存至本地。

iii. 将*error.html*文件上传至examplebucket根目录下。上传文件时，您需要将文件读写权限设置为公共读。

使用阿里云SDK

以下仅列举常见SDK的设置静态网站托管的代码示例。关于其他SDK的设置静态网站托管的代码示例，请参见SDK简介。

Python

```
import com.aliyun.oss.ClientException;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.OSSException;
import com.aliyun.oss.model.SetBucketWebsiteRequest;
public class Demo {
    public static void main(String[] args) throws Exception {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 填写Bucket名称。
            SetBucketWebsiteRequest request = new SetBucketWebsiteRequest(bucketName);
            // 设置静态网站托管的默认主页。
            request.setIndexDocument("index.html");
            // 设置静态网站托管的默认404页。
            request.setErrorDocument("error.html");
            ossClient.setBucketWebsite(request);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用REST API

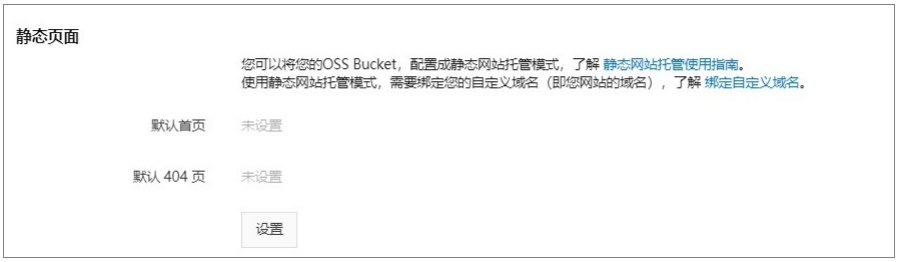
如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。更多信息，请参见PutBucketWebsite。

常见问题

开启静态网站托管功能后是否支持关闭？

当您不再需要使用默认首页、默认404页等静态网站托管配置时，请按如下步骤关闭静态网站托管功能：

1. 在左侧导航栏，选择**基础设置 > 静态页面**。
2. 在**静态页面**区域，单击**设置**。
3. 清空默认首页和默认404页配置，然后单击**保存**。
   返回如下页面，表示已成功关闭静态网站托管功能。



更多参考

- 您可以通过存储空间（Bucket）托管静态网站，并让访问者通过Bucket绑定的自定义域名（例如example.com）访问您的网站。具体操作，请参见使用自定义域名设置静态网站托管。
- 您可以使用React框架，通过OSS的静态网站托管功能在前端快速部署一个线上可用的单页应用SPA（Single-Page Application）。具体操作，请参见教程示例：通过静态网站托管部署单页应用。

9.2. 教程示例：使用自定义域名设置静态网站托管

您可以通过存储空间（Bucket）托管静态网站，并让访问者通过Bucket绑定的自定义域名（例如example.com）访问您的网站。无论您是想在OSS上托管已有静态网站还是从零开始建站，都可以从此教程中获得帮助。

步骤1：注册域名

搭建静态网站前，您需要为网站准备一个域名。建议您使用阿里云域名服务快速注册一个属于您的域名。详细步骤，请参见注册通用域名。

本示例使用 `example.com` 作为测试域名。

> **注意** 若您注册的域名需绑定在中国内地的Bucket上，您还需在中国工信部备案域名。详细步骤，请参见备案。

步骤2：创建Bucket

您需要创建一个公共读的Bucket，用以设置静态网站托管及存放网站数据。

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击**创建Bucket**。
3. 在**创建Bucket**面板配置Bucket参数，其中：

参数	说明
Bucket名称	设置Bucket名称。本示例设置为**examplebucket**。
地域	选择Bucket所在地域。本示例选择**华东1（杭州）**。
存储类型	选择**标准存储**。
读写权限	选择**公共读**。

其他参数保持默认配置。更多信息，请参见创建存储空间。

步骤3：创建网页文件并上传

您需要创建静态网站首页和404错误页面的网页文件，并上传至目标Bucket。

1. 在本地创建两个HTML格式的文件。
   - 默认首页

     本示例使用以下内容生成静态网站的首页文件*index.html*。实际环境中，请根据您的需求生成首页文件内容。

```
<html>
   <head>
       <title>Hello OSS!</title>
       <meta charset="utf-8">
   </head>
   <body>
       <p>开始阿里云OSS托管</p>
       <p>这是索引页面</p>
   </body>
 </html>
```

   - 默认404页

     本示例使用以下内容生成静态网站的404错误页文件*error.html*。实际环境中，请根据您的需求生成404错误页文件的内容。

```
<html>
<head>
   <title>Hello OSS!</title>
   <meta charset="utf-8">
</head>
<body>
   <p>这是404错误页面</p>
</body>
</html>
```

2. 将网页文件上传至目标Bucket。
   i. 登录OSS管理控制台。
   ii. 单击**Bucket列表**，然后单击目标Bucket。
   iii. 单击**文件管理**，然后单击**上传文件**。
   iv. 在**上传文件**面板的**上传文件**区域，单击**直接上传**并选中刚刚创建的两个网页文件。其他参数均保持默认配置。

步骤4：配置静态网站托管

1. 单击**基础设置 > 静态页面**。
2. 单击**设置**，将*index.html*设置为**默认首页**；将*error.html*设置为**默认404页**。

   说明 若您希望访问子目录时可以跳转到子目录下*index.html*文件，您可以选择开通**子目录首页**。详情请参见设置静态网站托管。

3. 单击**保存**。

步骤5：绑定自定义域名

现在，您已有了根域名 `example.com` 和名为examplebucket的Bucket，接下来您需要将域名绑定到Bucket，以便能够使用您的域名访问Bucket。

1. 在Bucket管理页面，单击**传输管理 > 域名管理**。
2. 单击**绑定域名**。
3. 在**域名**文本框输入**example.com**，并打开**自动添加CNAME记录**开关。
4. 单击**确定**。

步骤5：（可选）使用阿里云CDN加快网站速度

您可以使用阿里云CDN改善网站性能。CDN可以让您的网站文件（如HTML、图像和视频）缓存至全球各地的数据中心。当访问者从您的网站请求文件时，CDN会自动将请求重定向到最近数据中心上的文件副本，提升访问者的下载速度。

1. 在Bucket管理页面，单击**传输管理 > 域名管理**。
2. 单击目标域名右侧的**未配置**，跳转至CDN控制台。
3. 在**添加域名**页面配置以下参数：

参数	说明
加速域名	保持默认配置。
资源分组	选择默认资源组。
业务类型	不同的业务类型有不同的流量分配，按照您存储的内容及使用情况选择合适的业务类型。本示例选择**图片小文件**。
源站信息	保持默认配置
端口	选择CDN服务端口。本示例选择**80端口**。
加速区域	选择CDN的加速区域。本示例选择**仅中国内地**。

4. 单击**下一步**，然后单击**返回域名列表**。
5. 记录域名生成的CNAME值**example.com.w.kunlunsl.com**，然后修改绑定自定义域名时增加的CNAME记录。
   i. 进入云解析DNS控制台。
   ii. 在域名解析列表中，单击目标域名右侧的**解析设置**。
   iii. 找到绑定自定义域名时自动添加的CNAME记录，单击**修改**。
   iv. 将**记录值**修改为**example.com.w.kunlunsl.com**，其他参数保持原来配置。
   v. 单击**确认**。
6. （可选）回到**域名管理**页签，打开目标域名右侧的**CDN缓存自动刷新**开关。

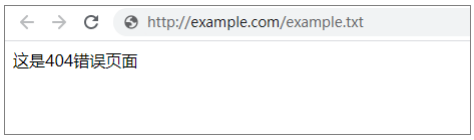
   如果您希望针对指定操作触发CDN缓存自动刷新，可以提交工单申请开通针对指定操作触发CDN缓存自动刷新的功能。开通后，您需要单击目标域名右侧**支持的操作**，然后选中指定操作类型。支持的操作类型，请参见开启CDN缓存自动刷新。

步骤6：测试网站

在浏览器中访问以下URL以验证网站是否正常运行：

- 访问静态网站首页 `http://example.com` ，配置正确时显示如下页面：

- 访问Bucket中不存在的文件 `http://example.com/example.txt` ，配置正确时显示如下页面：



步骤7：清理资源

本教程中创建的资源仅为测试环境使用，建议您在测试完成后清理资源，以免产生不必要的费用。

- CDN服务绑定的域名。操作步骤，请参见停止加速服务。
- 云解析服务添加的CNAME记录。操作步骤，请参见删除记录
- OSS中创建的Bucket及上传的文件。操作步骤，请参见删除文件和删除存储空间。

9.3. 教程示例：通过静态网站托管部署单页应用

本文介绍如何使用React框架，通过OSS的静态网站托管功能在前端快速部署一个线上可用的单页应用SPA（Single-Page Application）。

什么是单页应用

单页应用是只有一个Web页面的应用，是一种网络应用程序或网站的模型。通过动态重写当前页面与用户进行交互，而非从服务器重新加载整个新页面。单页应用避免了因页面之间的切换打断用户体验，使应用程序更像一个桌面应用程序。在单页应用中，所有必要的代码（HTML、JavaScript和CSS）都通过单个页面的加载而检索，或者通常是为了响应用户操作等需要，动态装载适当的资源并添加到页面。

前提条件

- 已安装Node.js SDK。具体操作，请参见安装。
- 已创建Bucket。本教程以创建名为examplebucket的Bucket为例，具体操作，请参见创建Bucket。
- 已为名为examplebucket绑定自定义域名 `example.com` 。具体操作，请参见绑定自定义域名。

步骤一：使用React快速创建单页应用

1. 打开命令行终端 `cmd` 或者PowerShell，本教程以 `cmd` 为例。
2. 执行以下命令创建项目。

   ```
   npx create-react-app spa-demo
   ```

   返回示例以下：

   ```
   Need to install the following packages:
   create-react-app
   Ok to proceed? (y)
   ```

3. 在 `Ok to proceed? (y)` 后输入*y*，然后回车。

   等待几分钟后项目将自动创建完成，同时会自动安装依赖。
4. 执行以下命令进入已创建的项目。

   ```
   cd spa-demo
   ```

5. 执行以下命令预览已创建的项目。

   ```
   npm run start
   ```

   App.js文件如下图所示：

   ```
   import logo from './logo.svg';
   import './App.css';

   function App() {
     return (
       <div className="App">
         <header className="App-header">
           <img src={logo} className="App-logo" alt="logo" />
           <p>
             Edit <code>src/App.js</code> and save to reload.
           </p>
           <a
             className="App-link"
             href="https://reactjs.org"
             target="_blank"
             rel="noopener noreferrer"
           >
             Learn React
           </a>
         </header>
       </div>
     );
   }  >>> (You 4 days ago) Initialize project using Create React App

   export default App;
   ```

6. 调试并预览检查无误后，执行以下命令打包生产环境代码。

   ```
   npm run build
   ```

   项目根目录下生成*build*目录。

步骤二：为examplebucket配置静态网站托管

1.
2. 单击**Bucket列表**，然后单击**examplebucket**。
3. 选择**基础设置** > **静态页面**，在**静态页面**区域单击**设置**，按以下说明配置各项参数：

参数	说明
默认首页	默认首页是您通过浏览器访问静态网站域名时，OSS返回的网站首页。此处设置为*index.html*。
子目录首页	选择**不开通**，此时访问静态网站根域名或者根域名下任何一个以正斜线（/）结尾的URL都会返回根目录默认首页。
默认404页	访问Bucket内文件出现404错误时，OSS返回的错误页面。当前配置用于前端单页应用，请将默认404页也配置为应用入口，即与默认首页相同的*index.html*。
错误文档响应码	配置返回错误文档时的HTTP响应码为200。

4. 单击**保存**。

步骤三：上传build目录下的所有文件

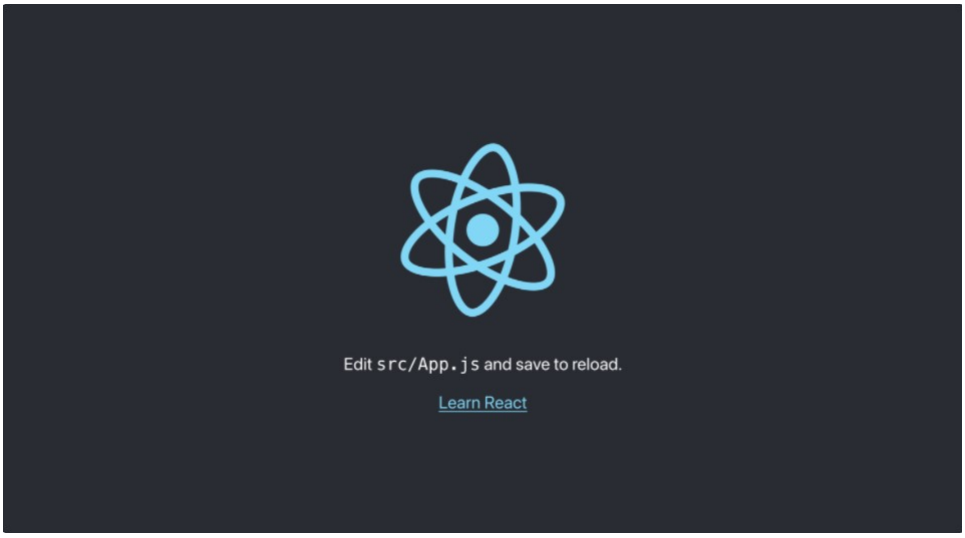
1. 单击左侧导航栏的**Bucket列表**，然后单击**examplebucket**。
2. 在左侧导航栏，选择**文件管理 > 文件管理**。
3. 在**文件管理**页签，单击**上传文件**。
4. 在**上传文件**面板，按以下说明配置各项参数。

参数	说明
上传到	选中**当前目录**。
文件ACL	设置文件读写权限ACL为**公共读**。
待上传文件	单击**扫描文件夹**，选中React生成的*spa-demo*项目中*build*目录下的所有文件。

5. 单击**上传**。
   此时，您可以在**上传列表**页签查看各个文件的上传进度。上传完成后，您可以在examplebucket的文件列表下找到名为**index.html**文件。

步骤四：通过自定义域名访问已部署的单页应用

1. 打开浏览器。
2. 本教程以自定义域名*example.com*为例，输入*https://example.com/index.html*，访问单页应用。
   效果如下图所示：



常见问题

- 部署好应用之后，切换路由能成功渲染，但页面刷新会出现404报错，怎么解决？

  原因可能是配置静态网站托管时，默认首页和默认404页配置有误。请确保**默认首页**和**默认404页**均配置为*index.html*。
- 切换路由之后，页面能正常显示。但HTTP状态码依然为404，怎样才能正常返回200？

  原因可能是配置静态网站托管时，错误文档响应码未配置或配置错误。请确保**错误文档响应码**配置为**200**。

10.数据处理与索引

10.1. 数据处理与索引介绍

您可以通过简单的REST API口在任何时间、任何地点、任何互联网设备上对存储在OSS中的数据进行分析处理。

数据处理与索引包含以下内容：

- 图片处理

  图片处理包括图片缩放、自定义裁剪、图片样式等。更多信息，请参见图片处理简介。

- 视频截帧

  OSS支持对视频编码格式为H264和H265的视频文件进行视频截帧。更多信息，请参见视频截帧。

- 数据索引

  数据索引是OSS对外提供的Object元数据索引能力。您可以利用Object的元数据自定义索引的条件以快速获取Object列表，帮助您更好地管理与了解数据结构，方便您后续查询、统计和管理Object。更多信息，请参见数据索引（Data Indexing）。

10.2. 新版图片处理指南

10.2.1. 简介

针对OSS内存储的图片文件（Object），您可以在Get Object请求中携带图片处理参数对图片文件进行处理。例如添加图片水印、转换格式等。

处理参数

OSS支持直接使用一个或多个参数处理图片，也支持将多个参数封装在一个样式中批量处理图片。有关图片样式的详情，请参见图片样式。

当存在多个图片处理参数时，OSS将按照参数顺序对图片进行处理。处理参数说明如下：

图片处理	参数	说明
图片缩放	resize	将图片缩放至指定大小。
图片水印	watermark	为图片添加图片或文字水印。
自定义裁剪	crop	裁剪指定大小的矩形图片。
质量变换	quality	调整JPG和WebP格式图片的质量。
格式转换	format	转换图片格式。
获取信息	info	获取图片信息，包括基本信息、EXIF信息。
自适应方向	auto-orient	将携带旋转参数的图片进行自适应旋转。
内切圆	circle	以图片中心点为圆心，裁剪出指定大小的圆形图片。
索引切割	indexcrop	按指定x或y轴的大小切分图片，之后选取其中一张图片。
圆角矩形	rounded-corners	按指定圆角大小将图片裁剪成圆角矩形。
模糊效果	blur	对图片进行模糊处理。
旋转	rotate	按指定角度以顺时针方向旋转图片。
渐进显示	interlace	将JPG格式的图片调整为渐进显示。
获取图片主色调	average-hue	获取图片主色调。
亮度	bright	调整图片亮度。
锐化	sharpen	对图片进行锐化处理。
对比度	contrast	调整图片对比度。

例如，对原图 `example.jpg` 添加图片缩放 `resize` 以及质量变换 `quality` 参数后，文件URL为 `https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_300/quality,q_90` 。您可以通过配置不同的规则，实现CDN回源原图或者经图片处理参数后的图片。

- 回源原图

  通过CDN开启过滤参数后，文件URL请求中问号（?）之后的参数将全部去除，即直接命中原图 `example.jpg` 。

- 回源处理后的图片

  通过CDN开启保留回源参数后，文件URL请求中问号（?）之后的所有参数将全部保留，即直接命中经图片处理参数后的图片。

有关CDN回源规则的配置详情，请参见过滤参数。

操作方式

您可以通过文件URL、API、SDK对图片进行处理。操作方式，请参见图片处理操作方式。

使用限制

使用图片处理服务时有如下限制：

- 原图限制
  - 图片格式只支持JPG、PNG、BMP、GIF、WebP、TIFF。

- 原图大小不能超过20 MB。
- 除图片旋转对应的原图高或者宽不能超过4,096 px外，其他图片操作对应的原图高或者宽不能超过30,000 px，且总像素不能超过2.5亿 px。

  动态图片（例如GIF图片）的像素计算方式为 `宽*高*图片帧数` ；非动态图片（例如PNG图片）的像素计算方式为 `宽*高` 。

- 动态图片限制

  仅支持对动态图片（例如GIF格式图片）进行缩放、裁剪、旋转以及添加图片水印的操作。

- 缩放后图片限制

  宽或高不能超过16,384 px，且总像素不能超过16,777,216 px。

- 样式限制

  每个存储空间下最多能创建50个样式。如您的业务有更多样式的需求，请联系技术支持。

费用说明

使用图片处理服务时，会产生如下费用：

- 图片处理费用

  未超出免费额度时，不产生费用；超出免费额度后，按处理的原图实际大小计费。计费详情，请参见数据处理费用。

- 请求费用

  处理图片时会产生一次Get Object请求，按请求次数收费。计费详情，请参见请求费用。

- 流量费用

  根据处理的原图大小收取外网流出流量费用。计费详情，请参见流量费用。

版本说明

图片处理服务目前提供新版和旧版两个版本的API接口，本文档介绍新版接口的使用，旧版接口的功能今后不再更新。有关新旧版本接口使用兼容性的详细说明，请参见新旧版本图片处理服务及使用说明。

10.2.2. 图片处理操作方式

您可以通过文件URL、API、SDK对OSS内图片进行处理，本文介绍如何使用这三种方式进行图片处理。

通过文件URL处理图片

通过文件URL处理图片有两种方式，一种是添加图片处理参数，另一种是添加图片样式参数。

- 对于允许匿名访问的公共读或者公共读写文件，可直接在文件URL中通过添加图片处理参数或者图片样式参数的方式处理图片。
- 对于不允许匿名访问的私有图片文件，您需要通过SDK的方式将图片处理操作加入签名URL中。

> **注意** 通过文件URL访问图片时，默认是下载行为。如需确保通过文件URL访问图片时是预览行为，您需要绑定自定义域名并添加CNAME记录。具体操作，请参见绑定自定义域名。

公共读或者公共读写图片文件

通过在文件URL中通过添加图片处理参数或者图片样式参数的方式处理图片的说明如下。

图片处理方式	添加图片处理参数	添加图片样式参数
图片处理URL	`https://bucketname.endpoint/objectname?x-oss-process=image/action,parame_value`	`https://bucketname.endpoint/objectname?x-oss-process=style/stylename`
参数说明	• `https://bucketname.endpoint/objectname` ：Object的访问地址。获取方式，请参见上传Object后如何获取访问URL？。 • `x-oss-process=image/` ：固定参数，表明使用图片处理参数对图片文件进行处理。 • `action,parame_value` ：图片处理的操作（action）、参数（parame）和值（value），用于定义图片处理的方式。多个操作以正斜线（/）隔开，OSS将按图片处理参数的顺序处理图片。例如 `image/resize,w_200/rotate,90` 表示将图片先按比例缩放至宽200 px，再将图片旋转90°。图片处理支持的参数，请参见处理参数。	• `https://bucketname.endpoint/objectname` ：Object的访问地址。获取方式，请参见上传Object后如何获取访问URL？。 • `x-oss-process=style/` ：固定参数，表明使用图片样式参数对图片文件进行处理。 • `stylename` ：您提前在OSS控制台设置的样式名称。配置方法，请参见创建样式。 如果您设置了自定义分隔符，可使用分隔符代替 `?x-oss-process=style/` 内容，进一步简化图片处理URL。例如分隔符设置为感叹号（!），则图片处理URL为： `<https://bucketname.endpoint/objectname!stylename` 。自定义分隔符的配置方式，请参见设置自定义分隔符。
示例	https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_300/quality,q_90	https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/panda_style

私有图片文件

以下仅列举常见SDK的生成带图片处理参数的文件签名URL的代码示例。关于其他SDK的生成带图片处理参数的文件签名URL的代码示例，请参见SDK简介。

Python

```
// yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
String endpoint = "yourEndpoint";
// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 填写Bucket名称。
String bucketName = "examplebucket";
// 填写Object完整路径。Object完整路径中不能包含Bucket名称。
String objectName = "exampleobject.jpg";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 将图片缩放为固定宽高100 px后，再旋转90°。
String style = "image/resize,m_fixed,w_100,h_100/rotate,90";
// 指定签名URL过期时间为10分钟。
Date expiration = new Date(new Date().getTime() + 1000 * 60 * 10 );
GeneratePresignedUrlRequest req = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.GET);
req.setExpiration(expiration);
req.setProcess(style);
URL signedUrl = ossClient.generatePresignedUrl(req);
System.out.println(signedUrl);
// 关闭OSSClient。
ossClient.shutdown();
```

使用SDK处理图片

您可以通过在SDK中添加图片处理参数或图片样式参数的方式来处理图片。

添加图片处理参数

通过添加图片处理参数的方式处理图片时，多个图片处理参数之间须以正斜线（/）分隔。以下仅列举常见SDK的添加图片处理参数的代码示例，关于其他SDK的添加图片处理参数的代码示例，请参见简介。

Python

```
// yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
String endpoint = "yourEndpoint";
// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 填写Bucket名称。
String bucketName = "examplebucket";
// 填写Object完整路径。Object完整路径中不能包含Bucket名称。
String objectName = "exampleobject.jpg";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 将图片缩放为固定宽高100 px后，再旋转90°。
String style = "image/resize,m_fixed,w_100,h_100/rotate,90";
GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
// 将处理后的图片命名为example-new.jpg并保存到本地。
// 填写本地文件的完整路径，例如D:\\localpath\\example-new.jpg。如果指定的本地文件存在会覆盖，不存在则新建。
// 如果未指定本地路径只填写了文件名称（例如example-new.jpg），则文件默认保存到示例程序所属项目对应本地路径中。
ossClient.getObject(request, new File("D:\\localpath\\example-new.jpg"));
// 关闭OSSClient。
ossClient.shutdown();
```

添加图片样式

您可以通过图片样式将多个图片处理参数封装在一个样式中，然后使用图片样式批量处理图片。以下仅列举常见SDK的使用图片样式处理图片的代码示例，关于其他SDK的使用图片样式处理图片的代码示例，请参见简介。

Python

```
// yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
String endpoint = "yourEndpoint";
// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 填写Bucket名称。
String bucketName = "examplebucket";
// 填写Object完整路径。Object完整路径中不能包含Bucket名称。
String objectName = "exampleobject.jpg";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 使用自定义样式处理图片。
// yourCustomStyleName填写通过OSS管理控制台创建的图片样式名称。
String style = "style/yourCustomStyleName";
GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
// 将处理后的图片命名为example-new.jpg并保存到本地。
// 填写本地文件的完整路径，例如D:\\localpath\\example-new.jpg。如果指定的本地文件存在会覆盖，不存在则新建。
// 如果未指定本地路径只填写了文件名称（例如example-new.jpg），则文件默认保存到示例程序所属项目对应本地路径中。
ossClient.getObject(request, new File("D:\\localpath\\example-new.jpg"));
// 关闭OSSClient。
ossClient.shutdown();
```

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。

您可以通过在Get Object接口中添加图片处理参数或图片样式参数的方式来处理图片。更多信息，请参见Get Object。

- 添加图片处理参数

  请求示例如下：

```
GET /oss.jpg?x-oss-process=image/resize,w_100 HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 28 Oct 2022 06:40:10 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:UNQDb7GapEgJkcde6OhZ9J****
```

- 添加图片样式参数

  请求示例如下：

```
GET /oss.jpg?x-oss-process=style/styleexample HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 28 Oct 2022 06:40:10 GMT
Authorization: OSS qn6qrrawuk53oqxo2otf****:UNapEgQDb7GJkcde6OhZ9J****
```

更多参考

图片处理服务默认不保存处理后的图片。您可以在图片处理的请求内添加转存参数，将处理后的图片作为Object保存至指定的Bucket内。具体操作，请参见图片处理持久化。

10.2.3. 图片处理参数

10.2.3.1. 图片缩放

您可以通过图片缩放参数，调整OSS内存储的图片大小。本文介绍对象存储OSS图片处理中的图片缩放功能参数及示例。

参数说明

操作名称：`resize`

相关参数如下：

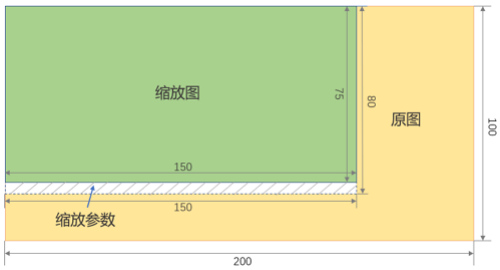
- 指定宽高缩放

名称	是否必须	描述	取值范围
m	是	指定缩放的模式。	○ lfit（默认值）：等比缩放，缩放图限制为指定w与h的矩形内的最大图片。 ○ mfit：等比缩放，缩放图为延伸出指定w与h的矩形框外的最小图片。 ○ fill：将原图等比缩放为延伸出指定w与h的矩形框外的最小图片，之后将超出的部分进行居中裁剪。 ○ pad：将原图缩放为指定w与h的矩形内的最大图片，之后使用指定颜色居中填充空白部分。 ○ fixed：固定宽高，强制缩放。 更多信息请参见表格下方示例。 说明 当取值为lfit或mfit进行等比缩放时，如果等比为小数，则四舍五入保留整数。
w	是	指定目标缩放图的宽度。	[1,4096]
h	是	指定目标缩放图的高度。	[1,4096]
l	是	指定目标缩放图的最长边。 说明 长边是指原尺寸与目标尺寸的比值大的那条边；短边是指原尺寸与目标尺寸的比值小的那条边。例如原图为400 px*200 px，缩放为800 px*100 px。由于（400/800）<（200/100），所以在这个缩放操作中，200那条是长边，400那条是短边。	[1,4096]
s	是	指定目标缩放图的最短边。	[1,4096]
limit	否	指定当目标缩放图大于原图时是否进行缩放。	0、1 ○ 1（默认值）：表示不按指定参数进行缩放，直接返回原图。 ○ 0：按指定参数进行缩放。
color	是（仅当 `m`为`pad` 时）	当缩放模式选择为pad（缩放填充）时，可以设置填充的颜色。	RGB颜色值，例如：000000表示黑色，FFFFFF表示白色。 默认值：FFFFFF（白色）

示例：原图大小为200 px*100 px，缩放参数为w=150 px，h=80 px。则不同的缩略模式，得到的缩放图如下：

- lfit
  - 等比缩放：要求缩放图的w/h等于原图的w/h。所以，若w=150 px，则h=75 px；若h=80 px，则w=160 px。
  - 限制在指定w与h的矩形内的最大图片：即缩放图的w*h不能大于150 px*80 px。

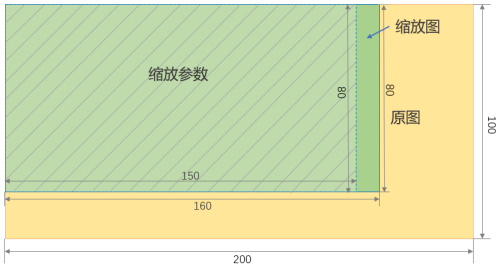
  通过以上条件得出缩略图大小为150 px*75 px。



lfit

- mfit
  - 等比缩放：要求缩放图的w/h等于原图的w/h。所以，若w=150 px，则h=75 px；若h=80 px，则w=160 px。
  - 延伸出指定w与h的矩形框外的最小图片：即缩放图必须是大于150 px*80 px的一个最小矩形。

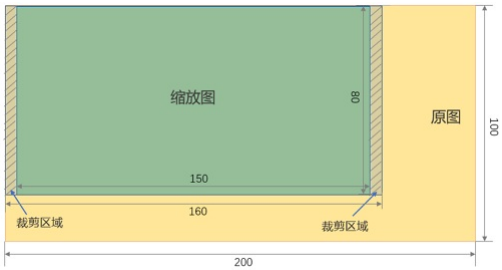
  通过以上条件得出缩放图的大小为160 px*80 px。



mfit

- fill

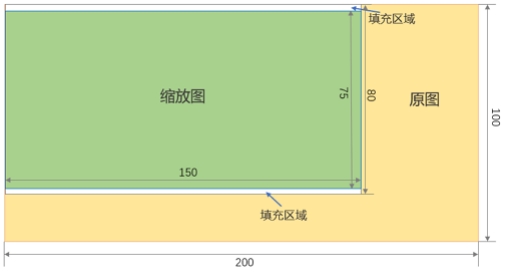
  fill参数会先将图片等比缩放为延伸出指定w与h的矩形框外的最小图片，之后按照固定宽高进行裁剪。即先将原图缩放为160 px*80 px，之后将w居中裁剪为150 px，得到大小为150 px*80 px的缩放图。



fill

- pad

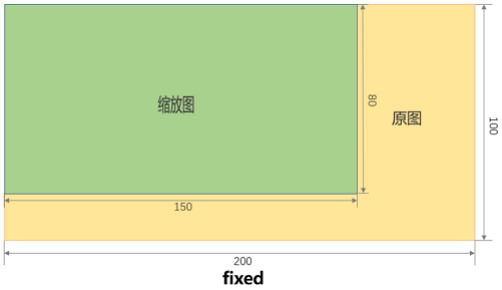
  pad参数会先将图片等比缩放为限制在指定w与h的矩形内的最大图片，之后按照固定宽高进行填充。即先将原图缩放为150 px*75 px，之后将h居中填充到80 px，得到大小为150 px*80 px的缩放图。



pad

- fixed

  fixed参数会将图片按照固定宽高进行缩放，若宽高与原图宽高比例不同，则会导致图片变形。



- 按比例缩放

名称	是否必须	描述	取值范围
p	是	按百分比缩放图片。	[1,1000] 小于100为缩小，大于100为放大。

注意事项

- 原图限制
  - 图片格式只能是：JPG、PNG、BMP、GIF、WebP、TIFF。其中GIF格式的图片支持指定宽高缩放，不支持等比缩放（等比缩放情况下，动态图会变成静态图）。
  - 原图大小不能超过20 MB。
  - 宽或高不能超过30,000 px，且总像素不能超过2.5亿 px。

    动态图片（例如GIF图片）的像素计算方式为 `宽*高*图片帧数` ；非动态图片（例如PNG图片）的像素计算方式为 `宽*高` 。
- 缩放图限制

  宽或高不能超过16,384 px，且总像素不能超过16,777,216 px。
- 缩放优先级

  如果图片处理URL中同时指定按宽高缩放和等比缩放参数，则只执行指定宽高缩放。
- 缩放时只指定宽度或者高度
  - 等比缩放时，会按比例缩放图片。例如原图为200 px*100 px，将高缩放为100 px，则宽缩放为50 px。
  - 固定宽高缩放时，会将原图宽高按照指定值进行缩放。例如原图为200 px*100 px，将高缩放为100 px，则宽也缩放为100 px。
- 如果缩放模式为mfit，且为目标缩放图的最长边*l*或者目标缩放图的最短边*s*指定了值，则*l*和*s*都会应用指定的值进行缩放。
- 如果指定了缩放模式*m*，且为目标缩放图的宽度*w*或目标缩放图的高度*h*指定了值，则目标缩放图的最长边*l*或目标缩放图的最短边*s*的取值不会生效。
- 目标缩放图比原图尺寸大时，默认返回原图。您可以增加 `limit_0` 参数放大图片。例如， `https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_500,limit_0`

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 等比缩放
  - 按宽高缩放

    需求及处理参数如下：
    - 图片缩放为高100 px： `resize,h_100`
    - 缩放模式为lfit： `m_lfit`

    图片处理的URL为http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,h_100,m_lfit

- 按长边缩放

  需求及处理参数为：图片缩放为长边100 px，即 `resize,l_100`

  图片处理的URL为http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,l_100

- 固定宽高缩放

  需求及处理参数如下：

  - 将原图缩放成宽高100 px：`resize,h_100,w_100`
  - 缩放模式fixed：`m_fixed`

  图片处理的URL为：http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,m_fixed,h_100,w_100

- 固定宽高，自动裁剪

  需求及处理参数如下：

  - 将原图缩放成宽高100 px：`resize,h_100,w_100`
  - 缩放模式fill：`m_fill`

  图片处理的URL为：http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,m_fill,h_100,w_100

- 固定宽高，缩放填充

  需求及处理参数如下：

  - 将原图缩放成宽高100 px：`resize,h_100,w_100`
  - 缩放模式pad：`m_pad`
  - 以红色填充：`color_FF0000`

  图片处理的URL为：http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,m_pad,h_100,w_100,color_FF0000

- 按比例缩放

  需求及处理参数如下：

  将原图缩放50%：`resize,p_50`

  图片处理的URL为：http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,p_50

常见问题

压缩后的图片读写权限为私有，如何正常访问？

必须对图片文件URL完成签名操作后才能正常访问。具体操作，请参见上传Object后如何获取访问URL。

10.2.3.2. 图片水印

您可以通过图片水印参数，为您存储在OSS中的图片文件增加水印文字或水印图。本文介绍为图片添加水印时所用到的参数及示例。

注意事项

- 图片水印只能使用当前存储空间内的图片，网络或本地图片需上传至当前存储空间内方可使用。
- 图片水印目前仅支持JPG、PNG、BMP、WebP、TIFF格式。
- 单张图片最多支持添加3张不同的图片水印，且各个图片水印的位置不能完全重叠。
- 文字水印暂不支持繁体中文。

参数说明

操作名称：watermark

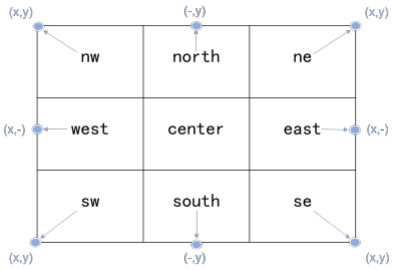
相关参数如下：

- 基础参数

参数	是否必须	描述	取值范围
t	否	指定图片水印或水印文字的透明度。	[0,100] 默认值：100， 表示透明度100%（不透明）。
g	否	指定水印在图片中的位置。	○ nw：左上 ○ north：中上 ○ ne：右上 ○ west：左中 ○ center：中部 ○ east：右中 ○ sw：左下 ○ south：中下 ○ se（默认值）：右下 详情请参见下方基准点图片。
x	否	指定水印的水平边距， 即距离图片边缘的水平距离。这个参数只有当水印位置是左上、左中、左下、右上、右中、右下才有意义。	[0,4096] 默认值：10 单位：像素（px）
y	否	指定水印的垂直边距，即距离图片边缘的垂直距离， 这个参数只有当水印位置是左上、中上、右上、左下、中下、右下才有意义。	[0,4096] 默认值：10 单位：px
voffset	否	指定水印的中线垂直偏移。当水印位置在左中、中部、右中时，可以指定水印位置根据中线往上或者往下偏移。	[-1000,1000] 默认值：0 单位：px

水平边距、垂直边距、中线垂直偏移不仅可以调节水印在图片中的位置，当图片存在多重水印时，还可以调节水印在图中的布局。

区域数值以及每个区域对应的基准点如下图所示。



- 图片水印参数

参数	是否必须	描述	取值范围
image	是	用于指定作为图片水印Object的完整名称，Object名称需进行Base64编码。详情请参见水印编码。 例如，作为图片水印的Object为Bucket内*image*目录下的*panda.png*，则需要编码的内容为*image/panda.png*，编码后的字符串为 `aW1hZ2UvcGFuZGEucG5n` 。 **说明** 水印图片只能使用当前存储空间内的Object。	Base64编码后的字符串。

- 水印图片预处理参数

您可以使用图片缩放、自定义裁剪、索引切割、圆角矩形及图片旋转操作中的所有参数对水印图片进行预处理。此外，水印图片在进行缩放操作时，还额外支持参数P：

参数	描述	取值范围
P	指定图片水印按照原图的比例进行缩放，取值为缩放的百分比。如设置参数值为10，如果原图为100×100， 则图片水印大小为10×10。当原图变成了200×200，则图片水印大小为20×20。	[1,100]

- 文字水印参数

参数	是否必须	描述	取值范围
text	是	指定文字水印的文字内容，文字内容需进行Base64编码。详情请参见水印编码。	Base64编码之前中文字符串的最大字节长度为64个字符。

参数	是否必须	描述	取值范围
type	否	指定文字水印的字体，字体名称需进行Base64编码。	支持的字体及字体编码详情请参见文字类型编码对应表。 默认值：wqy-zenhei（编码后的值为d3F5LXplbmhlaQ）
color	否	指定文字水印的文字颜色，参数值为RGB颜色值。	RGB颜色值，例如：000000表示黑色，FFFFFF表示白色。 默认值：000000（黑色）
size	否	指定文字水印的文字大小。	(0,1000] 默认值：40 单位：px
shadow	否	指定文字水印的阴影透明度。	[0,100] 默认值：0，表示没有阴影。
rotate	否	指定文字顺时针旋转角度。	[0,360] 默认值：0，表示不旋转。
fill	否	指定是否将文字水印铺满原图。	0、1 ○ *1*：表示将文字水印铺满原图。 ○ *0*（默认值）：表示不将文字水印铺满全图。

type参数中可选的文字类型及编码如下表所示。

参数值	中文含义	编码值
wqy-zenhei	文泉驿正黑	d3F5LXplbmhlaQ
wqy-microhei	文泉微米黑	d3F5LW1pY3JvaGVp
fangzhengshusong	方正书宋	ZmFuZ3poZW5nc2h1c29uZw
fangzhengkaiti	方正楷体	ZmFuZ3poZW5na2FpdGk
fangzhengheiti	方正黑体	ZmFuZ3poZW5naGVpdGk
fangzhengfangsong	方正仿宋	ZmFuZ3poZW5nZmFuZ3Nvbmc
droidsansfallback	DroidSansFallback	ZHJvaWRzYW5zZmFsbGJhY2s

- 图文混合水印参数

参数	是否必须	描述	取值范围
order	否	指定文字和图片水印的前后顺序。	0、1 ○ 0（默认值）：表示图片水印在前。 ○ 1：表示文字水印在前。
align	否	指定文字水印和图片水印的对齐方式。	0、1、2 ○ 0：表示文字水印和图片水印上对齐。 ○ 1：表示文字水印和图片水印中对齐。 ○ 2（默认值）：表示文字水印和图片水印下对齐。
interval	否	指定文字水印和图片水印间的间距。	[0,1000] 默认值：0 单位：px

水印编码

在添加水印操作中，文字水印的文字内容、文字颜色、文字字体、图片水印的水印图片名称等参数需要进行URL安全的Base64编码。编码步骤如下：

1. 将内容编码成Base64。
2. 将结果中的部分编码替换。
   - 将结果中的加号（+）替换成短划线（-）。
   - 将结果中的正斜线（/）替换成下划线（_）。
   - 将结果中尾部的等号（=）省略。

推荐通过URL-safe Baes64编码工具对文字水印的文字内容、文字颜色、文字字体、图片水印的水印图片名称等参数进行编码。

> **注意** 水印编码后的内容仅应用在水印操作的特定参数中，请勿将其用在签名字符串（Signature）中。

示例一：添加文字水印

以华北3（张家口）地域名为image-demo的Bucket中的图片example.jpg为例，图片访问URL为https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg

为example.jpg图片添加文字水印示例如下：

- 快速添加Hello World的文字水印

  对文字水印的内容Hello World进行URL安全的Base64位编码。具体操作，请参见水印编码。编码结果为 `SGVsbG8gV29ybGQ` ，图片处理URL为https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-process=image/watermark,text_SGVsbG8gV29ybGQ。

- 添加文字水印时配置多个图片处理参数

  为example.jpg图片添加Hello World的文字水印的同时，需要对水印文字以及原图做如下相应处理：

  - 将*example.jpg*缩略为宽高300： `resize,w_300,h_300`
  - 水印文字字体为文泉驿正黑： `type_d3F5LXplbmhlaQ` （d3F5LXplbmhlaQ是文泉驿正黑经过Base64编码后的值）
  - 水印内容为“Hello World”： `text_SGVsbG8gV29ybGQ`
  - 水印文字颜色为白色、字体大小为30： `color_FFFFFF,size_30`
  - 文字阴影透明度为50%： `shadow_50`
  - 水印文字位置是右下、水平边距10、中线垂直偏移10： `g_se,x_10,y_10`

  图片处理的URL为：https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_300,h_300/watermark,type_d3F5LXplbmhlaQ,size_30,text_SGVsbG8gV29ybGQ,color_FFFFFF,shadow_50,t_100,g_se,x_10,y_10

示例二：添加图片水印

为example.jpg图片添加图片水印示例如下：

- 快速添加名为panda.png的水印图片

  对水印图片名称panda.png进行URL安全的Base64位编码，编码结果为 `cGFuZGEucG5n` ，图片处理URL为https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-process=image/watermark,image_cGFuZGEucG5n。

- 添加图片水印时配置多个图片处理参数

  为example.jpg图片添加图片水印panda.png的同时，需要对图片水印以及原图做如下相应处理：

  - 将example.jpg缩略为宽高300： `resize,w_300,h_300`

- 将example.jpg图片质量设为90%：`quality,q_90`
- 添加水印图片*panda.png*：`watermark,image_cGFuZGEucG5n`（cGFuZGEucG5n是panda.png进行Base64编码后的值）
- 水印图片透明度90%：`t_90`
- 水印图片位于原图的右下方、水平边距10、中线垂直偏移10：`g_se,x_10,y_10`

图片处理的URL为：https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_300,h_300/quality,q_90/watermark,image_cGFuZGEucG5n,t_90,g_se,x_10,y_10

- 对图片水印进行预处理后配置多个图片处理参数

  为example.jpg图片添加图片水印panda.png的同时，需要对图片水印以及原图做如下相应处理：

  - 将*example.jpg*缩略为宽300：`resize,w_300`
  - 将水印图片*panda.png*进行预处理（缩放30%）：`image_cGFuZGEucG5nP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMzA`（`cGFuZGEucG5nP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMzA`为`panda.png?x-oss-process=image/resize,P_30`经过Base64编码后的值）
  - 水印的透明度为90%、位置是右下、水平边距是10、中线垂直偏移是10：`t_90,g_se,x_10,y_10`

  图片处理的URL为：https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_300/watermark,image_cGFuZGEucG5nP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMzA,t_90,g_se,x_10,y_10

示例三：添加图片和文字混合水印

为example.jpg图片添加图片和文字混合水印的示例如下：

- 快速添加panda.png图片水印和Hello World文字水印

  结合以上示例中panda.png以及Hello World的编码结果，可得出图片处理的URL为https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-process=image/watermark,image_cGFuZGEucG5nP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMzA,text_SGVsbG8gV29ybGQ。

- 添加多个图片和文字水印

  为example.jpg图片添加2个不同的文本水印（Watermark 1和Watermark 2）和3个图片水印（Jellyfish.jpg、Koala.jpg以及Tulips.jpg）。添加多个水印时，需使用正斜线（/）将不同的水印操作隔开。

  - 添加Jellyfish.jpg图片水印，对图片水印进行预处理（缩放20%），图片水印位于原图的左上角，水平边距10，中线垂直偏移10。图片参数处理结果为`watermark,image_cGljcy9KZWxseWZpc2guanBnP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMjA,g_nw,x_10,y_10`，其中`cGljcy9KZWxseWZpc2guanBnP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMjA`为Jellyfish.jpg图片水印预处理后经过Base64编码后的值。
  - 添加Koala.jpg图片水印，对图片水印进行预处理（缩放20%），图片水印位于原图右下角，水平边距10，中线垂直偏移10。图片参数处理结果为`watermark,image_cGljcy9Lb2FsYS5qcGc_eC1vc3MtcHJvY2Vzcz1pbWFnZS9yZXNpemUsUF8yMA,g_se,x_10,y_10`，其中`cGljcy9Lb2FsYS5qcGc_eC1vc3MtcHJvY2Vzcz1pbWFnZS9yZXNpemUsUF8yMA`为Koala.jpg图片水印预处理后经过Base64编码后的值。
  - 添加Tulips.jpg图片水印，对图片水印进行预处理（缩放20%），图片水印位于原图左中部，水平边距10，中线垂直偏移10。图片参数处理结果为`watermark,image_cGljcy9UdWxpcHMuanBnP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMjA,g_west,x_10,y_10`，其中`cGljcy9UdWxpcHMuanBnP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMjA`为Tulips.jpg图片水印预处理后经过Base64编码后的值。
  - 添加Watermark 1文字水印，字体大小为20，水印文字位于原图的右上角，水平边距10、中线垂直偏移200。图片参数处理结果为`watermark,text_V2F0ZXJtYXJrIDE,g_ne,size_20,x_10,y_200`，其中`V2F0ZXJtYXJrIDE`为Watermark 1经过Base64编码后的值。
  - 添加Watermark 2文字水印，字体大小为20，颜色为深蓝色，水印文字位于原图的左下角，水平边距100、中线垂直偏移50。图片参数处理结果为`watermark,text_V2F0ZXJtYXJrIDI,color_0000b7,size_20,g_sw,x_100,y_50`，其中`V2F0ZXJtYXJrIDI`为Watermark 2经过Base64编码后的值。

  图片处理的URL为：https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-

process=image/watermark,image_cGljcy9KZWxseWZpc2guanBnP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMjA,g_nw,x_10,y_10/watermark,image_cGljcy9Lb2FsY

常见问题

如何使用网络图片或本地图片作为水印图片？

通过OSS的图片处理为图片添加图片水印时，仅可以使用相同存储空间内的图片作为水印图片。若您希望使用网络图片或本地图片作为水印图片，需要先将图片上传到原图所在存储空间，之后再使用上传的图片作为水印图片处理原图。

10.2.3.3. 自定义裁剪

您可以通过自定义裁剪参数，在OSS存储的原图上裁剪指定大小的矩形图片。本文介绍自定义裁剪图片时所用到的参数及示例。

参数说明

操作名称：crop

参数说明如下：

参数	描述	取值范围
w	指定裁剪宽度。	[0,图片宽度] 默认为最大值。
h	指定裁剪高度。	[0,图片高度] 默认为最大值。
x	指定裁剪起点横坐标（默认左上角为原点）。	[0,图片边界]
y	指定裁剪起点纵坐标（默认左上角为原点）。	[0,图片边界]
g	设置裁剪的原点位置。原点按照九宫格的形式分布，一共有九个位置可以设置，为每个九宫格的左上角顶点。	• nw：左上 • north：中上 • ne：右上 • west：左中 • center：中部 • east：右中 • sw：左下 • south：中下 • se：右下 详情请参见下方裁剪原点位置参数示意图。

裁剪原点位置参数示意图如下。

nw	north	ne
west	center	east
sw	south	se

注意事项

在使用自定义裁剪功能时，请注意以下事项：

- 如果指定起点的横纵坐标大于原图，将会返回 `BadRequest` 错误，错误信息为：Advance cut's position is out of image。
- 如果从起点开始指定的宽度和高度超过了原图，将会直接裁剪到原图边界为止。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 从（100,50）开始，裁减至图片边界

  需求及处理参数如下：

  - 裁剪起点为（100,50）：`crop,x_100,y_50`
  - 裁减至图片边界：裁剪时默认使用w和h的最大值，所以可省略w和h参数。

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/crop,x_100,y_50

- 从（100，50）开始，裁剪100 px*100 px大小的图片

  需求及处理参数如下：

  - 裁剪起点为（100,50）：`crop,x_100,y_50`
  - 裁减范围100 px*100 px：`w_100,h_100`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/crop,x_100,y_50,w_100,h_100

- 裁剪原图右下角200 px*200 px的范围

  需求及处理参数如下：

  - 裁剪起点为原图右下角：`crop,g_se`
  - 裁减范围200 px*200 px：`w_200,h_200`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/crop,w_200,h_200,g_se

- 裁剪图右下角200 px*200 px的范围，起点为相对右下九宫格的左上顶点再位移（10,10）

  需求及处理参数如下：

  - 起点为原图右下角再位移（10,10）：`crop,g_se,x_10,y_10`
  - 裁减范围200 px*200 px：`w_200,h_200`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/crop,x_10,y_10,w_200,h_200,g_se

10.2.3.4. 质量变换

质量变换操作是使用原图本身的格式对图片进行压缩。您可以通过质量变换参数，修改存储在OSS内原图的质量。本文介绍对图片进行质量变换时所用到的参数及示例。

质量变换仅支持JPG和WebP，其他图片格式不支持。

参数说明

操作名称：quality

参数说明如下：

参数	描述	取值范围
q	设置图片的相对质量，对原图按百分比进行质量压缩。 例如原图质量为100%，添加 `quality,q_90` 参数会得到质量为90%的图片。原图质量为80%，添加 `quality,q_90` 参数会得到质量72%的图片。 **说明** 只有JPG格式的原图添加该参数，才可以决定图片的相对质量。如果原图为WebP格式，添加该参数相当于指定了原图绝对质量，即与参数Q的作用相同。	[1,100]
Q	设置图片的绝对质量，将原图质量压缩至Q%，如果原图质量小于指定参数值，则按照原图质量重新进行压缩。 例如原图质量是95%，添加 `quality,Q_90` 参数会得到质量90%的图片。原图质量是80%，添加 `quality,Q_90` 只能得到质量80%的图片。 **说明** 该参数只能对保存格式为JPG、WebP的图片使用，对其他格式的图片无效果。	[1,100]

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 变换图片相对质量

  需求及处理参数如下：

  - 原图缩放为宽100 px：`resize,w_100`
  - 图片相对质量设置为80%：`quality,q_80`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/quality,q_80

- 变换图片绝对质量

  需求及处理参数如下：

  - 原图缩放为宽100 px：`resize,w_100`
  - 图片绝对质量设置为80%：`quality,Q_80`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/quality,Q_80

10.2.3.5. 格式转换

您可以通过格式转换参数，转换存储在OSS内原图的格式。本文介绍对图片进行格式转换时所用到的参数及示例。

参数说明

操作名称：format

参数说明如下：

取值范围	描述

取值范围	描述
jpg	将原图保存为JPG格式，如果原图是PNG、WebP、BMP等存在透明通道的格式，默认会把透明填充成白色。 注意 不支持将存在透明通道的HEIC格式的图片保存为JPG格式。
png	将原图保存为PNG格式。
webp	将原图保存为WebP格式。
bmp	将原图保存为BMP格式。
gif	原图为GIF图片则继续保存为GIF格式；原图不是GIF图片，则按原图格式保存。
tiff	将原图保存为TIFF格式。

注意事项

- 图片处理包含缩放操作时，建议将格式转换参数放到处理参数的最后。

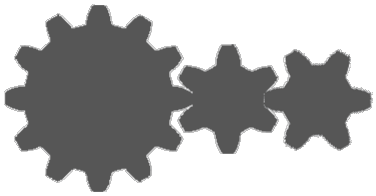
  例如 `image/resize,w_100/format,jpg`
- 图片处理包含缩放和水印操作时，建议将格式转换参数添加在缩放参数之后。

  例如 `image/reisze,w_100/format,jpg/watermark,...`

示例

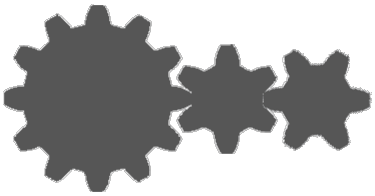
本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.gif



- 将原图转换为PNG格式

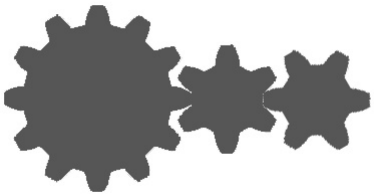
  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.gif?x-oss-process=image/format,png



- 将原图转换成JPG格式，并支持渐进显示

  需求及处理参数如下：
  - 图片设置为渐进显示：`interlace,1`
  - 图片转换为JPG格式：`format,jpg`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.gif?x-oss-process=image/interlace,1/format,jpg



- 将原图缩放为宽200 px，并转换为WebP格式

  需求及处理参数如下：

  - 图片缩放为宽200 px：`resize,w_200`
  - 图片转换为WebP格式：`format,webp`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.gif?x-oss-process=image/resize,w_200/format,webp

10.2.3.6. 获取信息

部分图片可能包含可交换图像文件EXIF信息，该信息主要用于记录数码照片的属性信息和拍摄数据。如果您希望获取图片的EXIF信息，请在图片URL中添加info参数。

> **说明** EXIF信息包括压缩比Compression、方向Orientation、水平分辨率XResolution、垂直分辨率YResolution等。有关EXIF的更多信息，请参见EXIF2.31。

参数说明

操作名称：info

返回的图片信息为JSON格式。

示例

- 获取不包含EXIF信息的原图示例

  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/info

  当图片中不包含EXIF信息，则在文件URL中添加info参数时，仅返回图片的基本信息，例如图片大小、格式、图片高度以及图片宽度等。

```
{
  "FileSize": {"value": "21839"},
  "Format": {"value": "jpg"},
  "ImageHeight": {"value": "267"},
  "ImageWidth": {"value": "400"}
}
```

- 获取包含EXIF信息的原图示例

  http://image-demo.oss-cn-hangzhou.aliyuncs.com/f.jpg?x-oss-process=image/info

  当图片中包含EXIF信息，则在文件URL中添加info参数时，返回图片的基本信息以及EXIF信息。

```
{
  "Compression": {"value": "6"},
  "DateTime": {"value": "2015:02:11 15:38:27"},
  "ExifTag": {"value": "2212"},
  "FileSize": {"value": "23471"},
  "Format": {"value": "jpg"},
  "GPSLatitude": {"value": "0deg "},
  "GPSLatitudeRef": {"value": "North"},
  "GPSLongitude": {"value": "0deg "},
  "GPSLongitudeRef": {"value": "East"},
  "GPSMapDatum": {"value": "WGS-84"},
  "GPSTag": {"value": "4292"},
  "GPSVersionID": {"value": "2 2 0 0"},
  "ImageHeight": {"value": "333"},
  "ImageWidth": {"value": "424"},
  "JPEGInterchangeFormat": {"value": "4518"},
  "JPEGInterchangeFormatLength": {"value": "3232"},
  "Orientation": {"value": "7"},
  "ResolutionUnit": {"value": "2"},
  "Software": {"value": "Microsoft Windows Photo Viewer 6.1.7600.16385"},
  "XResolution": {"value": "96/1"},
  "YResolution": {"value": "96/1"}}
```

10.2.3.7. 自适应方向

您可以通过自适应方向参数，指定OSS内存储的原图是否按自适应方向旋转。本文介绍进行自适应方向旋转时所用到的参数及示例。

参数说明

操作名称：auto-orient

参数说明如下：

参数	描述	取值
[value]	指定图片是否进行自适应旋转。	0、1 • 0：保持原图方向，不进行自适应旋转。 • 1：将图片进行自适应旋转。

注意事项

- 如果原图没有旋转参数（Orientation），添加auto-orient操作不会对图片进行旋转。
- 目前，大多数工具都会对携带旋转参数的图片进行自适应旋转，所以您看到的图片可能是经过自适应旋转后的图片。
- 添加auto-orient参数处理后的图片会重新压缩，导致与原图大小不一致。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/f.jpg

- 缩放图片并维持图片旋转方向

  需求及处理参数如下：

  - 将图片缩略为宽100 px：`resize,w_100`
  - 图片不进行自动旋转：`auto-orient,0`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/f.jpg?x-oss-process=image/resize,w_100/auto-orient,0

- 缩放图片并进行自适应旋转

  需求及处理参数如下：

  - 将图片缩略为宽100 px：`resize,w_100`
  - 图片进行自动旋转：`auto-orient,1`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/f.jpg?x-oss-process=image/resize,w_100/auto-orient,1

10.2.3.8. 内切圆

您可以通过内切圆参数，将OSS内存储的图片处理成内切圆。本文介绍对象存储OSS图片处理中的内切圆功能参数及示例。

参数说明

操作名称：circle

相关参数如下：

参数	描述	取值范围
r	指定内切圆的半径。	[1,4096]

注意事项

- 如果图片的最终格式是PNG、WebP或BMP等支持透明通道的图片，那么图片非圆形区域的部分将会以透明填充。如果图片的最终格式是JPG，那么非圆形区域是以白色进行填充。推荐保存成PNG格式。
- 当r取值大于原图最小边的一半时，以原图最小边的一半为值返回内切圆，即r=（原图最小边-1）÷2，返回图片的宽和高为r×2+1。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 裁剪半径为100，保存为JPG格式，外围以白色填充

  图片处理URL为：http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/circle,r_100

- 裁剪半径为100，保存为PNG格式，外围以透明色填充

  图片处理URL为：http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/circle,r_100/format,png

10.2.3.9. 索引切割

您可以通过索引切割参数，将OSS存储的原图按指定大小分割，并截取需要的图片。本文介绍索引切割所用到的参数及示例。

参数说明

操作名称：indexcrop

参数说明如下：

参数	描述	取值范围
x	指定在x轴切割出的每块区域的长度。x参数与y参数只能任选其一。	[1,图片宽度]
y	指定在y轴切割出的每块区域的长度。x参数与y参数只能任选其一。	[1,图片高度]
i	选择切割后返回的图片区域。	[0,区域数) 默认为0，表示第一块。

注意事项

- 如果指定的索引值大于切割后形成的区域数量，将返回原图。
- 当x和y同时指定且值合法时，以y参数的值为准。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

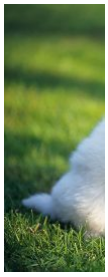
https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 按x轴切割图片

  需求及处理参数如下：

  - 将图片在x轴按100 px为单位切割：`indexcrop,x_100`
  - 选取切割后的第1块区域：`i_0`

图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/indexcrop,x_100,i_0



- 按y轴切割图片

  需求及处理参数如下：

  - 将图片在y轴按100 px为单位切割：`indexcrop,y_100`
  - 选取切割后的第11块区域：`i_10`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/indexcrop,y_100,i_10

  由于10大于切割后形成的区域数量，因此返回原图。

10.2.3.10. 圆角矩形

您可以通过圆角矩形参数，将存储在OSS内矩形图片的4个角切成圆角。本文介绍使用圆角矩形裁剪图片时所用到的参数及示例。

参数说明

操作名称：rounded-corners

参数说明如下：

参数	描述	取值范围
r	将图片切出圆角，指定圆角的半径。	[1,4096]

注意事项

- 如果图片的最终格式是PNG、WebP、BMP等支持透明通道的图片，那么图片圆角外的区域将会以透明填充。如果图片的最终格式是JPG，那么图片圆角外的区域以白色进行填充。推荐保存成PNG格式。
- 如果指定圆角的半径大于原图最大内切圆的半径，则按照图片最大内切圆的半径设置圆角（即r=原图最小边÷2）。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 使用圆角矩形裁剪原图

  需求及处理参数如下：

  - 裁剪圆角半径为30 px：`rounded-corners,r_30`
  - 保存格式为JPG：`format,jpg`（原图为JPG格式，该参数可省略）

图片处理的URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/rounded-corners,r_30

- 先将图片自定义裁剪后再进行圆角矩形裁剪，图片保存为PNG格式

  需求及处理参数如下：

  - 从默认起始位置将原图裁剪为100 px*100 px：`crop,w_100,h_100`
  - 裁剪圆角半径为10 px：`rounded-corners,r_10`
  - 保存格式为PNG：`format,png`

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/crop,w_100,h_100/rounded-corners,r_10/format,png

10.2.3.11. 模糊效果

您可以通过模糊参数，为存储在OSS内的原图增加模糊效果。本文介绍为图片添加模糊效果时所用到的参数及示例。

参数说明

操作名称：blur

参数说明如下：

参数	是否必须	描述	取值范围
r	是	设置模糊半径。	[1,50] 该值越大，图片越模糊。
s	是	设置正态分布的标准差。	[1,50] 该值越大，图片越模糊。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

对图片进行半径为3，标准差为2的模糊处理，则图片处理的URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/blur,r_3,s_2

10.2.3.12. 旋转

您可以通过旋转参数，将存储在OSS内的原图按指定方向旋转。本文介绍旋转图片时所用到的参数和示例。

参数说明

操作名称：rotate

参数说明如下：

参数	描述	取值范围
[value]	图片按顺时针旋转的角度。	[0,360] 默认值：0，表示不旋转。

注意事项

- 若图片旋转的角度不是90°、180°、270°、360°时，会导致处理后的图片尺寸变大。
- 旋转功能对图片的尺寸有限制，图片的宽或者高不能超过4096 px。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 将原图按顺时针旋转90°

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/rotate,90

- 将原图按顺时针旋转70°

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/rotate,70

10.2.3.13. 渐进显示

您可以通过渐进显示参数，将OSS内存储的原图修改为渐进显示。本文介绍设定图片渐进显示时所用到的参数及示例。

当网络环境较差或图片较大时，图片在网页上有两种显示方式：

- 标准显示：图片会按照从上到下的顺序一行一行地加载显示。
- 渐进显示：先显示整个图片的模糊轮廓，然后逐渐加载直至显示完整的图片。

目前，图片处理的渐进显示操作仅适用于将原图处理为JPG格式图片的情况，若原图不为JPG格式的图片，您需要增加 `format,jpg` 参数将图片改为JPG格式。

参数说明

操作名称：interlace

参数说明如下：

参数	描述	取值
[value]	指定是否设置图片为渐进显示。	0、1 • 1：表示将原图设置成渐进显示。 • 0：表示将原图设置成标准显示。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，Bucket的外网访问地址为：`https://image-demo.oss-cn-hangzhou.aliyuncs.com`，使用的图片是根目录下example.jpg和panda.png两张图片。

- 将原图格式为JPG的图片缩放成宽200 px，并设置成渐进显示

  需求及处理参数如下：

  - 图片缩放为宽200 px：`resize,w_200`
  - 图片设为渐进显示：`interlace,1`

  处理后的URL为：http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_200/interlace,1

- 将PNG格式的图片保存为JPG格式，之后设置成渐进显示

  需求及处理参数如下：

  - 图片转换为JPG格式：`format,jpg`
  - 图片设为渐进显示：`interlace,1`

  处理后的URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/panda.png?x-oss-process=image/format,jpg/interlace,1

10.2.3.14. 获取图片主色调

本文介绍获取图片主色调时所用到的参数及示例。

参数说明

操作名称：average-hue

返回的色调信息格式为：0xRRGGBB （RR、GG、BB都是十六进制数，表示红、绿、蓝三种颜色）

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

获取示例图片主色调的URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/average-hue

在浏览器中返回的平均色调信息为：0x5c783b，对应的颜色为RGB（92,120,59）。

10.2.3.15. 亮度

您可以通过亮度参数，调节存储在OSS内的原图亮度。本文介绍调节图片亮度时所用到的参数及示例。

参数说明

操作名称：bright

参数说明如下：

参数	描述	取值范围
[value]	指定图片的亮度。	[-100, 100] • 取值 < 0：降低图片亮度。 • 取值=0：不调整图片亮度。 • 取值 > 0：提高图片亮度。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 将图片亮度提高50

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/bright,50

- 将图片亮度降低50

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/bright,-50

10.2.3.16. 锐化

您可以通过锐化参数，提高存储在OSS内原图的清晰度。本文介绍对图片进行锐化时所用到的参数及示例。

参数说明

操作名称：sharpen

参数说明如下：

参数	描述	取值范围
[value]	设置锐化效果的强度。	[50,399] 取值越大，图片越清晰，但过大的值可能会导致图片失真。为达到较优效果，推荐取值为100。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

对原图进行锐化处理，锐化参数为100。图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/sharpen,100

10.2.3.17. 对比度

对比度是指一幅图像中明暗区域最亮的白和最暗的黑之间不同亮度层级的测量，即指一幅图像灰度反差的大小。您可以通过对比度参数，调整存储在OSS内原图的对比度。本文介绍调节图片对比度时所用到的参数及示例。

参数说明

操作名称：contrast

参数	描述	取值范围
[value]	指定图片的对比度。	[-100,100] • 取值＜0：降低图片对比度。 • 取值=0：维持原图对比度。 • 取值＞0：提高图片对比度。

示例

本文示例使用的Bucket为杭州地域名为image-demo的Bucket，图片外网访问地址为：

https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg

- 对比度降低50

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/contrast,-50

- 对比度提高50

  图片处理URL为：https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/contrast,50

10.2.4. 图片样式

您可以在一个样式（Style）中包含多个图片处理参数，快速实现复杂的图片处理操作。本文介绍如何创建和使用图片样式。

创建样式

一个存储空间（Bucket）最多可创建50个样式，这些样式仅支持作用于该Bucket下的图片文件。如您的业务有更多样式的需求，请联系技术支持。

1. 登录OSS管理控制台。
2. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**数据处理 > 图片处理**，然后单击**新建样式**。
4. 在**新建样式**面板，按以下说明新建样式。

   您可以使用**基础编辑**和**高级编辑**两种方式新建样式：

   - **基础编辑**：选择您需要的图片处理方式，例如缩放图片、添加水印、修改图片格式等。
   - **高级编辑**：使用API代码编辑图片处理方式，格式为 `image/action1,parame_value1/action2,parame_value2/...` 。

     例如，`image/resize,p_63/quality,q_90` 表示先将图片缩放到原图的63%，再设置图片相对质量为90%。

     > **说明** 如果您需要在样式中同时包含水印图片和水印文字的操作，请使用**高级编辑**新建样式。

   关于图片处理参数的更多信息，请参见简介。
5. 单击**确定**。

样式使用规则

图片样式配置完成后，您可以通过图片处理URL和阿里云SDK的方式使用样式来处理图片。

> **注意** 使用样式处理动态图片（如GIF格式的图片），需要在样式中加入格式转换参数/format,gif，否则可能会导致动态图片在处理后变为静态图。

使用图片处理URL

您可以直接将图片样式添加到图片的访问URL上，格式为 `http(s)//:BucketName.Endpoint/ObjectName?x-oss-process=style/<StyleName>` ，示例为https://image-demo-oss-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/example.jpg?x-oss-process=style/small。

如果您设置了自定义分隔符，可使用分隔符代替 `?x-oss-process=style/` 内容，进一步简化图片处理URL。关于设置自定义分隔符的具体操作，请参见设置自定义分隔符。

例如，分隔符设置为英文感叹号（!），则图片处理URL为 `http(s)//:BucketName.Endpoint/ObjectName!StyleName`

使用阿里云SDK

您可以使用阿里云SDK将多个图片处理参数封装在一个样式中，然后使用图片样式批量处理图片。以下仅列举常见SDK的使用图片样式处理图片的代码示例，关于其他SDK的使用图片样式处理图片的代码示例，请参见SDK简介。

Python

```
// yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
String endpoint = "yourEndpoint";
// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 填写Bucket名称。
String bucketName = "examplebucket";
// 填写Object完整路径。Object完整路径中不能包含Bucket名称。
String objectName = "exampleobject.jpg";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 使用自定义样式处理图片。
// yourCustomStyleName填写通过OSS管理控制台创建的图片样式名称。
String style = "style/yourCustomStyleName";
GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
request.setProcess(style);
// 将处理后的图片命名为example-new.jpg并保存到本地。
// 填写本地文件的完整路径，例如D:\\localpath\\example-new.jpg。如果指定的本地文件存在会覆盖，不存在则新建。
// 如果未指定本地路径只填写了文件名称（例如example-new.jpg），则文件默认保存到示例程序所属项目对应本地路径中。
ossClient.getObject(request, new File("D:\\localpath\\example-new.jpg"));
// 关闭OSSClient。
ossClient.shutdown();
```

将样式应用于其他Bucket

您可以通过样式的导出和导入功能，将某个Bucket中的样式快速应用于其他Bucket。

1. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket名称。
2. 在左侧导航栏，选择**数据处理 > 图片处理**。
3. 单击**导出样式**，在**另存为**对话框选择样式的保存位置，之后单击**保存**。
4. 在**图片处理**页签，单击**导入样式**。
5. 在**打开**对话框选择刚导出的样式文件，然后单击**打开**。

   样式导入完成后，即可在新的Bucket中使用这些样式处理图片文件。

10.2.5. 图片处理持久化

对象存储OSS的图片处理服务默认不保存处理后的图片，您需要在图片处理的请求内添加转存参数，将处理后的图片作为文件（Object）保存至指定的存储空间（Bucket）内。

注意事项

- 权限要求

  进行图片转存操作要求具有源Bucket的 `oss:PostProcessTask` 权限，以及目标Bucket的 `oss:PutBucket` 和目标Object的 `oss:PutObject` 权限。
- 存储地域

  原图所在Bucket和处理后图片转存的目标Bucket可以相同也可以不同，但必须属于同一账号下的相同地域。
- 转存方式
  - 支持通过添加转存参数的方式将阿里云SDK处理后的图片保存至指定Bucket。具体操作，请参见阿里云SDK图片处理持久化示例。
  - 不支持将文件URL处理后的图片直接转存至指定Bucket。您可以将处理后的图片保存到本地，然后再上传至指定Bucket。
- 转存图片读写权限ACL

  转存图片的读写权限ACL默认继承Bucket，不支持自定义。
- 转存图片存储时长

  如果您需要调整转存图片的存储时长，请结合生命周期规则配置合理的文件过期策略。

使用阿里云SDK

以下仅列举常见SDK的图片处理持久化的代码示例。关于其他SDK的图片处理持久化的代码示例，请参见SDK简介。

Python

```
import com.aliyun.oss.*;
import com.aliyun.oss.common.utils.BinaryUtil;
import com.aliyun.oss.common.utils.IOUtils;
import com.aliyun.oss.model.GenericResult;
import com.aliyun.oss.model.ProcessObjectRequest;
import java.util.Formatter;
public class Demo {
    public static void main(String[] args) throws Throwable {
        // Endpoint以华东1（杭州）为例，其它Region请按实际情况填写。
        String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
        String accessKeyId = "yourAccessKeyId";
        String accessKeySecret = "yourAccessKeySecret";
        // 填写Bucket名称，例如examplebucket。
        String bucketName = "examplebucket";
        // 填写Object完整路径。Object完整路径中不能包含Bucket名称。
        String sourceImage = "exampleimage.png";
        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // 将图片缩放为固定宽高100 px。
            StringBuilder sbStyle = new StringBuilder();
            Formatter styleFormatter = new Formatter(sbStyle);
            String styleType = "image/resize,m_fixed,w_100,h_100";
            // 将处理后的图片命名为example-resize.png并保存到当前Bucket。
            // 填写Object完整路径。Object完整路径中不能包含Bucket名称。
            String targetImage = "example-resize.png";
            styleFormatter.format("%s|sys/saveas,o_%s,b_%s", styleType,
                    BinaryUtil.toBase64String(targetImage.getBytes()),
                    BinaryUtil.toBase64String(bucketName.getBytes()));
            System.out.println(sbStyle.toString());
            ProcessObjectRequest request = new ProcessObjectRequest(bucketName, sourceImage, sbStyle.toString());
            GenericResult processResult = ossClient.processObject(request);
            String json = IOUtils.readStreamAsString(processResult.getResponse().getContent(), "UTF-8");
            processResult.getResponse().getContent().close();
            System.out.println(json);
        } catch (OSSException oe) {
            System.out.println("Caught an OSSException, which means your request made it to OSS, "
                    + "but was rejected with an error response for some reason.");
            System.out.println("Error Message:" + oe.getErrorMessage());
            System.out.println("Error Code:" + oe.getErrorCode());
            System.out.println("Request ID:" + oe.getRequestId());
            System.out.println("Host ID:" + oe.getHostId());
        } catch (ClientException ce) {
            System.out.println("Caught an ClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with OSS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message:" + ce.getMessage());
        } finally {
            if (ossClient != null) {
                ossClient.shutdown();
            }
        }
    }
}
```

使用REST API

如果您的程序自定义要求较高，您可以直接发起REST API请求。直接发起REST API请求需要手动编写代码计算签名。

您可以通过PostObject接口调用图片处理服务时，并通过Body的方式传递x-oss-process，然后在图片处理请求中增加saveas参数将处理后的图片保存至指定Bucket。更多信息，请参见PostObject。

使用saveas参数时，您需要携带以下选项：

选项	含义
o	目标Object名称，名称需经过URL Safe的Base64编码。具体操作，请参见水印编码。
b	目标Bucket名称，名称需经过URL Safe的Base64编码。如果不指定目标Bucket，则默认保存至原图所在Bucket。

您可以通过以下两种方式处理图片并将图片转存至指定Bucket。

- 使用图片处理参数处理图片并转存至指定Bucket，示例如下：

```
POST /ObjectName?x-oss-process HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:KU5h8YMUC78M30dXqf3JxrT****=
// 将目标图片test.jpg等比缩放为宽100 px后，保存到名为test的Bucket中。
x-oss-process=image/resize,w_100|sys/saveas,o_dGVzdC5qcGc,b_dGVzdA
```

- 使用样式处理图片并转存至指定Bucket，示例如下：

```
POST /ObjectName?x-oss-process HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 247
Date: Fri, 04 May 2012 03:22:13 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:KU5h8YMUC78M30dXqf3JxrT****=
// 使用名为examplestyle的样式处理目标图片test.jpg后，保存到名为test的Bucket中。
x-oss-process=style/examplestyle|sys/saveas,o_dGVzdC5qcGc,b_dGVzdA
```

10.2.6. 错误响应

当用户访问图片处理服务出现错误的时候，图片处理服务会返回给用户相应的错误码和错误信息，以帮助用户定位与处理问题。

错误响应

图片处理服务错误响应的消息体示例如下：

```
<Error>
  <Code>BadRequest</Code>
  <Message>Input is not base64 decoding.</Message>
  <RequestId>52B155D2D8BD99A15D0005FF</RequestId>
  <HostId>userdomain</HostId>
</Error>
```

错误消息包含以下元素：

- `Code` ： 图片处理服务返回给用户的错误码。
- `Message` ： 图片处理服务给出的详细错误信息。
- `RequestId` ： 标识错误请求的唯一UUID，在无法解决问题时候，可以使用此错误ID发送给图片处理服务的工程师去定位错误的原因。
- `HostId` ：标识访问的图片处理服务集群。

错误码

图片处理服务包含的错误码如下：

错误码	描述	解决方案
InvalidArgument	参数错误	400错误
BadRequest	错误请求	
MissingArgument	缺少参数	
ImageTooLarge	图片大小超过限制	
WatermarkError	水印错误	
NotImplemented	方法未实现	
AccessDenied	拒绝访问	403错误
SignatureDoesNotMatch	签名不匹配	
NoSuchKey	图片不存在	404错误
NoSuchStyle	样式不存在	
InternalError	服务内部错误	500错误

SDK示例

- Java
- Python
- PHP
- Go
- C++
- C
- .NET
- Node.js
- Browser.js
- Android

10.2.7. 图片处理常见问题

本文主要介绍您在使用OSS图片处理时可能遇到的一些常见问题及处理方法。

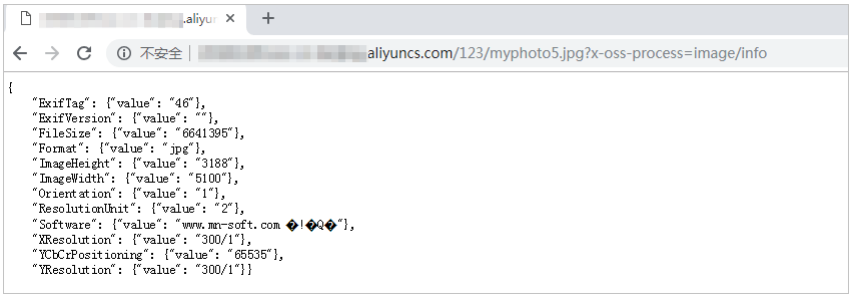
遇到问题时，如果有明显的参数超过显示等问题，可以使用 OSS 的 `?x-oss-process=image/info` 参数查看原始图片中的信息是否超标。OSS 单边长度不能超过 4096，乘积不能高于 4096*4096 。

案例：旋转图片时出现 Picture exceed the maximum allowable rotation range 报错

问题分析：出现这种问题基本都是原图的单边长度超过了 4096 的限制，或者四边乘积超过了 4096*4096。

排查步骤：

1. 使用 OSS 的 `?x-oss-process=image/info` 参数获取图片的信息判断是否超过限制。



2. 查看 `ImageWidth` 值是 5100，超过 4096 的限制。

处理方案：使用 `auto-orient,0` 参数关闭自适应，再使用 `resize` 参数调整图片大小，最后旋转图片。例如：*http://test.oss-cn-beijing.aliyuncs.com/123/myphoto5.jpg?x-oss-process=image/auto-orient,0/resize,m_lfit,h_2000,w_2000,limit_1/rotate,90*

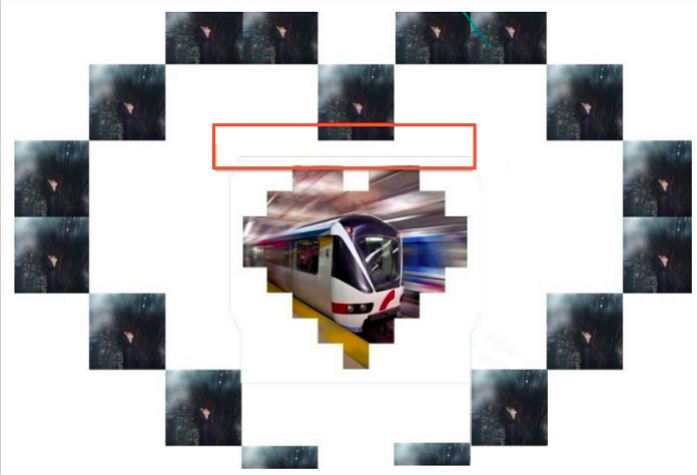
案例：通过 OSS 获取主色调和图片不符

问题分析：主色调计算不是按照屏幕颜色占比来计算的，而是按照图片中心的主颜色来定的色调，计算逻辑如下：

1. 计算整个图片的色调的平均值 (avg_hue)。
2. 遍历每个像素，计算该像素的色调值与 avg_hue 的色差（即将二者相减后取绝对值），如果该色差大于一个阈值，则将该像素加入到“醒目像素”的列表。
3. 计算整个“醒目像素列表”的颜色均值，得到的结果即为该图片的主色调。

处理方案：可以使用 `?x-oss-process=image/average-hue` 参数获取 OSS 图片的主色调参数。

案例：图片水印合成出现黑线



问题分析：这个黑印不是因为图片处理造成的。水印的方式是将两张图片重合，如果水印两张图是不同的 RGB 图片，覆盖后因色差产生黑线是正常现象，任何图片处理工具都会存在这个问题。

使用 `?x-oss-process=image/average-hue` 参数可查询图片 RGB 的参数，可在图片链接后加这个参数判断两个图片的 RGB 参数是否一致。

处理方案：案例中背景图 RGB 参数为 “0x0e0e0e”，水印的 RGB 参数为 “0xffffff”，增加水印会出现类似边框的效果。可以通过透明度参数 `t` 来调整透明度将边框去掉，t 的取值范围是 1-100。例如：*http://image-demo.img-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_300/watermark,image_cGFuZGEucG5nP3gtb3NzLXByb2Nlc3M9aW1hZ2UvcmVzaXplLFBfMzA,t_90,g_se,x_10,y_10,t_50*

案例：通过 CDN 回源 OSS 图片处理不生效

CDN 回源 OSS 图片处理不生效，无论使用的是什么效果，请使用 OSS 的访问域名进行测试，利用下面的 URL 进行基础分析。

http://test.oss-cn-beijing.aliyuncs.com/MomClass/ChuXin/3_2_336_462.jpg@30-30bl

http://test.img-cn-beijing.aliyuncs.com/MomClass/ChuXin/3_2_336_462.jpg@30-30bl

- *img-cn-region.aliyuncs.com*是老版本的 OSS 域名，图片处理的分隔符和图片处理语法和新版的 oss 域名都不一样。
- *oss-cn-region.aliyuncs.com*这类域名是 2017 年以后使用的新域名，不兼容 img 域名的图片处理语法和分隔符 "@"，需要在 OSS 控制台上手动执行同步，将 img 域名图片处理同步到 OSS 。

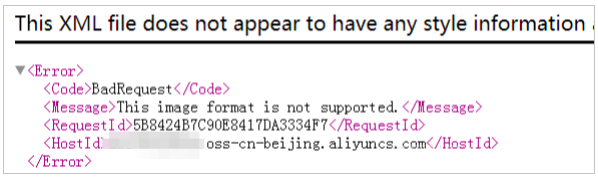
上述的老域名的高斯处理效果，如果搬迁到 OSS 的域名后，需要按照新的方式来处理，如下：

http://test.oss-cn-beijing.aliyuncs.com/MomClass/ChuXin/3_2_336_462.jpg?x-oss-process=image/blur,r_3,s_30

案例：图片缩略后颜色变亮了

处理分析：可以使用 PS 等工具获取原图的颜色模式，如果原图是 RGB 的话，压缩是不会变色的，如果原图是 CMYK 的话，压缩后会产生偏色。目前对 CMYK 的兼容还在支持中，图片色彩空间被挤压会产生色彩的变化。

案例：图片在本地可以正常打开，进行图片处理时提示已损坏



问题现象：图片文件在本地可以正常打开，但是上传到到 OSS 无法进行图片处理，反馈图片损坏。

排查步骤：

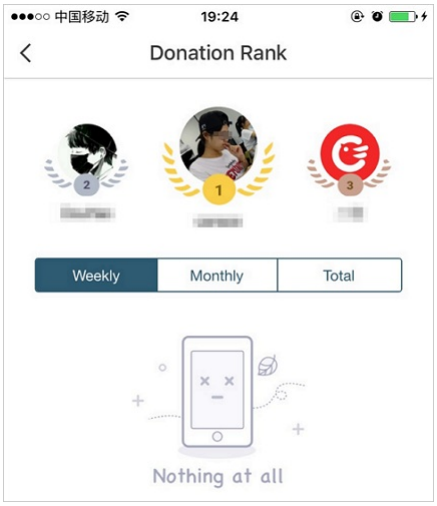
1. 获取原始的 OSS URL 地址，使用 `?x-oss-process=image/info` 查看原图信息，如果查不到图片信息，直接报错，说明原图就是损坏的。
2. 可以使用开源的 imagemagic 工具来验证这个问题，将图片做任意调整，如果出现 error 说明图片是损坏的。下面是一个 resize 的测试用例：

```
convert -resize 1024x768 1123331261_15353307414801n.jpg
```

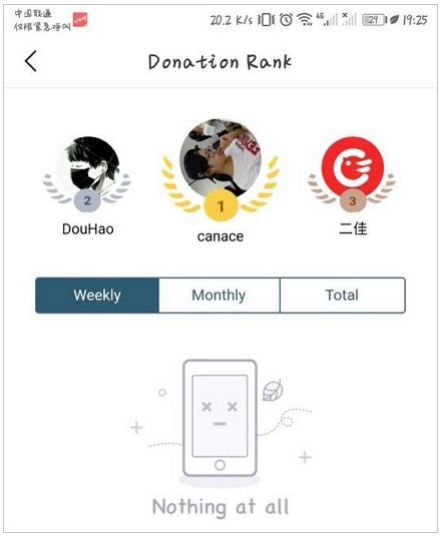
损坏的图片文件在本地可以显示是因为本地的图片查看工具是对图片做了补偿修复。而 OSS 不对损坏的图片进行处理，所以在浏览器上无法显示。

案例：存储在 OSS 内的图片旋转了 90 度

问题现象：通过 OSS 域名访问图片正常。



通过 CDN 访问，发现图片被旋转了 90 度。



问题分析：直接访问 OSS 正常，说明 OSS 存储是没问题的。但是通过 CDN 访问时出现了旋转，说明是浏览器的处理问题，通过图片处理参数 `?x-oss-process=image/info` 查看原图参数带有 `rotation 90` 旋转参数。

处理方案：删除旋转参数即可。

案例：经过 CDN 加速后图片处理没有效果

排查步骤：检查是否开启了 CDN 的**过滤参数**功能，若开启性能参数功能，回源时会去除 URL 中问号（?）之后的参数。详情请参见 CDN 过滤参数。

案例：使用图片处理出现“Picture exceed the maximum allowable rotation range”报错

排查方法：

- 可以使用 imagemagic 工具查看原图是否自带了 auto-orient 自适应旋转的属性。
- 使用 `auto-orient,0` 参数处理图片，可以正常处理就说明原图带了自适应旋转的属性的。

带了自适应旋转的属性后，要求图片的宽高不能超过 4096。

案例：苹果手机端携带了图片处理参数访问经过 CDN 加速的图片时变成空白图片，刷新后可以访问，电脑访问正常

问题分析：电脑端访问正常，手机端访问异常，可以判断出 OSS 是正常的，否则电脑访问也会异常。

排查步骤：

1. 使用手机直接访问 OSS 查看图片访问是否正常。
   - 若直接访问 OSS 正常，而通过 CDN 访问异常，说明 CDN 节点网络问题导致加载失败或 CDN 缓存了错误内容。
   - 若直接访问 OSS 也异常，那么 CDN 访问应该也是异常的，若 CDN 刷新一下就正常，可能是因为 CDN 的缓存导致。
2. 使用电脑端直接访问 OSS 查看图片访问是否正常。
   - 若电脑端访问正常，则问题出在手机端。
   - 若电脑端访问也异常，则可能是图片出问题了。

本案例中，因图片是 webp 格式，苹果手机不支持该格式。

案例：存储在 OSS 的原图和经过图片处理后的图片都打不开

排查步骤：

1. 下载图片。

```
[root@edas02 aliyun-oss-php-sdk]# wget https://zh.mobi/test/123.jpg
--2018-11-2210:55:16--  https://zh.mobi/test/123.jpg
正在解析主机 zh.mobi (zh.mobi)...
已发出 HTTP 请求，正在等待回应... 200 OK
长度：4141232 (3.9M) [image/jpeg]
正在保存至：“123.jpg”
100%[===========================================================================>] 4,141,232 12.5MB/s 用时 0.3s
2018-11-2210:55:16 (12.5 MB/s) - 已保存 “123.jpg” [4141232/4141232])
```

2. 用开源工具 imagemagick 查看图片的编码构成是否有问题。

```
[root@edas02 aliyun-oss-php-sdk]# identify 123.jpg
identify: Not a JPEG file: starts with 0x000x00 `123.jpg' @ error/jpeg.c/JPEGErrorHandler/316.
[root@edas02 aliyun-oss-php-sdk]#
```

   检查发现图片编码构成有问题，并非是存储到 OSS 后出现的问题，类似问题都可以用这个工具分析。

案例：图片处理完后背景色多了一条分割线

问题分析：图片中出现的并非是分割线，而图片处理后色彩构成出现问题。原图是 RGB 的真彩色（ImageHeight": {"value": "2560" "ImageWidth": {"value": "1440" ）。经过图片处理后，像素被裁剪到 h_1920,w_1080，导致 RGB 的像素点位被压缩，图片显示异常。

解决方法：使用 `quality,Q_100` 参数将图片的绝对质量提高到 100 即可。

案例：图片处理出现“BadRequest”报错

```
<Error>
<Code>BadRequest</Code>
<Message>This image format is not supported.</Message>
<RequestId>5BA33754CBF4583BA2</RequestId>
<HostId>b.oss-cn-beijing.aliyuncs.com</HostId>
</Error>
```

排查步骤：

1. 使用 imagemagic 工具的 convert 命令看下原图的格式。
2. 确认 OSS 是否支持该图片格式。有关 OSS 支持图片格式的更多信息，请参见使用限制。

解决方案：将图片格式转换为 OSS 支持的格式。

案例：使用图片处理出现“InvalidArgument”报错

```
<Code>InvalidArgument</Code>
<Message>The value: 0 of parameter: w is invalid.</Message>
<RequestId>5BA21FD8A642F41E6478</RequestId>
<HostId>luo.oss-cn-beijing.aliyuncs.com</HostId>
</Error>
```

问题分析：遇到这种参数错误，需先查看一下原图的请求参数，类似 `20180899269957.jpg@0w_2e_1l_1an.src` 这种请求参数的，都是历史 `img-cn-xx` 域名支持的格式。转换成新的 `oss-cn-xxx` 域名后是不支持 img 域名的请求方式的，并且老域名不支持 https 访问方式。

OSS 能否识别请求的自定义 query 参数动态缩放

目前 OSS 还无法适配这种业务需求。

一个文字水印是否可以分两行显示？一个图片是否可以添加多个文字水印？

OSS 图片水印不支持将一个文字水印分行显示，但是一个图片可以添加多个文字水印。详情请参见图片水印。

10.2.8. 新旧版本图片处理服务及使用说明

图片处理服务目前提供新旧两版服务，本文介绍两版服务的主要区别。

新旧版本图片处理服务的主要区别

在添加处理参数时，新旧版本服务中的格式不同，区别如下：

- 新版参数格式：`http://bucket.<endpoint>/object?x-oss-process=image/action,parame_value`

  所有的图片处理操作都通过 `x-oss-process` 进行传递。每个action之间顺序执行。
- 旧版参数格式：`http://channel.<endpoint>/object@action.format`

  操作通过 `@` 作为分隔符进行处理。

通过OSS域名及通过IMG域名访问处理后图片的区别

使用旧版图片处理服务处理图片，会为图片生成专门的IMG域名。而使用新版服务处理图片，图片仍然使用OSS域名。通过两种域名访问处理后图片时的区别如下表所示。

对比项	采用IMG域名访问	直接使用OSS域名访问
使用方式	存储与处理两套域名系统	上传、管理、处理、分发，一站式处理

对比项	采用IMG域名访问	直接使用OSS域名访问
是否支持新版API	支持	支持
是否支持旧版API	支持	默认不支持
是否支持HTTPS	不支持	支持
是否支持VPC网络	不支持	支持
是否支持多域名绑定	不支持	支持
是否支持源站更新自动刷新阿里CDN	不支持	支持

说明

- OSS域名已全面支持图片处理服务，不过只能使用新版服务的API。而原有的IMG域名能够使用新旧两个版本的API。
- 如果IMG域名期望能够进行CDN加速，可以通过在CDN配置回源host的方式直接访问IMG域名，不需要进行域名绑定来完成CDN加速。

使用新旧版本图片处理功能Bucket的区别

开启过旧版图片处理服务的Bucket：

- 与旧版图片处理功能逻辑基本一致。用户看到的图片域名是使用旧版服务时生成的IMG域名，以及之前已经绑定的自定义域名。
- 通过旧版服务进行的原图保护等配置，只对之前生成的IMG域名有效，对于文件的OSS域名没有效果。当在跨区域复制中开启同步时，会将原图保护以及样式分隔符同步到OSS域名。
- 当用户关闭当前Bucket的图片处理服务时，会清空样式配置以及域名绑定，并自动跳转到新版的页面。

新创建的Bucket或者之前没有开启旧版图片处理服务的Bucket：

- 默认能够使用图片处理服务，无需开通。
- 无需绑定域名，域名绑定操作直接同Bucket本身的域名管理一致。

使用旧版图片处理服务的用户如何切换至新版图片处理服务

旧版图片处理服务的API暂时无法在新版图片处理服务中使用，如有特殊情况可以工单联系售后技术支持。但如果您在旧版图片处理服务中只通过样式访问图片，则可以通过以下步骤进行切换：

- 在当前图片服务配置里面开启配置同步，样式分隔符以及原图保护能够同步到新版图片处理服务。
- 如果使用了自定义域名，将原有的自定义域名CNAME改到OSS域名即可。

新旧版图片处理服务的样式配置是否一致

所有的样式配置在新旧版的图片处理服务中是共享的，旧版图片处理服务的样式配置在新版中可以正常使用。

10.3. 视频截帧

本文介绍视频截帧操作涉及的参数说明及使用示例。

注意事项

- 使用视频截帧时，按视频截帧截取的图片数量计费。有关计费详情的更多信息，请参见数据处理费用。
- 仅支持对视频编码格式为H264和H265的视频文件进行视频截帧。
- OSS默认不保存视频截帧的图片，视频截帧的图片需手动下载并保存至本地。

参数说明

操作分类：`video`

操作名称：snapshot

参数	描述	取值范围
t	指定截图时间。	[0,视频时长] 单位：ms
w	指定截图宽度，如果指定为0，则自动计算。	[0,视频宽度] 单位：像素（px）
h	指定截图高度，如果指定为0，则自动计算；如果w和h都为0，则输出为原视频宽高。	[0,视频高度] 单位：像素（px）
m	指定截图模式，不指定则为默认模式，根据时间精确截图。如果指定为fast，则截取该时间点之前的最近的一个关键帧。	枚举值：*fast*
f	指定输出图片的格式。	枚举值：*jpg*和*png*

参数	描述	取值范围
ar	指定是否根据视频信息自动旋转图片。	枚举值：*auto*、*h*和*w* 各枚举值说明如下： • *auto*：指定在截图生成之后根据视频信息进行自动旋转。 • *h*：指定在截图生成之后根据视频信息强制按高大于宽的模式旋转。 • *w*：指定在截图生成之后根据视频信息强制按宽大于高的模式旋转。

使用示例

- 使用fast模式截取视频7s处的内容，输出为JPG格式的图片，宽度为800，高度为600。

  处理后的URL为：`<原视频URL>?x-oss-process=video/snapshot,t_7000,f_jpg,w_800,h_600,m_fast`

- 使用精确时间模式截取视频50s处的内容，输出为JPG格式的图片，宽度为800，高度为600。

  处理后的URL为：`<原视频URL>?x-oss-process=video/snapshot,t_50000,f_jpg,w_800,h_600`

生成带签名的视频截帧URL

您可以通过SDK生成带签名的视频截帧URL，以Java SDK为例，代码如下：

```
// Endpoint以杭州为例，其它Region请按实际情况填写。
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。
String accessKeyId = "yourAccessKeyId";
String accessKeySecret = "yourAccessKeySecret";
// 填写视频文件所在的Bucket名称，例如examplebucket。
String bucketName = "examplebucket";
// 填写视频文件的完整路径。若视频文件不在Bucket根目录，需携带文件访问路径，例如examplefolder/videotest.mp4。
String objectName = "examplefolder/videotest.mp4";
// 创建OSSClient实例。
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// 使用精确时间模式截取视频50s处的内容，输出为JPG格式的图片，宽度为800，高度为600。
String style = "video/snapshot,t_50000,f_jpg,w_800,h_600";
// 指定过期时间为10分钟。
Date expiration = new Date(new Date().getTime() + 1000 * 60 * 10 );
GeneratePresignedUrlRequest req = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.GET);
req.setExpiration(expiration);
req.setProcess(style);
URL signedUrl = ossClient.generatePresignedUrl(req);
System.out.println(signedUrl);
// 关闭OSSClient。
ossClient.shutdown();
```

生成带签名的视频截帧URL与生成带签名的图片处理URL的方法类似。如果您需要通过不同语言SDK生成带签名的视频截帧URL，请将如下不同语言SDK的图片处理操作替换为视频截帧操作。

- Python SDK
- PHP SDK
- Go SDK
- C SDK
- C++ SDK
- .NET SDK

- Android SDK
- iOS SDK
- Node.js SDK
- browser.js SDK

10.4. 数据索引（Data Indexing）

数据索引是OSS对外提供的文件（Object）元数据索引能力。您可以利用Object的元数据自定义索引的条件以快速获取Object列表，帮助您更好地管理与了解数据结构，方便您后续查询、统计和管理Object。

使用场景

基于数据审计或者数据监管等原因，您可能需要从存放于OSS存储空间（Bucket）内多达上亿的海量Object中查找符合特定条件的Object。Object本身包含大量的元数据，例如Object名称、Object ETag、Object存储类型、Object大小、Object标签、Object最后修改时间等。通过元数据索引功能，您可以在查找目标Object时结合具体的业务场景，通过组合简单查询条件以及聚合操作，提升查找目标Object的效率。

注意事项

- 支持地域

  仅澳大利亚（悉尼）地域支持使用数据索引功能。

- 费用说明

  开启元数据管理会产生一定的费用，但公测期间暂不收费。关于数据索引计费项的更多信息，请参见数据索引费用。

- Object数量

  当前仅允许对存储的Object个数等于或者小于3000万的Bucket开启元数据管理。如果您希望对Object个数大于3000万的Bucket开启元数据管理，请联系技术支持。

- 分片上传

  对于通过分片上传生成的Object，则查询结果中只显示已通过CompleteMultipartUpload操作将碎片（Part）合成的完整Object，不显示已初始化但未完成（Complete）或者未中止（Abort）的碎片。

使用OSS控制台

1. 登录OSS管理控制台。
2. 单击**Bucket列表**，然后单击目标Bucket名称。
3. 在左侧导航栏，选择**数据处理与索引 > 数据索引**。
4. 在**元数据管理**区域，开启元数据管理。

   开启元数据管理需要等待一定的时间，具体等待时长取决于Bucket中Object的数量。
5. 设置Object基础过滤条件。

   在**Object基础过滤条件**区域，按需设置以下基础过滤条件。

过滤条件	说明
存储类型	默认选中OSS支持的四种存储类型，即标准存储、低频访问存储、归档存储以及冷归档存储。您可以按需选择希望在查询结果中显示的Object存储类型。
读写权限	默认选中OSS支持的四种读写权限ACL，即继承Bucket、私有、公共读以及公共读写。您可以按需选择希望在查询结果中显示的Object读写权限。
文件名	支持**模糊匹配**和**等于**。如果您希望在查询结果中显示某个文件名，例如exampleobject.txt。您可以通过以下两种方式匹配目标文件： ○ 选择**等于**，然后输入完整的文件名称*exampleobject.txt*。 ○ 选择**模糊匹配**，然后输入文件前缀或者后缀，例如*example*或者*.txt*。 注意 模糊匹配可命中Object名称的任意字符，例如输入*test*，则查询结果中将显示*localfolder/test/.example.jpg*、*localfolder/test.jpg*等。
上传类型	默认选中OSS支持的四种Object类型，您可以按需选择希望在查询结果中显示的Object类型。Object类型说明如下： ○ **Normal**：通过简单上传方式生成的Object。 ○ **Multipart**：通过分片上传方式生成的Object。 ○ **Appendable**：通过追加上传方式生成的Object。 ○ **Symlink**：为快速访问Object创建的软链接。
最后修改时间	指定Object被最后修改的**起始日期**和**结束日期**，时间精确到秒。
文件大小	支持**等于**、**大于**、**大于等于**、**小于**和**小于等于**五种筛选条件，文件大小单位为KB。
对象版本	仅支持查询当前版本Object。

6. （可选）设置Object其他过滤条件。

   如果您需要对查询结果中的Object进行排序或者使用标签过滤等，请单击**展开更多过滤条件**。
   - 设置Object排序方式。

     在**对象排序方式**区域，结合**最后修改时间**、**文件名**和**文件大小**的筛选条件，选择查询结果中Object按照这三种筛选条件进行**升序**或**降序**排列。
   - 设置Object标签过滤条件。

     在**对象标签过滤**区域，输入您希望在查询结果中显示的Object对应的ETag或标签信息。
     - ETag仅支持精确匹配。可输入多个ETag，每行一个。

- 以键值对（Key-Value）的形式指定**对象标签**。对象标签的Key和Value均区分大小写。关于标签规则的更多信息，请参见对象标签。

- 设置Object数据聚合方式。

  如果您希望在查询结果中对数据进行分类统计，例如统计所有文件大小、去重统计文件存储类型等，请添加数据聚合方式。

11.监控服务

11.1. 监控服务概览

OSS监控服务为您提供系统基本运行状态、性能以及计量等方面的监控数据指标，并且提供自定义报警服务，帮助您跟踪请求、分析使用情况、统计业务趋势，及时发现以及诊断系统的相关问题。

OSS监控指标主要分为基础服务指标、性能指标和计量指标，详见OSS监控指标参考。

高实时性

高实时性能够暴露可能隐藏的峰谷问题，显示出实际的波动情况，有助于分析和评估业务场景。OSS监控指标的实时性（除了计量指标）是按照分钟粒度采集聚合的，输出延时不超过1分钟，即每分钟内的用户信息都会聚合成一个值，并在一分钟内输出，代表这一分钟的监控情况。

计量指标相关说明

为了保持和计费策略的统一，计量指标的收集和展现存在一定的特殊性，说明如下：

- 计量指标数据是按照小时粒度输出的，即每个小时内的资源计量信息都会聚合成一个值，代表这个小时总的计量情况。
- 计量指标数据会有近半个小时的延时输出。
- 计量指标数据的数据时间是指该数据所统计时间区间的开始时间。
- 计量采集截止时间是当月最后一条计量数据所统计时间区间的结束时间，如果当月没有产生任何一条计量监控数据，那么计量数据采集截止时间为当月1号0点。
- 计量指标数据的展示都是尽最大可能推送的，准确计量请参考费用中心—使用记录。

举个例子，假设用户只使用PutObject这个请求上传数据，每分钟平均10次。那么在2018-05-10 08:00:00到2018-05-10 09:00:00这一个小时时间区间内，用户的PUT类请求数的计量数据值为600次（10*60分钟），数据时间为2018-05-10 08:00:00，这条数据将会在2018-05-10 09:30:00左右被输出。如果这条数据是从2018-05-01 00:00:00开始到现在的最后一条计量监控数据，那么当月的计量数据采集截止时间就是2018-05-10 09:00:00。如果2018年5月该用户没有产生任何的计量数据，那么计量采集截止时间为2018-05-01 00:00:00。

OSS报警服务

每个账号最多能够设置1000项报警规则。除计量指标和统计指标，其他的监控指标均可配置为报警规则加入报警监控，并且一个监控指标可以配置为多个不同的报警规则。

- 报警服务相关概念请参见报警服务概览。
- OSS报警服务使用指南请参见使用报警服务。
- OSS具体监控指标请参见监控指标参考。

监控数据保留策略

监控数据保留31天，过期自动清除，如果需要离线分析监控数据或者长期下载并保存历史监控，需要使用工具或者编写代码来读取云监控数据存储，请参见OpenAPI访问监控数据。

控制台展示最近7天的数据，如果希望查询7天以上的历史数据，建议使用云监控提供的SDK进行查询，请参见OpenAPI访问监控数据。

OpenAPI访问监控数据

OSS服务的相关监控指标数据可以通过云监控提供的OpenAPI访问，使用方法请参见：

- 云监控SDK参考
- 访问监控数据

监控、诊断和故障排除

监控诊断和故障排除通过详细介绍以下各个方面的内容帮助您更好的了解OSS服务的运行状态并进行自主诊断和故障排除：

- 服务监控

  介绍如何使用监控服务持续监控OSS存储服务的运行状况和性能。
- 跟踪诊断

  介绍如何使用OSS监控服务和logging记录功能诊断问题，以及如何关联各种日志文件中的相关信息进行跟踪诊断。
- 故障排除

  提供常见的问题场景和故障排除方法。

注意事项

OSS Bucket全局唯一，如果删掉Bucket之后再创建同名的Bucket，那么被删掉的Bucket的监控以及报警规则会作用在新的同名Bucket上。

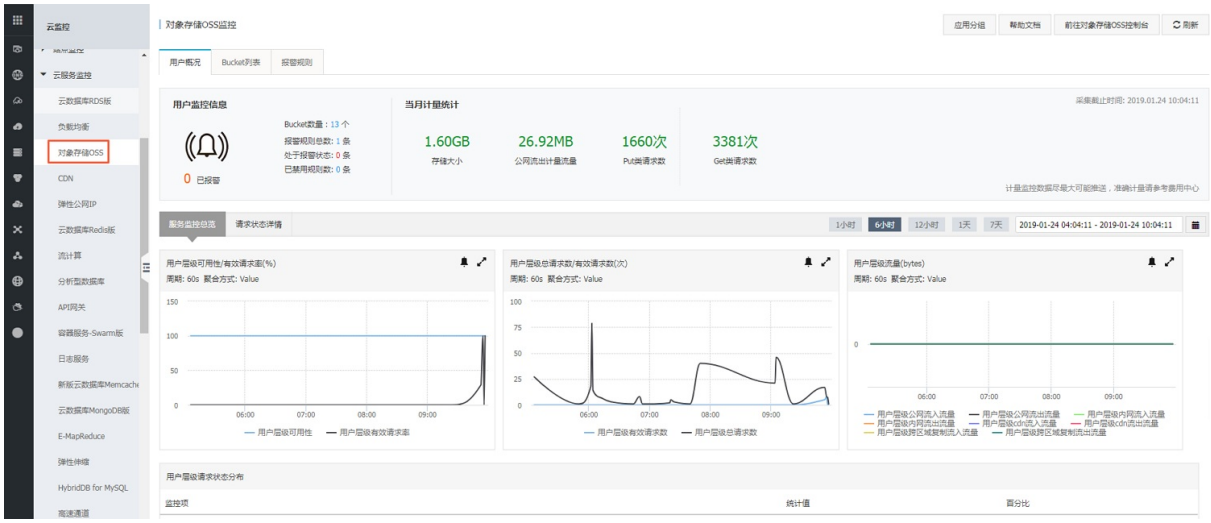
11.2. 使用监控服务

本文介绍如何使用对象存储OSS的监控服务。

OSS监控服务入口

OSS监控服务处于云监控控制台中。可以通过如下两种方式进入。

- 登录OSS管理控制台，在OSS概览页右边单击**设置报警规则**，进入OSS监控服务。
- 直接进入云监控控制台查看OSS监控服务。



OSS监控服务页面

OSS监控服务主页的主体由如下三部分组成。

- 用户概况
- Bucket列表
- 报警规则

该页面没有自动刷新功能，可以单击右上角的**刷新**按钮自动更新数据信息。

单击**前往对象存储OSS控制台**可以直接进入OSS控制台界面。

用户概况

用户概况页面从用户层级监控用户相关的信息。主要包括用户监控信息、当月计量统计和用户层级监控指标三大部分。

- 用户监控信息

  该模块主要展示该账号所拥有的Bucket总数以及相关的报警规则情况。

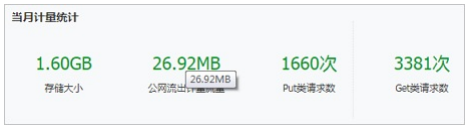
  - 单击**Bucket数量**的数字，链接到Bucket列表Tab页。
    - 单击**报警规则总数**的数字，链接到报警规则Tab页。
    - 单击**处于报警状态**的数字，链接到报警规则Tab页，并且此时该页展示的报警规则均处于告警状态。
    - 单击**已禁用规则数**的数字，链接到报警规则Tab页，并且此时该页展示的报警规则均被禁用。
    - 单击警铃图标下面的数字，链接到报警规则Tab页，并且此时该页展示的报警规则均处于告警状态。

- 当月计量统计

  当月计量统计展示了从当月1号0点开始，到采集截止时间为止，这段时间内所使用的OSS服务的计费相关的资源信息，包括如下指标：

  - 存储大小
  - 公网流出流量
  - Put类请求数
  - Get类请求数

各个计量框中展示的数据根据量级自动调整单位，鼠标停留在数字上方会显示精确的数值。

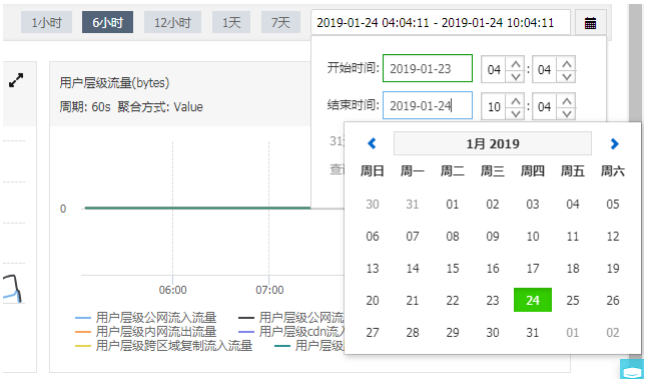


- 用户层级监控指标

  该模块主要展示具体的用户层级的监控图表，主要包括服务监控总览和请求状态详情两部分，下面会详细介绍。

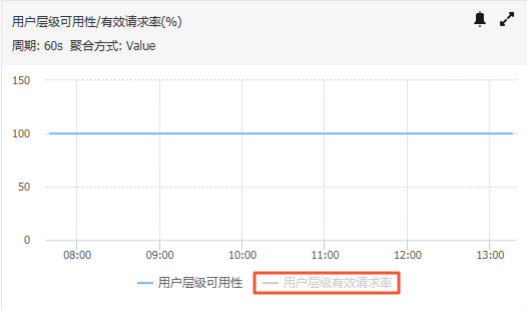
  图表展现提供了快速时间范围选择按钮和自定义时间框。

  - 快速时间范围选择按钮提供1小时、6小时、12小时、1天和7天的时间范围选择，默认为1小时。
  - 自定义时间框可以自定义起始时间和结束时间，精确到分钟级别。注意，不支持查询8天以前的数据。



  图表展示还支持以下功能：

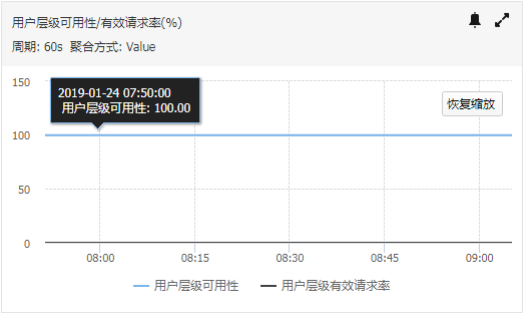
  - 单击相关图例可以将该指标曲线隐去，如下图：



  - 单击图形右上图标

    可以将图形放大展示。注意，表格不支持放大展示。
  - 单击图形右上图标

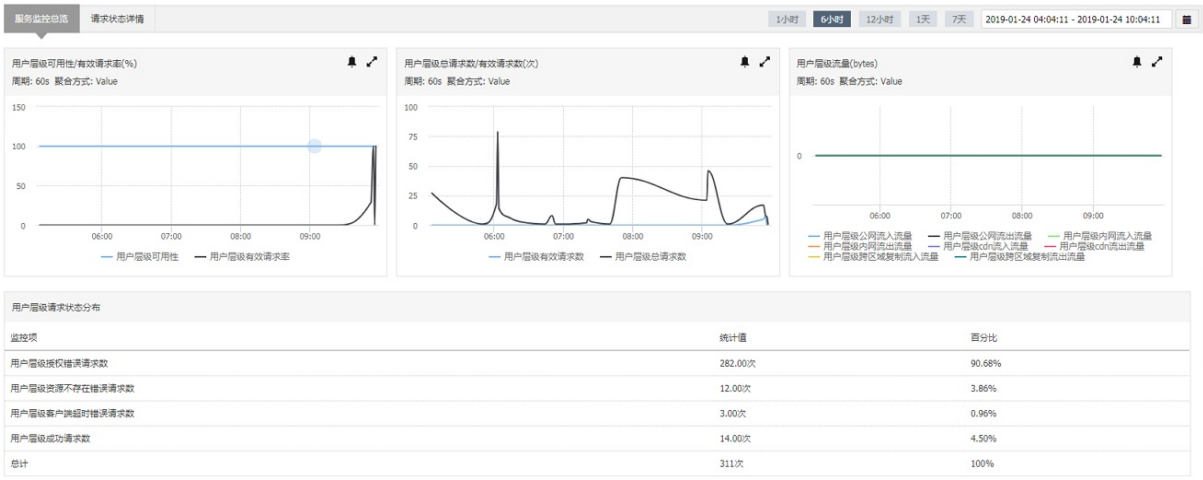
    可以对该图中展示的指标项设置相关报警规则。详见报警服务使用指南。注意，表格和计量参考指标不支持报警设置。
  - 鼠标按住图形曲线区域拖放，可以进行时间范围快速调整放大，单击**恢复缩放**回归到拖放之前的时间范围。



- 服务监控总览

  服务监控总览页面主要包括下面监控指标图：

  - 用户层级可用性/有效请求率： 包括可用性和有效请求率两项指标。
  - 用户层级总请求数/有效请求数： 包括总请求数和有效请求数两项指标。
  - 用户层级流量： 包括公网流出流量、公网流入流量、内网流出流量、内网流入流量、CDN流出流量、CDN流入流量、跨区域复制流出流量和跨区域复制流入流量八项指标。
  - 用户层级请求状态分布： 该表格中展示选定时间范围内各个请求类型的个数以及占比。



监控项	统计值	百分比
用户层级授权错误请求数	282.00次	90.68%
用户层级资源不存在错误请求数	12.00次	3.86%
用户层级客户端超时错误请求数	3.00次	0.96%
用户层级成功请求数	14.00次	4.50%
总计	311次	100%

- 请求状态详情

  请求状态详情是对请求状态分布统计的一个具体监控，主要包括下面的监控指标图：

  - 用户层级服务端错误请求数。
  - 用户层级服务端错误请求占比。
  - 用户层级网络错误请求数。
  - 用户层级网络错误请求占比。
  - 用户层级客户端错误请求数：包括资源不存在错误请求数、授权错误请求数、客户端超时错误请求数和客户端其他错误请求数四项指标。
  - 用户层级客户端错误请求占比：包括资源不存在错误请求占比、授权错误请求占比、客户端超时错误请求占比和客户端其他错误请求占比四项指标。
  - 用户层级有效请求数： 包括成功请求数和重定向请求数两项指标。
  - 用户层级有效请求占比： 包括成功请求占比和重定向请求占比两项指标。

Bucket列表

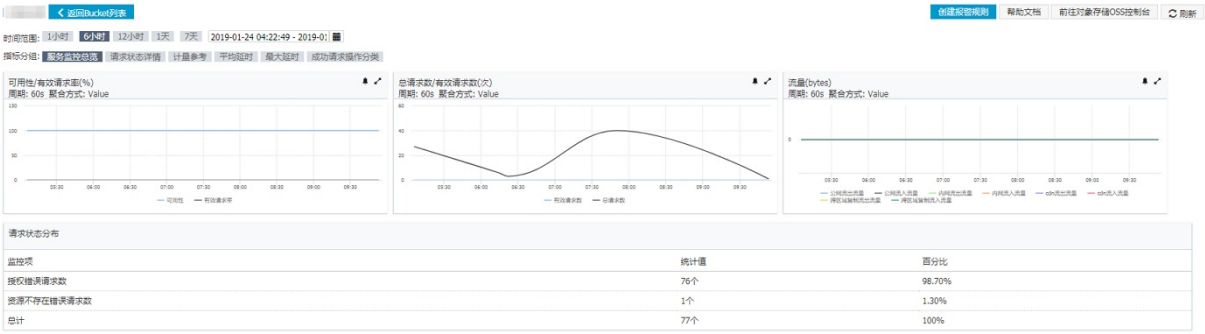
- Bucket列表信息

列表展现该账号所拥有的Bucket的名称、所属地域、创建时间、当月计量数据统计信息以及相关操作。

- 当月计量统计包括每个Bucket的存储量、公网流出流量、Put类请求数和Get类请求数。
- 单击监控图表或者对应的Bucket名称，能够进入具体的Bucket监控视图页。
- 单击报警规则，进入报警规则Tab页，并且展现所有属于该Bucket的报警规则。
- 通过上面的搜索框能够模糊匹配快速找到具体的Bucket。
- 选中Bucket复选框，并单击设置报警规则可以批量设置报警规则，详见报警服务使用指南。

- Bucket层级监控视图

  单击Bucket列表中具体的Bucket行中的**监控图表**，就能进入对应的Bucket的监控视图页，如下图：



  Bucket监控视图页按指标分组进行展示监控图，主要包含六个指标分组：

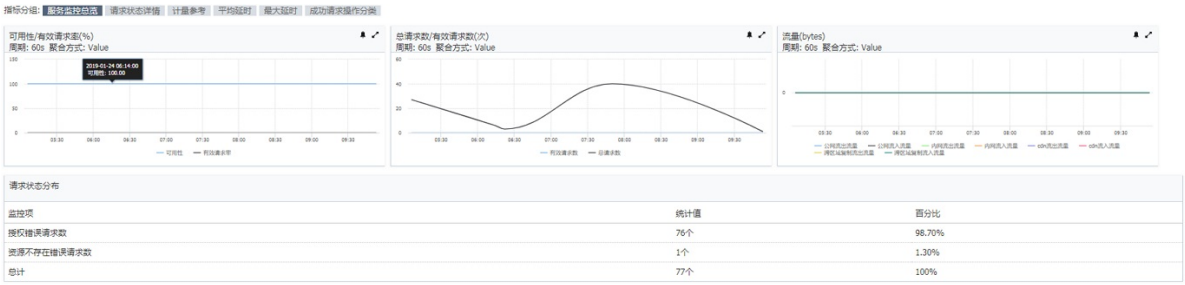
  - 服务监控总览
  - 请求状态详情
  - 计量参考
  - 平均延时
  - 最大延时
  - 成功请求操作分类

  除了计量参考，所有的指标项都是分钟级别聚合展示的。不同于用户层级默认时间展现为最近1小时，Bucket层级的监控展示默认为6小时。单击上方的**返回Bucket列表**能够回到Bucket列表Tab页。

  - 服务监控总览

    该指标分组同用户层级的服务监控总览，只是从具体的Bucket进行监控，主要包括下面监控指标图：

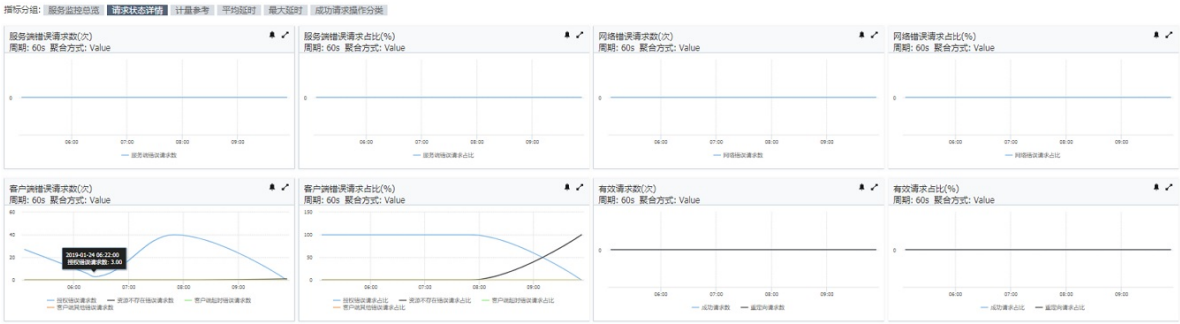
    - 可用性/有效请求率： 包括可用性和有效请求率两项指标。
    - 总请求数/有效请求数： 包括总请求数和有效请求数两项指标。
    - 流量： 包括公网流出流量、公网流入流量、内网流出流量、内网流入流量、cdn流出流量、cdn流入流量、跨区域复制流出流量和跨区域复制流入流量八项指标。
    - 请求状态分布： 该表格中展示选定时间范围内各个请求类型的个数以及占比。



- 请求状态详情

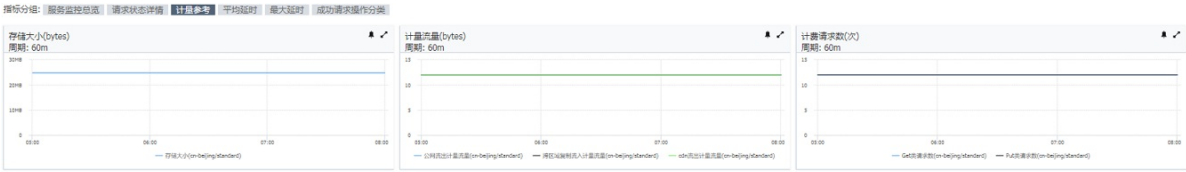
  该指标分组同用户层级的请求状态详情，只是从具体的Bucket进行监控，主要包括下面监控指标图：

  - 服务端错误请求数
  - 服务端错误请求占比
  - 网络错误请求数
  - 网络错误请求占比
  - 客户端错误请求数： 包括资源不存在错误请求数、授权错误请求数、客户端超时错误请求数和客户端其他错误请求数四项指标。
  - 客户端错误请求占比： 包括资源不存在错误请求占比、授权错误请求占比、客户端超时错误请求占比和客户端其他错误请求占比四项指标。
  - 有效请求数： 包括成功请求数和重定向请求数两项指标。
  - 有效请求占比： 包括成功请求占比和重定向请求占比两项指标。



- 计量参考

  计量参考分组展示各个计量相关的指标信息，以小时粒度收集展现，如下图所示：



  包含以下计量指标监控图：

  - 存储大小
  - 公网流出流量
  - 计费请求数：包括Get类请求数和Put类请求数两项指标项。

  如果新建Bucket，需要到当前时间点的下一个整小时点才会采集到新数据，然后在半个小时内展示出来。

- 平均延时

  该分组包含分API类型监控的各项平均延时指标，包含如下几个指标图：

  - Get Object请求平均延时
  - Head Object请求平均延时
  - Put Object请求平均延时
  - Post Object请求平均延时
  - Append Object请求平均延时
  - Upload Part请求平均延时
  - Upload Part Copy请求平均延时

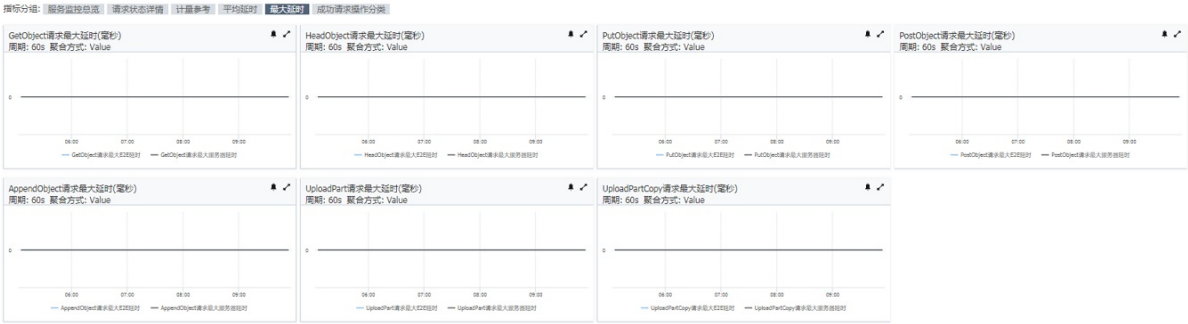
  每个指标图中都包含对应的平均E2E延时和平均服务器延时，如下图所示：

- 最大延时

  该分组包含分API类型监控的各项最大延时指标，包含如下几个指标图：

  - GetObject请求最大延时
  - HeadObject请求最大延时
  - PutObject请求最大延时
  - PostObject请求最大延时
  - AppendObject请求最大延时
  - UploadPart请求最大延时
  - UploadPartCopy请求最大延时

  每个指标图中都包含对应的最大E2E延时和最大服务器延时，如下图所示：



- 成功请求操作分类

  该分组包含分API类型监控的各项成功请求数指标，包含如下几个指标图：

  - GetObject成功请求
  - HeadObject成功请求
  - PutObject成功请求
  - PostObject成功请求
  - AppendObject成功请求
  - UploadPart成功请求
  - UploadPartCopy成功请求
  - DeleteObject成功请求
  - DeleteObjects成功请求

  如下图所示：

报警规则

报警规则Tab页能够展示和管理报警规则，如下图所示：

报警规则页的使用和相关说明请参见使用报警服务。

监控关注事项以及使用指导

监控关注点以及使用指南请参见监视诊断和故障排除。

11.3. 使用报警服务

本文主要介绍OSS监控服务控制台中报警规则的概览及配置方法。

在介绍OSS监控服务控制台之前，请先阅读云监控提供的监控服务文档，了解基本概念并进行报警联系人和报警联系组的配置。

- 报警服务概览
- 报警联系人和联系组

因为OSS的报警规则是根据OSS监控项设置的，所以类似于OSS监控项的维度分类，将其分成两个报警维度：用户层级和Bucket层级。

报警规则页

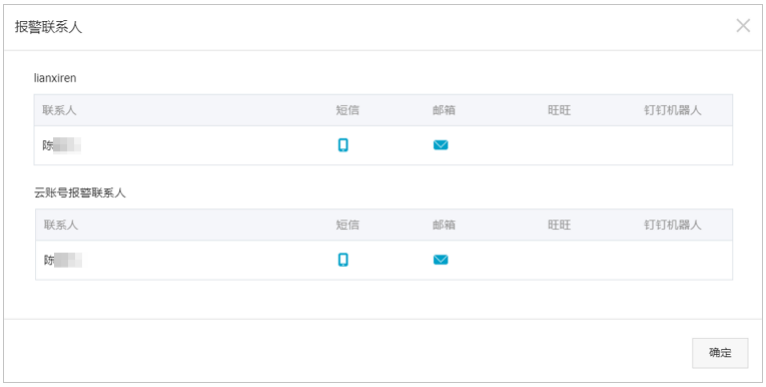
报警规则页是OSS监控报警相关的规则管理页面，您可以查看、修改、启用、禁用和删除对应的报警规则，而且能够查看该报警规则对应的历史报警情况。

- 单击对应报警规则的**查看**，可以查看该报警规则的内容。
- 单击对应报警规则的**修改**，就可以对该报警规则进行修改。
- 单击对应报警规则的**删除**，就可以删除该报警规则。选中多条报警规则，然后单击表格最下方的**删除**按钮，可以批量删除报警规则。
- 如果报警规则处于*已启用*状态，单击该报警规则的**禁用**，可以禁用该报警规则，报警规则失效，用户不能再收到对应的告警信息。选中多条报警规则，然后单击表格最下方的**禁用**按钮，可以批量禁用报警规则。
- 如果报警规则处于*已禁用*状态，单击该报警规则的**启用**，可以启用该报警规则，报警规则重新生效，能检测并发出对应的告警信息。选中多条报警规则，然后单击表格最下方的**启用**按钮，可以批量启用报警规则。
- 单击对应报警规则的**报警历史**，可以查看该报警规则历史发生的所有的告警情况。

相关概念：

- 报警历史指的是该报警规则的状态变化历史，例如从正常变成告警状态，是一个状态变化；从告警变成正常也是状态变化；还有一个特殊的状态变化：通道沉默。
- 当通知对象为**通道沉默**时，表示该报警规则触发告警之后的指定时间内一直满足报警触发状态（即一直在告警，没有恢复到正常状态）。此时，系统不向通知对象发送告警信息，直到通道沉默时间结束，才会有新的报警信息发送到通知对象。
- 报警历史信息能够保存一个月，即一个月之前的告警信息会被自动清理。查询时一次最多只能查询3天的数据，但不支持查询31天前的数据。

单击具体报警规则的通知对象后的**查看**，可以显示该通知对象（报警联系组）的成员以及每个成员接收告警信息的方式（短信、邮箱或者旺旺），如下图所示：



查看报警规则

根据报警规则页中下面的控件信息能够快速定位到被搜索的报警规则。

- 报警维度下拉框： 全部和Bucket层级。当选项为*全部*时，显示所有用户层级和Bucket层级的报警规则。

- Bucket下拉框：当报警维度下拉框为*Bucket层级*时，这里可以罗列该账号下所有的Bucket。选择对应的Bucket，可以展示属于该Bucket的所有报警规则。

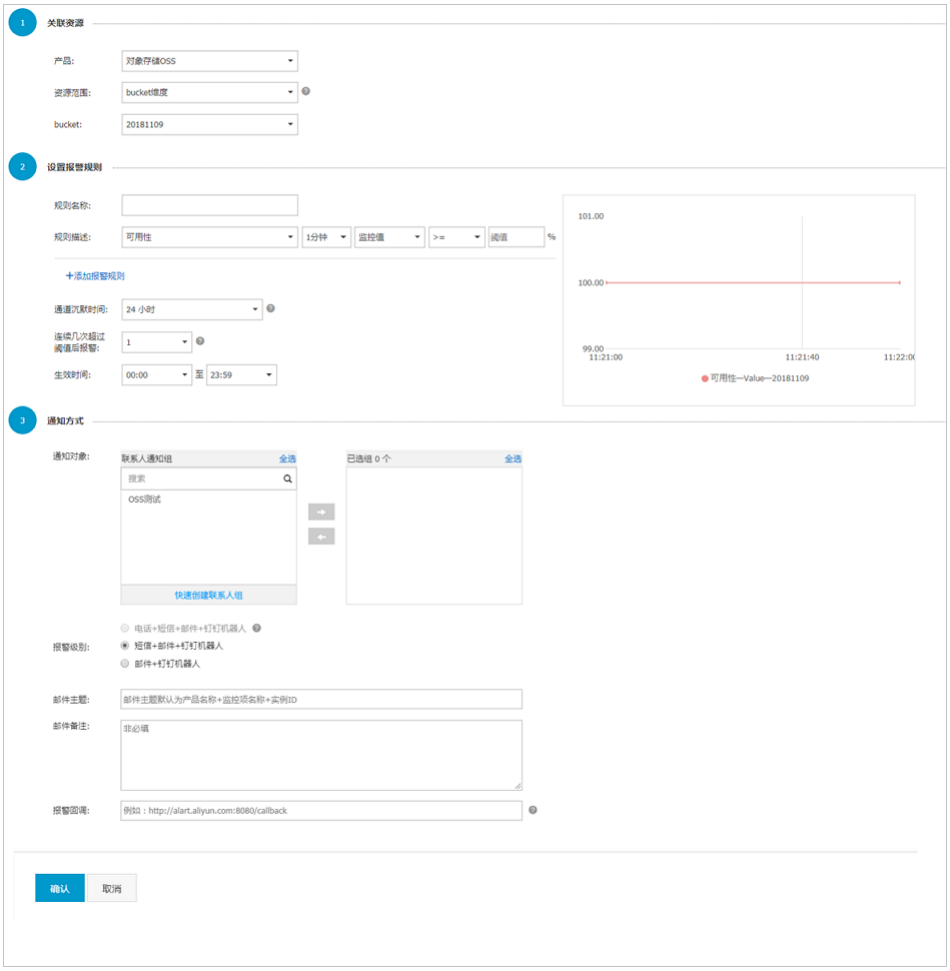
- 监控项下拉框：罗列所有的OSS的监控项，包括用户层级和Bucket层级的监控项。当选项为*全部*时，显示用户层级或者Bucket层级所有监控项的报警规则。
- 状态下拉框：可选择显示处于指定状态的报警规则，如*全部*、*正常状态*、*报警状态*、*数据不足*、*启用*、*禁用*。选择全部时，显示所有状态的报警规则。
- 维度下拉框：可分维度显示报警规则，如*全部用户维度*、*分组维度*、*实例维度*。

添加报警规则

1. 进入创建报警规则页面，您可以通过如下方式进入：
   - 在用户概况Tab页单击**服务监控总览**任意图表内的

     按钮。
   - 在Bucket列表Tab页选中指定的Bucket，之后单击**创建报警规则**按钮。
   - 在Bucket列表Tab页选中指定的Bucket，之后单击**服务监控总览**任意图表内的

     按钮。



2. 根据需求配置报警规则。
   - **关联资源**
     - *产品*：选择*对象存储OSS*。
     - *资源范围*：根据您的需求选择*全部资源*或*Bucket维度*。
     - *Bucket*（针对Bucket维度）：选择指定的Bucket，可一次选中多个Bucket。
   - **设置报警规则**
     - **规则名称**：自定义。
     - **规则描述**：根据需要选择监控的内容、时间及数值。
     - **+添加报警规则**：单击可添加多条规则。
     - **通道沉默时间**：报警发生后如果未恢复正常，间隔多久重复发送一次报警通知。
     - **连续几次超过阈值后报警**：即连续几次报警的探测结果符合您设置的规则描述，才会触发报警。例如：设置的规则描述为“一分钟内公网流出量大于100MBytes，连续3次超过阈值后报警”。则连续出现3次一分钟内公网流出量大于100MBytes的情况，才会触发报警。

- *生效时间*：选择报警规则的生效时间。

- **通知方式**
  - **通知对象**：若事先已经按照报警联系人和联系组设置好报警联系组，直接选择对应的联系人组；若没有设置过报警联系人组，则单击**快创建联系人组**，按照提示创建即可。
  - **报警级别**：设置报警规则通知的方式。
  - **邮件备注**（可选）：设置邮件备注信息。
  - **报警回调**：填写公网可访问的URL，云监控会将报警信息通过POST请求推送到该地址，目前仅支持HTTP协议。

3. 单击**确认**，完成报警规则的设置。

注意事项

目前属于某个Bucket的报警规则存在性并没有与该Bucket的存在性强关联，即如果删除了某个Bucket，属于这个Bucket的报警规则依然存在。建议您在删除Bucket之前先删除对应的报警规则。

11.4. 访问监控数据

OSS监控服务为您提供系统基本运行状态、性能以及计量等方面的监控数据指标，帮助您跟踪请求、分析使用情况、统计业务趋势，及时发现以及诊断系统的相关问题。本文介绍如何使用云监控服务提供的API或SDK查询OSS监控数据。

> **说明** 云监控服务SDK示例，请参见Java SDK使用手册。

Space

Space用于指定监控的云服务。OSS监控服务使用的Namespace为 `acs_oss_dashboard` 。

例如，通过Java SDK指定监控OSS服务的示例代码如下：

```
DescribeMetricListRequest request = new DescribeMetricListRequest();
request.setNamespace("acs_oss_dashboard");
```

StartTime和EndTime

StartTime和EndTime用于指定查询监控数据的时间范围。云监控的时间参数取值范围采用左开右闭的形式 `(StartTime, EndTime]` ，即可以查询StartTime到EndTime之间的数据（包含EndTime的数据）。StartTime和EndTime的时间间隔不能大于31天，且无法查询31天以前的数据。

例如，通过Java SDK指定查询监控数据时间范围的示例代码如下：

```
//设置监控数据的结束时间。
request.setEndTime("2019-05-13 11:06:27");
//设置监控数据的开始时间。
request.setStartTime("2019-05-13 10:20:27");
```

Dimensions

Dimensions用于指定待查询的Bucket。不指定Dimensions时，表示查询用户层级数据。层级说明请参见Metric。

例如，通过Java SDK查询Bucket数据的示例代码如下：

```
//填写待查询数据的Bucket名称。
request.setDimensions("{\"BucketName\":\"<yourBucketName>\"}");
```

Period

Period用于指定指标项的查询周期。OSS监控的计量类指标查询周期为3600s，其他所有指标的查询周期均为60s。各指标项的说明，请参见Metric。

例如，通过Java SDK监控某个非计量类指标的示例代码如下：

```
request.setPeriod("60");
```

Metric

Metric用于指定查询的指标。Java SDK代码示例如下：

```
//设置Metric名称。
request.setMetric("<MetricName>");
```

Metric分为非计量类以及计量类指标两大类：

- 非计量类指标

层级	Metric	对应指标项名称	单位
	UserAvailability	可用性	%
	UserRequestValidRate	有效请求率	%
	UserTotalRequestCount	总请求数	次数
	UserValidRequestCount	有效请求数	次数
	UserInternetSend	公网流出流量	字节

层级	Metric	对应指标项名称	单位
用户层级	UserInternetRecv	公网流入流量	字节
	UserIntranetSend	内网流出流量	字节
	UserIntranetRecv	内网流入流量	字节
	UserCdnSend	CDN流出流量	字节
	UserCdnRecv	CDN流入流量	字节
	UserSyncSend	跨区域复制流出流量	字节
	UserSyncRecv	跨区域复制流入流量	字节
	UserServerErrorCount	服务端错误请求总数	次数
	UserServerErrorRate	服务端错误请求占比	%
	UserNetworkErrorCount	网络错误请求总数	次数
	UserNetworkErrorRate	网络错误请求占比	%
	UserAuthorizationErrorCount	客户端授权错误请求总数	次数
	UserAuthorizationErrorRate	客户端授权错误请求占比	%
	UserResourceNotFoundErrorCount	客户端资源不存在错误请求总数	次数
	UserResourceNotFoundErrorRate	客户端资源不存在错误请求占比	%
	UserClientTimeoutErrorCount	客户端超时错误请求总数	次数
	UserClientOtherErrorRate	客户端超时错误请求占比	%
	UserClientOtherErrorCount	客户端其他错误请求总数	次数
	UserClientOtherErrorRate	客户端其他错误请求占比	%
	UserSuccessCount	成功请求总数	次数
	UserSuccessRate	成功请求占比	%
	UserRedirectCount	重定向请求总数	次数
	UserRedirectRate	重定向请求占比	%
	Availability	可用性	%
	RequestValidRate	有效请求率	%
	TotalRequestCount	总请求数	次数
	ValidRequestCount	有效请求数	次数
	InternetSend	公网流出流量	字节
	InternetRecv	公网流入流量	字节
	IntranetSend	内网流出流量	字节
	IntranetRecv	内网流入流量	字节
	InternetSendBandwidth	公网流出带宽	bps
	InternetRecvBandwidth	公网流入带宽	bps
	IntranetSendBandwidth	内网流出带宽	bps
	IntranetRecvBandwidth	内网流入带宽	bps
	CdnSend	CDN流出流量	字节
	CdnRecv	CDN流入流量	字节
	SyncSend	跨区域复制流出流量	字节
	SyncRecv	跨区域复制流入流量	字节
	ServerErrorCount	服务端错误请求总数	次数
	ServerErrorRate	服务端错误请求占比	%
	NetworkErrorCount	网络错误请求总数	次数
	NetworkErrorRate	网络错误请求占比	%

层级	Metric	对应指标项名称	单位
Bucket层级	AuthorizationErrorCount	客户端授权错误请求总数	次数
	AuthorizationErrorRate	客户端授权错误请求占比	%
	ResourceNotFoundErrorCount	客户端资源不存在错误请求总数	次数
	ResourceNotFoundErrorRate	客户端资源不存在错误请求占比	%
	ClientTimeoutErrorCount	客户端超时错误请求总数	次数
	ClientTimeoutErrorRate	客户端超时错误请求占比	%
	ClientOtherErrorCount	客户端其他错误请求总数	次数
	ClientOtherErrorRate	客户端其他错误请求占比	%
	SuccessCount	成功请求总数	次数
	SuccessRate	成功请求占比	%
	RedirectCount	重定向请求总数	次数
	RedirectRate	重定向请求占比	%
	GetObjectE2eLatency	GetObject请求平均E2E延时	毫秒
	GetObjectServerLatency	GetObject请求平均服务器延时	毫秒
	MaxGetObjectE2eLatency	GetObject请求最大E2E延时	毫秒
	MaxGetObjectServerLatency	GetObject请求最大服务器延时	毫秒
	HeadObjectE2eLatency	HeadObject请求平均E2E延时	毫秒
	HeadObjectServerLatency	HeadObject请求平均服务器延时	毫秒
	MaxHeadObjectE2eLatency	HeadObject请求最大E2E延时	毫秒
	MaxHeadObjectServerLatency	HeadObject请求最大服务器延时	毫秒
	PutObjectE2eLatency	PutObject请求平均E2E延时	毫秒
	PutObjectServerLatency	PutObject请求平均服务器延时	毫秒
	MaxPutObjectE2eLatency	PutObject请求最大E2E延时	毫秒
	MaxPutObjectServerLatency	PutObject请求最大服务器延时	毫秒
	PostObjectE2eLatency	PostObject请求平均E2E延时	毫秒
	PostObjectServerLatency	PostObject请求平均服务器延时	毫秒
	MaxPostObjectE2eLatency	PostObject请求最大E2E延时	毫秒
	MaxPostObjectServerLatency	PostObject请求最大服务器延时	毫秒
	AppendObjectE2eLatency	AppendObject请求平均E2E延时	毫秒
	AppendObjectServerLatency	AppendObject请求平均服务器延时	毫秒
	MaxAppendObjectE2eLatency	AppendObject请求最大E2E延时	毫秒
	MaxAppendObjectServerLatency	AppendObject请求最大服务器延时	毫秒
	UploadPartE2eLatency	UploadPart请求平均E2E延时	毫秒
	UploadPartServerLatency	UploadPart请求平均服务器延时	毫秒
	MaxUploadPartE2eLatency	UploadPart请求最大E2E延时	毫秒
	MaxUploadPartServerLatency	UploadPart请求最大服务器延时	毫秒
	UploadPartCopyE2eLatency	UploadPartCopy请求平均E2E延时	毫秒
	UploadPartCopyServerLatency	UploadPartCopy请求平均服务器延时	毫秒
	MaxUploadPartCopyE2eLatency	UploadPartCopy请求最大E2E延时	毫秒
	MaxUploadPartCopyServerLatency	UploadPartCopy请求最大服务器延时	毫秒
	GetObjectCount	GetObject成功请求数	次数
	HeadObjectCount	HeadObject成功请求数	次数
	PutObjectCount	PutObject成功请求数	次数

层级	Metric	对应指标项名称	单位
	PostObjectCount	PostObject成功请求数	次数
	AppendObjectCount	AppendObject成功请求数	次数
	UploadPartCount	UploadPart成功请求数	次数
	UploadPartCopyCount	UploadPartCopy成功请求数	次数
	DeleteObjectCount	DeleteObject成功请求数	次数
	DeleteObjectsCount	DeleteObjects成功请求数	次数

- 计量类指标

  查询以下Metric时如果指定了Dimensions，则表示查询的是Bucket层级的数据。如果未指定Dimensions，则表示查询的是用户层级的数据。

Metric	对应指标项名称	单位
MeteringStorageUtilization	存储大小	字节
MeteringGetRequest	Get类请求数	次数
MeteringPutRequest	Put类请求数	次数
MeteringInternetTX	公网流出计量流量	字节
MeteringCdnTX	CDN流出计量流量	字节
MeteringSyncRX	跨区域复制流入计量流量	字节

11.5. 监控指标参考

根据用户使用场景，OSS的指标分为用户层级和存储空间（Bucket）层级。为了更好地观察监控数据以及匹配计费策略，OSS对现有的监控指标项进行统计分析，提供了一段时间内的统计数据，如请求状态分布统计和当月计量统计。

> **说明**
> - 监控指标项中包含按分钟级别汇总的时间序指标（例如求和、求最大值或者求均值等），以及按小时级别汇总的计量指标等。
> - 云监控默认为您提供ECS监控大盘，展示ECS部分监控数据。如果您需要查看其他云服务（例如OSS用户层级或者Bucket层级指标）监控项，可以将相关监控项通过添加图表的形式添加到同一个监控大盘。具体步骤，请参见管理自定义监控大盘中的监控图表。

用户层级指标

用户层级指标是指从阿里云账号级别对OSS使用的总体情况进行监控的指标信息，即该阿里云账号下所有Bucket相关监控数据的汇总。包括当月计量统计、服务监控总览和请求状态详情三个方面。

- 当月计量统计指标是指从当月的1号0点开始，到当月计量采集截止时间之内计量指标的统计数据。具体指标项如下：

指标名称	单位	描述
存储大小	字节	从本月1号0点开始累积到计量采集截止时间为止，用户所有Bucket占用的存储总大小。
公网流出流量	字节	从本月1号0点开始累积到计量采集截止时间为止，用户所使用的所有公网流出流量的总和。
Put类请求数	次数	从本月1号0点开始累积到计量采集截止时间为止，用户所使用的所有Put类请求的总和。
Get类请求数	次数	从本月1号0点开始累积到计量采集截止时间为止，用户所使用的所有Get类请求的总和。

当月计量统计

- 服务监控总览指标属于基础服务指标。具体指标项如下：

指标名称	单位	描述
可用性	%	存储服务的系统可用性衡量指标。通过公式 `1 - 服务端错误请求（返回状态码为5xx）占总请求的百分比` 获取。
有效请求率	%	有效请求占总请求数的百分比。
总请求数	次数	被OSS服务端接收并处理的请求总数。
有效请求数	次数	返回状态码为2xx和3xx的请求总数。
公网流出流量	字节	通过互联网网络的下行流量。
公网流入流量	字节	通过互联网网络的上行流量。
内网流出流量	字节	通过服务系统内部网络的下行流量。
内网流入流量	字节	通过服务系统内部网络的上行流量。
CDN流出流量	字节	开通CDN加速服务之后，通过CDN产生的下行流量，即回源流量。

指标名称	单位	描述
CDN流入流量	字节	开通CDN加速服务之后，通过CDN产生的上行流量。
跨区域复制流出流量	字节	开通跨区域复制之后，数据复制过程产生的下行流量。
跨区域复制流入流量	字节	开通跨区域复制之后，数据复制过程产生的上行流量。

服务监控总览

- 请求状态详情指标是指根据请求返回状态码或者OSS错误码进行分类的请求的监控信息，属于基础服务指标。具体指标项如下：

指标名称	单位	描述
服务端错误请求总数	次数	返回状态码为5xx的系统级错误请求总数。
服务端错误请求占比	%	服务端错误请求总数占总请求数的百分比。
网络错误请求总数	次数	HTTP状态码为499的请求总数。
网络错误请求占比	%	网络错误请求数占总请求数的百分比。
客户端授权错误请求总数	次数	返回状态码403的请求总数。
客户端授权错误请求占比	%	授权错误请求数占总请求数的百分比。
客户端资源不存在错误请求总数	次数	返回状态码为404的请求总数。
客户端资源不存在错误请求占比	%	资源不存在错误请求数占总请求数百分比。
客户端超时错误请求总数	次数	返回状态码为408或者返回的OSS错误码为RequestTimeout的请求总数。
客户端超时错误请求占比	%	客户端超时错误请求总数占总请求数的百分比。
客户端其他错误请求总数	次数	除了以上提到的客户端错误请求之外的其他返回状态码为4xx的请求总数。
客户端其他错误请求占比	%	客户端其他错误请求数占总请求数的百分比。
成功请求总数	次数	返回状态码为2xx的请求总数。
成功请求占比	%	成功请求数占总请求数的百分比。
重定向请求总数	次数	返回状态码为3xx的请求总数。
重定向请求占比	%	重定向请求数占总请求数的百分比。

请求状态详情

Bucket层级指标

Bucket层级指标除包含以上所有用户层级指标以外，还包括计量参考、延时和成功请求操作分类等计量指标和性能指标。

> **注意** 与用户层级指标监控当前账号下所有Bucket指标信息不同的是，Bucket层级指标仅监控单个Bucket的指标信息。例如，用户层级指标统计的存储大小，表示在计量采集截止时间前属于该账号下所有Bucket占用的存储总大小。而Bucket层级指标统计的存储大小，则表示在计量采集截止时间前该Bucket占用的存储总大小。

- 具体指标项如下：

指标名称	单位	描述
存储大小	字节	该Bucket每小时使用的平均存储大小。
公网流出流量	字节	该Bucket每小时的公网流出流量的总和。
Put类请求数	次数	该Bucket每小时的Put类请求的总和。
Get类请求数	次数	该Bucket每小时的Get类请求的总和。

计量参考

- 请求延时是系统性能的直观反映，且只对返回状态码为2xx的成功请求进行监控。监控服务提供了分钟级别的平均延时和最大延时两类指标，分别反映系统平均响应能力和系统抖动情况。

  延时监控指标分别从E2E和服务器两条不同的链路进行收集，便于分析性能热点以及环境问题，其中：

  - E2E延时是指向OSS系统发出的成功请求的端到端滞后时间，包括在OSS系统中读取请求、发送响应以及接受响应确认所需的处理时间。
  - 服务器延时是指OSS系统成功处理请求所使用的滞后时间，不包括E2E延时中的网络滞后时间。

  具体指标项如下：

指标名称	单位	描述
GetObject请求平均E2E延时	毫秒	请求API为GetObject的成功请求的平均端到端延时。
GetObject请求平均服务器延时	毫秒	请求API为GetObject的成功请求的平均服务器延时。
GetObject请求最大E2E延时	毫秒	请求API为GetObject的成功请求的最大端到端延时。

指标名称	单位	描述
GetObject请求最大服务器延时	毫秒	请求API为GetObject的成功请求的最大服务器延时。
HeadObject请求平均E2E延时	毫秒	请求API为HeadObject的成功请求的平均端到端延时。
HeadObject请求平均服务器延时	毫秒	请求API为HeadObject的成功请求的平均服务器延时。
HeadObject请求最大E2E延时	毫秒	请求API为HeadObject的成功请求的最大端到端延时。
HeadObject请求最大服务器延时	毫秒	请求API为HeadObject的成功请求的最大服务器延时。
PutObject请求平均E2E延时	毫秒	请求API为PutObject的成功请求的平均端到端延时。
PutObject请求平均服务器延时	毫秒	请求API为PutObject的成功请求的平均服务器延时。
PutObject请求最大E2E延时	毫秒	请求API为PutObject的成功请求的最大端到端延时。
PutObject请求最大服务器延时	毫秒	请求API为PutObject的成功请求的最大服务器延时。
PostObject请求平均E2E延时	毫秒	请求API为PostObject的成功请求的平均端到端延时。
PostObject请求平均服务器延时	毫秒	请求API为PostObject的成功请求的平均服务器延时。
PostObject请求最大E2E延时	毫秒	请求API为PostObject的成功请求的最大端到端延时。
PostObject请求最大服务器延时	毫秒	请求API为PostObject的成功请求的最大服务器延时。
AppendObject请求平均E2E延时	毫秒	请求API为AppendObject的成功请求的平均端到端延时。
AppendObject请求平均服务器延时	毫秒	请求API为AppendObject的成功请求的平均服务器延时。
AppendObject请求最大E2E延时	毫秒	请求API为AppendObject的成功请求的最大端到端延时。
AppendObject请求最大服务器延时	毫秒	请求API为AppendObject的成功请求的最大服务器延时。
UploadPart请求平均E2E延时	毫秒	请求API为UploadPart的成功请求的平均端到端延时。
UploadPart请求平均服务器延时	毫秒	请求API为UploadPart的成功请求的平均服务器延时。
UploadPart请求最大E2E延时	毫秒	请求API为UploadPart的成功请求的最大端到端延时。
UploadPart请求最大服务器延时	毫秒	请求API为UploadPart的成功请求的最大服务器延时。
UploadPartCopy请求平均E2E延时	毫秒	请求API为UploadPartCopy的成功请求的平均端到端延时。
UploadPartCopy请求平均服务器延时	毫秒	请求API为UploadPartCopy的成功请求的平均服务器延时。
UploadPartCopy请求最大E2E延时	毫秒	请求API为UploadPartCopy的成功请求的最大端到端延时。
UploadPartCopy请求最大服务器延时	毫秒	请求API为UploadPartCopy的成功请求的最大服务器延时。

延时

- 成功请求的监控一定程度上反应了系统处理访问请求的能力。具体指标项如下：

指标名称	单位	描述
GetObject成功请求数	次数	请求API为GetObject的成功请求数。
HeadObject成功请求数	次数	请求API为HeadObject的成功请求数。
PutObject成功请求数	次数	请求API为PutObject的成功请求数。
PostObject成功请求数	次数	请求API为PostObject的成功请求数。
AppendObject成功请求数	次数	请求API为AppendObject的成功请求数。
UploadPart成功请求数	次数	请求API为UploadPart的成功请求数。
UploadPartCopy成功请求数	次数	请求API为UploadPartCopy的成功请求数。
DeleteObject成功请求数	次数	请求API为DeleteObject的成功请求数。
DeleteObjects成功请求数	次数	请求API为DeleteObjects的成功请求数。

成功请求

- 具体指标项如下：

指标名称	单位	描述
[镜像回源]指定回源源站的正常请求流入流量	字节	统计指定某个源站时，返回值200和206的请求流入流量之和。
[镜像回源]指定返回值和回源源站的请求流入流量	字节	统计指定某个源站以及某个返回值时的请求流入流量。
[镜像回源]指定回源源站的正常请求平均传输速度	字节/秒	统计指定某个源站时，返回值200和206的请求流入速率的平均值。

指标名称	单位	描述
[镜像回源]指定返回值和回源源站的请求平均传输速度	字节/秒	统计指定某个源站以及某个返回值时的请求流入速率的平均值。
[镜像回源]指定回源源站的正常请求总数	次数	统计指定某个源站时，返回值为200和206的请求总数。
[镜像回源]指定返回值和回源源站的请求总数	次数	统计指定某个源站和返回值时的请求总数。
[镜像回源]指定回源源站的正常请求平均延时	毫秒	统计指定某个源站时，返回值为200和206的请求平均延时。
[镜像回源]指定返回值和回源源站的请求平均延时	毫秒	统计指定某个源站以及某个返回值时的请求平均延时。
[镜像回源]指定回源源站的状态码2xx、3xx、4xx、5xx所占总请求量的百分比	%	统计指定某个源站时，各类请求状态码（2xx、3xx、4xx、5xx）所占总请求次数的百分比。
[镜像回源]指定回源源站的状态码2xx、3xx、4xx、5xx的请求数量	次数	统计指定某个源站时，各类请求状态码（2xx、3xx、4xx、5xx）的请求次数。

镜像回源

11.6. 监控、诊断和故障排除

相对于传统应用程序，开发云端应用虽然降低了用户在基础设施搭建、运维等方面的成本，但却增大了监控、诊断和故障排查的难度。OSS存储服务为您提供了丰富的监控和日志信息，帮助您深刻洞察程序行为，及时发现并快速定位问题。

本文主要描述如何使用OSS监控服务、日志记录功能以及其他第三方工具来监控、诊断和排查应用业务使用OSS存储服务时遇到的相关问题，帮助您达到如下目标：

- 实时监控OSS存储服务的运行状况和性能，并及时报警通知。
- 获取有效的方法和工具来定位问题。
- 根据相关问题的处理和操作指南，快速解决与OSS相关的问题。

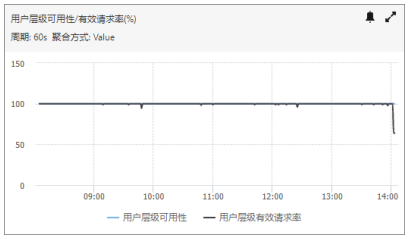
本文包括如下内容：

- 服务监控：介绍如何使用OSS监控服务持续监控OSS存储服务的运行状况和性能。
- 跟踪诊断：介绍如何使用OSS监控服务和日志记录功能诊断问题；另外，还介绍如何关联各种日志文件中的相关信息进行跟踪诊断。
- 故障排除：提供常见的问题场景和故障排除方法。

服务监控

- 监视总体运行状况
  - 可用性和有效请求率

    可用性和有效请求率是有关系统稳定性和用户是否正确使用系统的最重要指标，指标小于100%说明某些请求失败。



    可能因为一些系统优化因素出现暂时性的低于100%，例如为了负载均衡而出现分区迁移，此时OSS的SDK能够提供相关的重试机制无缝处理这类间歇性的失败情况，使得业务端无感知。

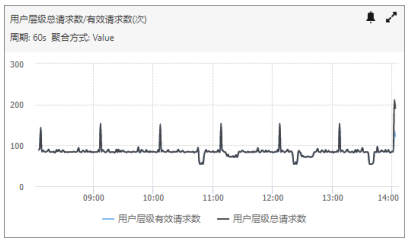
    对于有效请求率低于100%的情况，您需要根据自己的使用情况进行分析，可以通过请求分布统计或者请求状态详情确定错误请求的具体类型、原因，并排除故障。

    对于某些业务场景，出现有效请求率低于100%是符合预期的。例如，用户需要先检查访问的Object是否存在，然后根据Object的存在性采取一定的操作，如果Object不存在，检测Object存在性的读取请求会收到404错误（资源不存在错误），导致该指标项低于100%。

    对于系统可用性指标要求较高的业务，可以对其设置报警规则，当低于预期阈值时，进行报警。

  - 总请求数和有效请求数

    该指标从总访问量角度来反应系统运行状态，当有效请求数不等于总请求数时表明某些请求失败。



    您可以关注总请求数或者有效请求数的波动状况，特别是突然上升或者下降的情况，需要进行跟进调查，可以通过设置报警规则进行及时通知。具体操作，请参见报警服务使用指南。

- 请求状态分布统计

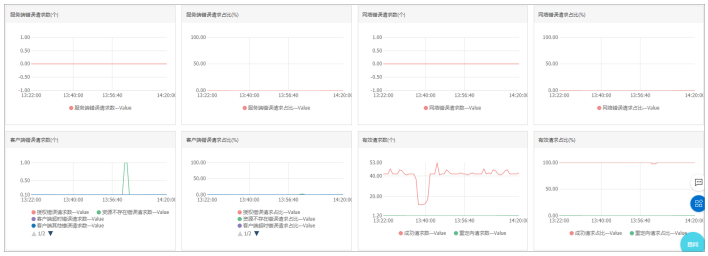
  当可用性或者有效请求率低于100%（或者有效请求数不等于总请求数时），可以通过查看请求状态分布统计快速确定请求的错误类型。有关该统计监控指标的更多信息，请参见OSS监控指标参考手册。

用户层级请求状态分布

监控项	统计值	百分比
服务端错误请求数	1246960次	5.56%
网络错误请求数	1232826次	5.49%
授权错误请求数	1246098次	5.55%
资源不存在错误请求数	1252403次	5.58%
客户端超时错误请求数	2485623次	11.08%
客户端其他错误请求数	12472186次	55.64%
成功请求数	1246815次	5.56%
重定向请求数	1232157次	5.49%
总计	22415068次	100%

- 监视请求状态详情

  请求状态详情是在请求状态分布统计的基础上进一步细化请求监控状态，可以更加深入、具体地监视某类请求。



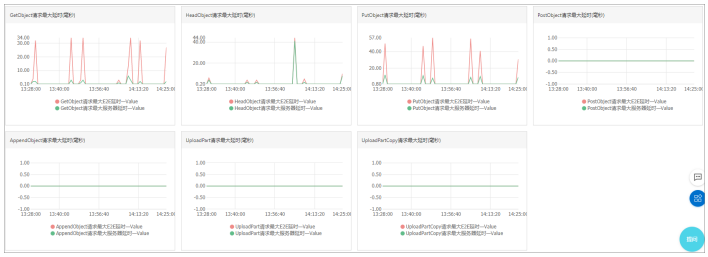
- 监视性能

  监控服务提供了以下监控项来监控性能相关的指标：

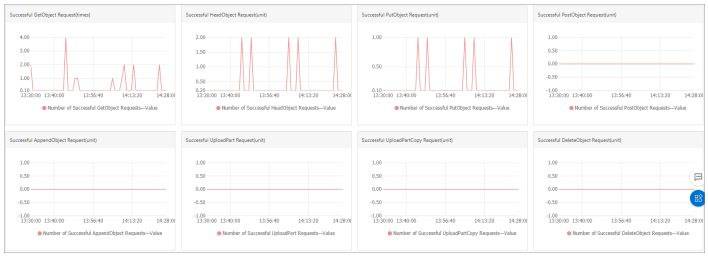
  - 平均延时，包括E2E平均延时和服务器平均延时

延时指标显示API操作类型处理请求所需的平均和最大时间。其中E2E延时是端到端延迟指标，除了包括处理请求所需的时间外，还包括读取请求和发送响应所需的时间以及请求在网络上传输的延时；而服务器延时只是请求在服务器端被处理的时间，不包括与客户端通信的网络延时。所以当出现E2E延时突然升高的情况下，如果服务器延时并没有很大的变化，那么可以判定是网络的不稳定因素造成的性能问题，排除OSS系统内部故障。

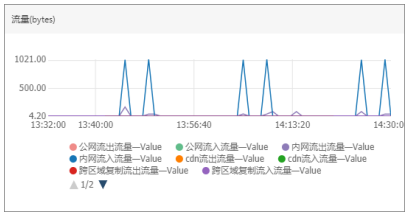
  - 最大延时，包括E2E最大延时和服务器最大延时



  - 成功请求操作分类



- 流量



流量指标从用户或者具体的Bucket层级的总体情况进行监控，关注公网、内网、CDN回源以及跨域复制等场景中的网络资源占用状况。

以上各个监控项（除流量）都分别从API操作类型进行分类监控，包括：

- GetObject
- HeadObject
- PutObject
- PostObject
- AppendObject
- UploadPart
- UploadPartCopy

成功请求操作分类除了以上提到的API之外，还提供以下两个API操作类型请求数量的监控：

- DeleteObject
- DeleteObjects

对于性能类指标项，您需要重点关注其突发的异常变化。例如，平均延时突然出现尖峰，或者长时间处于高出正常请求延时的基线上方等。您可以通过对性能指标设置对应的报警规则，当指标低于或者超过阈值时及时通知到相关人员。

- 监视计量

目前，OSS监控服务只支持监控存储空间大小、公网流出流量、Put类请求数和Get类请求数（不包括跨区域复制流出流量和CDN流出流量），不支持对计量数据的报警设置和OpenAPI的读取。

OSS监控服务对Bucket层级的计量监控数据进行小时级别的粒度采集，可以在具体的Bucket监控视图中查看其持续的监控趋势图。您可以根据监控视图分析其业务的存储服务资源使用趋势并且进行成本的预估等。

OSS监控服务还提供了用户层级和Bucket层级两个不同维度的当月消费的资源数统计。即统计阿里云账户或者某个Bucket当月消耗的OSS资源总量，每小时更新一次，帮助您了解本月资源的使用情况并计算消费。

有关OSS的计费项和计费方式的更多信息，请参见计量项和计费项。

> **说明** 监控服务中提供的计量数据是尽最大可能推送的，可能会与实际消费账单不一致，请以费用中心的数据为准。

跟踪诊断

- 问题诊断
  - 诊断性能

    对于应用程序的性能判断多带有主观因素，您需要根据具体的业务场景确定满足业务需求的基准线，来判定性能问题。另外，从客户端发起的请求，能引起性能问题的因素贯穿整个请求链路。例如OSS存储服务负载过大、客户端TCP配置问题或网络基础结构中存在的流量瓶颈等。

    因此诊断性能问题首先需要设置合理的基准线，然后通过监控服务提供的性能指标确定性能问题可能的根源位置，然后根据日志查到详细的信息以便进一步诊断并且排除故障。

  - 诊断错误

    客户端应用程序会在请求发生错误时接收到服务端返回的相关错误信息，监控服务也会记录并显示各种错误类型请求的计数和占比。您也可以通过检查服务器端日志、客户端日志和网络日志来获取相关单个请求的详细信息。通常，响应中返回的HTTP状态代码和OSS错误码以及OSS错误信息都指示请求失败的原因。

    有关错误响应的更多信息，请参见 OSS错误响应。

  - 使用日志功能

    OSS存储服务为用户的请求提供服务端日志记录功能，能帮助用户记录端到端的详细请求日志跟踪。

    有关如何开启并使用日志功能，请参见设置日志。

    有关日志服务的命名规则以及记录格式的更多信息，请参见设置访问日志记录。

  - 使用网络日志记录工具

    在大多数情况下，通过日志服务记录的存储日志和客户端应用程序的日志数据已足以诊断问题，但在某些情况下，可能需要更详细的信息，这时需要使用网络日志记录工具捕获客户端和服务器之间的流量，可以更详细地获取客户端和服务器之间交换的数据以及底层网络状况的详细信息，帮助问题的调查。例如，在某些情况下，用户请求可能会报告一个错误，而服务器端日志中却看不到任何该请求的访问情况，这时就可以使用OSS的日志服务功能记录的日志来调查该问题的原因是否出在客户端上，或者使用网络监视工具来调查网络问题。

    最常用的网络日志分析工具之一是Wireshark。该免费的协议分析器运行在数据包级别，能够查看各种网络协议的详细数据包信息，从而可以排查丢失的数据包和连接问题。

    有关如何使用WireShark的详情，请参见WireShark用户指南。

- E2E跟踪诊断

请求从客户端应用进程发起，通过网络环境，进入OSS服务端被处理，然后响应从服务端回到网络环境，被客户端接收。整个过程，是一个端到端的跟踪过程。关联客户端应用程序日志、网络跟踪日志和服务端日志，有助于排查问题的详细信息根源，发现潜在的问题。

在OSS存储服务中，提供了RequestID作为关联各处日志信息的标识符。另外，通过日志的时间戳，不仅可以迅速查找和定位日志范围，还能够了解在请求发生时间点范围内，客户端应用、网络或者服务系统发生的其他事件，有利于问题的分析和调查。

- RequestID

  OSS服务会为接收的每个请求分配唯一的服务器请求ID，即RequestID。在不同的日志中，RequestID位于不同的字段中：

  - 在OSS日志功能记录的服务端日志记录中，RequestID出现在“Request ID”列中。
  - 在网络跟踪（如WireShark捕获的数据流）中，RequestID作为x-oss-request-id标头值出现在响应消息中。
  - 在客户端应用中，需要客户端code实现的时候将请求的RequestID自行打印到客户端日志中。最新版本的Java SDK已经支持打印正常请求的RequestID信息，可以通过各个API接口返回的Result结果的getRequestId这个方法获取。OSS的各个版本SDK都支持打印出异常请求的RequestID，可以通过调用OSSException的getRequestId方法获取。

- 时间戳

  使用时间戳来查找相关日志项，需要注意客户端和服务器之间可能存在的时间偏差。在客户端上基于时间戳搜索日志功能记录的服务器端日志条目时，应加上或减去15分钟。

故障排除

- 性能相关常见问题
  - 平均E2E延时高，而平均服务端延时低

    前面介绍了平均E2E延时与平均服务器延时的区别。所以产生高E2E延时、低服务器延时可能的原因有两个：

    - 客户端应用程序响应慢
      - 可用连接数或可用线程数有限
        - 对于可用连接数问题，可以使用相关命令确定系统是否存在大量TIME_WAIT状态的连接。如果是，可以通过调整内核参数解决。
        - 对于可用线程数有限，可以先查看客户端CPU、内存、网络等资源是否已经存在瓶颈，如果没有，适当调大并发线程数。
        - 如果还不能解决问题，那么就需要通过优化客户端代码。例如，使用异步访问方式等。也可以使用性能分析功能分析客户端应用程序热点，然后具体优化。
      - CPU、内存或网络带宽等资源不足

        对于这类问题，需要先使用相关系统的资源监控查看客户端具体的资源瓶颈在哪里，然后通过优化代码使其对资源的使用更为合理，或者扩容客户端资源（使用更多的内核或者内存）。
    - 网络原因导致

      通常，因网络导致的端到端高延迟是由暂时状况导致的。可以使用Wireshark调查临时和持久网络问题，例如数据包丢失问题。

  - 平均E2E延时低，平均服务端延时低，但客户端请求延时高

    客户端出现请求延时高的情况，最可能的原因是请求还未达到服务端就出现了延时。所以应该调查来自客户端的请求为什么未到达服务器。

    对于客户端延迟发送请求，可能的客户端的原因有两个：

    - 可用连接数或可用线程数有限
      - 对于可用连接数问题，可以使用相关命令确定系统是否存在大量TIME_WAIT状态的连接。如果是，可以通过调整内核参数解决。
      - 对于可用线程数有限，可以先查看客户端CPU、内存、网络等资源是否已经存在瓶颈，如果没有，适当调大并发线程数。
      - 如果还不能解决问题，那么就需要通过优化客户端代码。例如，使用异步访问方式等。也可以使用性能分析功能分析客户端应用程序热点，然后具体优化。
    - 客户端请求出现多次重试，如果遇到这种情况，需要根据重试信息具体调查重试的原因再解决。可以通过下面方式确定客户端是否出现重试：
      - 检查客户端日志，详细日志记录会指示重试已发生过。以OSS的Java SDK为例，可以搜索如下日志提示，warn或者info的级别。如果存在该日志，说明可能出现了重试。

        ```
        [Server]Unable to execute HTTP request:
          或者
          [Client]Unable to execute HTTP request:
        ```

      - 如果客户端的日志级别为debug，以OSS的Java SDK为例，可以搜索如下日志，如果存在，那么肯定出现过重试。

        ```
        Retrying on
        ```

    如果客户端没有问题，则应调查潜在的网络问题，例如数据包丢失。可以使用工具（如Wireshark）调查网络问题。

  - 平均服务端延时高

    对于下载或者上传出现服务端高延时的情况，可能的原因有2个：

    - 大量客户端频繁访问同一个小Object

      这种情况，可以通过查看日志功能记录的服务端日志信息来确定是否在一段时间内，某个或某组Object被频繁访问。

      对于下载场景，建议用户为该Bucket开通CDN服务，利用CDN来提升性能，并且可以节约流量费用；对于上传场景，用户可以考虑在不影响业务需求的情况下，收回Object或Bucket的写访问权限。
    - 系统内部因素

      对于系统内部问题或者不能通过优化方式解决的问题，请提供客户端日志或者日志功能记录的日志信息中的RequestID，联系售后技术人员协助解决。

- 服务端错误问题

  对于服务端错误的增加，可以分为两个场景考虑：

  - 暂时性的增加

    对于这一类问题，您需要调整客户端程序中的重试策略，采用合理的退让机制，例如指数退避。这样不仅可以有效避免因为优化或者升级等系统操作（如为了系统负载均衡进行分区迁移等）暂时导致的服务不可用问题，还可以避开业务峰值的压力。
  - 永久性的增加

    如果服务端错误持续在一个较高的水平，那么请提供客户端日志或者日志功能记录的RequestID，联系售后技术人员协助调查。

- 网络错误问题

  网络错误是指服务端正在处理请求时，连接在非服务器端断开而来不及返回HTTP请求头的情况。此时系统会记录该请求的HTTP状态码为499。以下几种情况会导致服务器记录请求的状态码变为499：

  - 服务器在收到读写请求处理之前，会检查连接是否可用，不可用则为499。
  - 服务器正在处理请求时，客户端提前关闭了连接，此时请求被记录为499。

  在请求过程中，客户端主动关闭请求或者客户端网络断掉都会产生网络错误。对于客户端主动关闭请求的情况，需要调查客户端中的代码，了解客户端断开与存储服务连接的原因和时间。对于客户端网络断掉的情况，用户可以使用工具（如Wireshark）调查网络连接问题。

- 客户端错误问题
  - 客户端授权错误请求增加

    当监控中的客户端授权错误请求数增加，或者客户端程序接收到大量的403请求错误，那么最常见的可能原因有以下几个：

    - 用户访问的Bucket域名不正确
      - 如果用户直接用三级域名或者二级域名访问，那么可能的原因就是用户的Bucket并不属于该域名所指示的region内，例如，用户创建的Bucket的地域为杭州，但是访问的域名却为Bucket.oss-cn-shanghai.aliyuncs.com。这时需要确认Bucket的所属区域，然后更正域名信息。
      - 如果用户开启了CDN加速服务，那么可能的原因是CDN绑定的回源域名错了，请检查CDN回源域名是否为用户Bucket的三级域名。
      - 如果用户使用javascript客户端遇到403错误，可能的原因就是CORS（跨域资源共享）的设置问题，因为Web浏览器实施了“同源策略”的安全限制。用户需要先检查所属Bucket的CORS设置是否正确，并进行相应的更正。有关设置CORS的具体步骤，请参见跨域资源共享。
    - 访问控制问题
      - 用户使用主AK访问，那么用户需要检查是否AK设置出错，使用了无效AK。
      - 用户使用RAM子账号访问，那么用户需要确定RAM子账号是否使用了正确的子AK，或者对应子账号的相关操作是否已经授权。
      - 用户使用STS临时Token访问，那么用户需要确认一下这个临时Token是否已经过期。如果过期，需要重新申请。
      - 如果Bucket或者Object设置了访问控制，这个时候需要查看用户所访问的Bucket或者Object是否支持相关的操作。
    - URL过期

      授权第三方下载，即用户使用签名URL进行OSS资源访问，如果之前访问正常而突然遇到403错误，最大可能的原因是URL已经过期。
    - RAM子账号使用OSS周边工具的情况也会出现403错误。这类周边工具如ossftp、ossbrowser、OSS控制台客户端等，在填写相关的AK信息登入时就抛出错误，此时如果您的AK是正确填写的，那么您需要查看使用的AK是否为子账号AK，该子账号是否有GetService等操作的授权等。
  - 客户端资源不存在错误请求增加

    客户端收到404错误说明用户试图访问不存在的资源信息。当看到监控服务上资源不存在错误请求增加，那么最大可能是以下问题导致的：

    - 用户的业务使用方式。例如用户需要先检查Object是否存在来进行下一步动作，可以调用doesObjectExist（以JAVA SDK为例）方法，如果Object不存在，在客户端则收到false值，但是这时在服务器端实际上会产生一个404的请求信息。所以，这种业务场景下，出现404是正常的业务行为。
    - 客户端或其他进程以前删除了该对象。这种情况可以通过搜索logging功能记录的服务端日志信息中的相关对象操作即可确认。
    - 网络故障引起丢包重试。例如客户端发起一个删除操作删除某个Object，此时请求达到服务端，执行删除成功，但是响应在网络环境中丢包，然后客户端发起重试，第二次的删除操作可能就会遇到404错误。这种由于网络问题引起的404错误可以通过客户端日志和服务端日志确定：
      - 查看客户端应用日志是否出现重试请求。
      - 查看服务端日志是否对该Object有两次删除操作，前一次的删除操作HTTP Status为2xx。
  - 有效请求率低且客户端其他错误请求数高

    有效请求率为请求返回的HTTP状态码为2xx/3xx的请求数占总请求的比例。状态码为4XX和5XX范围内的请求将计为失败并降低该比例。客户端其他错误请求是指除服务端错误请求（5xx）、网络错误请求（499）、客户端授权错误请求（403）、客户端资源不存在错误请求（404）和客户端超时错误请求（408或者OSS错误码为RequestTimeout的400请求）这些错误请求之外的请求。

    您可以通过查看日志功能记录的服务端日志确定这些错误的具体类型，之后根据OSS错误响应码在客户端代码中查找具体原因进行解决。详情请参见OSS错误响应。

- 存储容量异常增加

  存储容量异常增加，如果不是上传类请求量增多，常见的原因应该是清理操作出现了问题，可以根据下面两个方面进行调查：

  - 客户端应用使用特定的进程定期清理来释放空间。针对这种请求的调查步骤是：
    a. 查看有效请求率指标是否下降，因为失败的删除请求会导致清理操作没能按预期完成。
    b. 定位请求有效率降低的具体原因，查看具体是什么错误类型的请求导致。然后还可以结合具体的客户端日志定位更详细的错误信息（例如，用于释放空间的STS临时Token已过期）。
  - 客户端通过设置LifeCycle来清理空间：针对这种请求，需要通过控制台或者API接口查看目前Bucket的LifeCycle是否为之前设置的预期值。如果不是，可以直接更正目前配置；进一步的调查可以通过Logging功能记录的服务端日志记录查询以前修改的具体信息。如果LifeCycle是正常的，但是却没有生效，请联系OSS系统管理员协助调查。

- 其他存储服务问题

  如果前面的故障排除章节未包括您遇到的存储服务问题，则可以采用以下方法来诊断和排查您的问题：

  i. 查看OSS监控服务，了解与预期的基准行为相比是否存在任何更改。监控视图可能能够确定此问题是暂时的还是永久性的，并可确定此问题影响哪些存储操作。
  ii. 使用监控信息来帮助您搜索日志功能记录的服务端日志数据，获取相关时间点发生的任何错误信息。此信息可能会帮助您排查和解决该问题。
  iii. 如果服务器端日志中的信息不足以成功排查此问题，则可以使用客户端日志来调查客户端应用程序，或者配合使用网络工具（Wireshark等）调查您的网络问题。

11.7. 常见问题

本文介绍在使用阿里云云监控产品监控 OSS 数据时遇到的一些常见问题及解决方案。

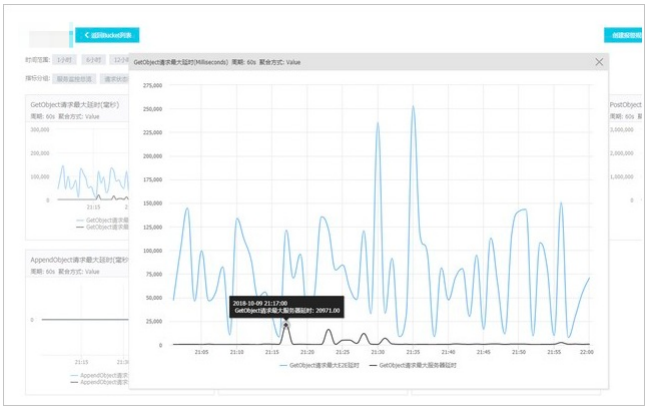
OSS 和云监控是两个独立的产品，OSS 将数据推送至云监控，由云监控产品进行分析处理。 OSS 控制台上看到的存储容量监控以及带宽流量监控来自于云监控产品的数据。

OSS 的数据推送到云监控会延迟 2-3 小时，同时云监控在接收 OSS 数据时存在窗口期，单次数据推送的时间间隔不能超过5分钟。如果OSS 推送数据超过5 分钟，那么云监控过会拒绝接收这个过期数据，同时也不支持补推。所以，不建议根据云监控的数据计算您的费用。如需核对费用，建议联系售后技术支持。

案例：报警规则的状态出现“数据不足”

问题分析：此问题可以查看**用户概况**的**服务监控总览**内的数据，若无数据产生，就会出现数据不足的情况。

案例：云监控上发现上传下载延迟



问题分析：云监控平台上查看到的数据是云监控产品节点发起探测请求获得的数据，并不代表真实用户环境。

解决方案：云监控平台监控到访问延迟较大的情况，可通过如下步骤排查：

1. 确认客户端访问是否真的有延迟。
2. 若用户访问对应的 Bucket 也出现延迟的情况，需通过抓包获取访问数据分析。
3. 您也可以通过日志分析对应时间内的访问数据，确认是否有访问延迟的情况。

案例：某公司自己的监控系统发现OSS请求数据有延迟

某公司因业务需求，自己搭建了一套监控系统监控 OSS 的数据，发现访问 OSS 延迟较大，可通过如下步骤排查：

1. 排查公司网络是否正常，可通过 ping 其他网站的形式测试延迟。
2. 在 OSS 同地域创建一个 ECS 服务器去访问 OSS 测试是否有延迟。
3. 将上传延迟的 OSS request ID 发送给售后技术支持，查看出现问题时访问是否真的延迟。
4. 通过抓包获取上传数据进行分析，可通过如下参数分析数据包：

```
tcpdump -i <出口网卡> -s0 ( 本机出口IP and OSS域名 ) -w result.pcap
```

案例：有效请求率降低

问题现象：云监控出现“对象存储 OSS (<)Bucket=p2xxx，userId=135114002(>)，有效请求率（30.51<90%），持续时间0分钟”的报错。

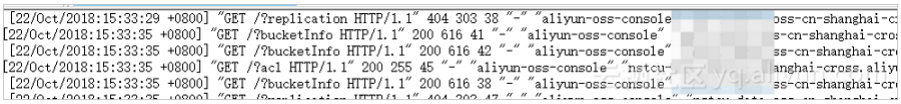
解决方案：异常请求率是通过 2xx+3xx/总体数量计算得出，您可以先查看云监控的 OSS 控制台统计的 2XX 3XX 以及其他异常状态码的占比，确认是否因异常状态码增加导致的有效请求率下降。您也可以通过开通 OSS 日志分析请求行为。

案例：云监控报警 404

问题现象：云监控出现“对象存储OSS实例：Bucket=***-ali，userId=197*****745，资源不存在错误请求数于11:45恢复正常，值为30次，持续时间5分钟”的报错。

问题分析：这种问题就是 Bucket 资源不存在导致的报警，属于正常的响应，并非是异常状态。

案例：云监控出现 NoSuchWebSiteConfigration



问题分析：此问题是客户端在请求 OSS 数据时加载的功能配置不存在，导致出现 404 的报错，200 的状态码是用户已经在 OSS 上配置的功能模块，并非异常现象。

案例：OSS 控制台 API 统计图无数据

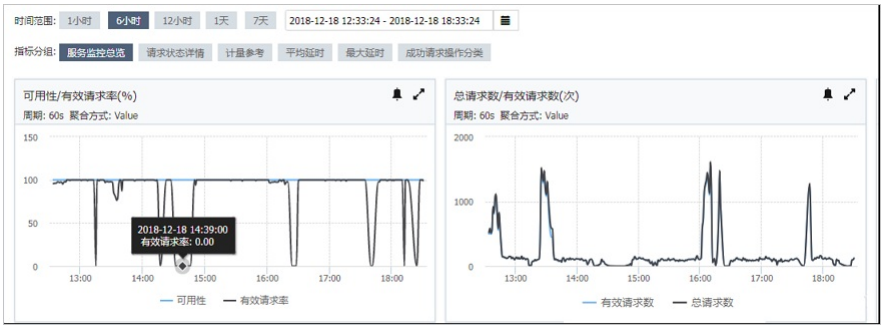
问题分析：API 的监控数据都是隔天显示，例如，今天是 10 月 12 日，完整的数据还没有出来，所以不能画点，需等到 13 号才能看到 12 号的完整数据。

案例：通过 OSS 监控计费核对账单发现数据不准确

OSS 的数据推送到云监控会延迟 1-2 小时，同时云监控在接收 OSS 数据时存在窗口期，单次数据推送的时间间隔不能超过5分钟。如果OSS 推送数据超过5 分钟，那么云监控过会拒绝接收这个过期数据，同时也不支持补推。所以，不建议使用云监控的数据和您的账号进行对账，因为数据并不准确，您可以通过以下方式对账：

- 提前开启 OSS 日志，之后通过 OSS 日志的统计和账单核对。
- 若您不愿意核对日志，可以开启 OSS 的日志分析功能，把 OSS 的日志导入通过日志分析处理后直接查看结果。

案例：云监控显示某个时间段的有效请求率下降为 0，但是 OSS 的 log 以及控制台的监控数据都是正常



问题分析：云监控有效请求率的计算公式是：100%-（2xx+3xx)/总请求数量。发现类似情况可查看 OSS 控制台或 OSS log 有没有异常即可。

出现这种问题是因为 OSS 将整个集群日志推送到云监控时超过了云监控的接收窗口期，而云监控不支持补推，所以导致数据为 0 。

12.事件通知

12.1. 事件通知概述

您可以在OSS管理控制台设置事件通知规则，自定义您关注的Object。当这些Object发生指定事件时，您可以及时收到通知。

前提条件

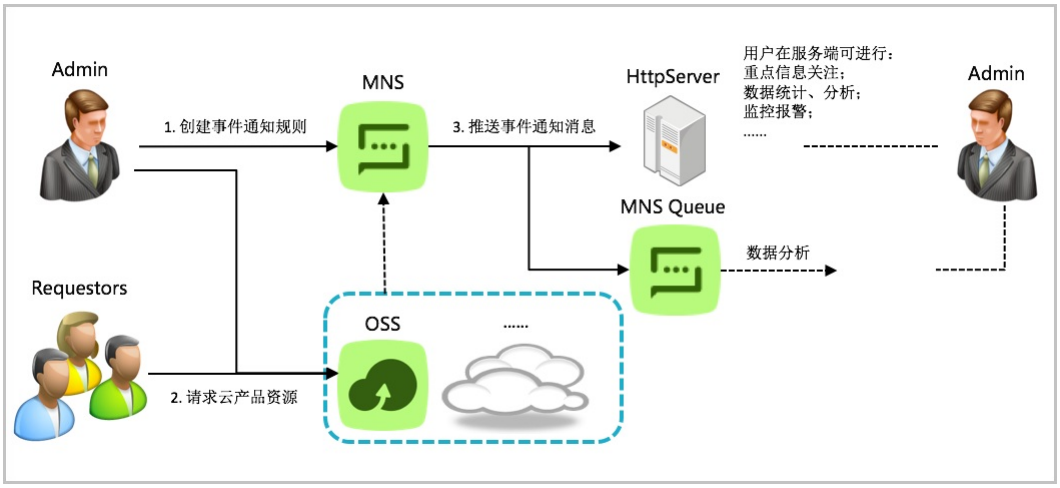
已开通消息服务MNS。您可以在MNS产品页开通MNS。

注意事项

- 使用事件通知功能会产生消息服务MNS的费用。计费详情，请参见价格说明。
- 华南2（河源）、华南3（广州）、华北5（呼和浩特）、华北6（乌兰察布）、阿联酋（迪拜）、马来西亚（吉隆坡）地域暂不支持事件通知功能。
- 同一地域仅支持配置10条事件通知规则。
- 通过RTMP推流方式生成的TS和M3U8文件不会触发事件通知规则。有关RTMP推流的介绍，请参见LiveChannel简介。

使用说明

您在创建事件通知规则后，若请求者对您OSS进行的操作触发了事件通知规则，消息服务MNS将请求者对OSS进行的相关操作发送到您配置的HTTP服务器或MNS的队列中。具体流程如下图所示：



事件类型

事件类型	说明
PutObject	通过简单上传创建或覆盖文件。
PostObject	通过表单上传创建或覆盖文件。
CopyObject	通过拷贝文件创建或覆盖文件。
InitiateMultipartUpload	初始化一个分片上传任务。
UploadPart	通过上传分片创建或覆盖文件。
UploadPartCopy	通过分片拷贝创建或覆盖文件。
CompleteMultipartUpload	完成分片上传。
AppendObject	通过追加上传创建或追加文件。
GetObject	通过简单下载获取文件。
DeleteObject	删除单个文件。
DeleteObjects	删除多个文件。
ObjectReplication:ObjectCreated	通过跨区域复制生成文件。
ObjectReplication:ObjectRemoved	通过跨区域复制删除文件。
ObjectReplication:ObjectModified	通过跨区域复制覆盖文件。
ObjectCreatedGroup	所有创建或覆盖文件操作。
ObjectDownloadedGroup	所有获取文件操作。
ObjectRemovedGroup	所有删除文件操作。

注意 ObjectCreatedGroup、ObjectDownloadedGroup、ObjectRemovedGroup三种事件类型目前仅支持在中国（香港）、美国（硅谷）、美国（弗吉尼亚）、德国（法兰克福）、澳大利亚（悉尼）、新加坡、英国（伦敦）地域配置。

消息通知

OSS的事件通知消息内容经过Base64编码，解码后是JSON格式，具体内容如下：

```
{"events": [
      {
        "eventName": "",  //事件通知类型。
        "eventSource": "", //设置事件通知的消息源，固定为"acs:oss"。
        "eventTime": "", //事件时间，使用ISO-8601时间表示法。
        "eventVersion": "", //事件通知版本号，目前为“1.0”。
        "oss": {
            "bucket": {
                "arn": "", //Bucket的唯一标识符，格式为"acs:oss:region:uid:bucketname"。
                "name": "", //目标Bucket名称。
                "ownerIdentity": "" //Bucket的拥有者。
            },
            "object": {
                "deltaSize": "", //Object大小的变化量。例如，新增一个文件，这个值就是文件大小；覆盖一个文件，这个值就是新文件与旧文件的大小差值，因此可能为负数。
                "eTag": "", //Object的ETag。
                "key": "", //Object名称。
                "position": "", //仅适用于ObjectCreated:AppendObject事件，表示此次请求开始追加的位置。首次AppendObject请求的位置从0字节开始。
                "readFrom": "", //仅适用于ObjectDownloaded:GetObject事件，表示文件开始读取的位置。对于非Range请求，此项为0；对于Range请求，此项为请求的开始字节。
                "readTo": "", //仅适用于ObjectDownloaded:GetObject事件，表示文件最后读取的位置。对于非Range请求，此项为文件的大小；对于Range请求，此项为Range请求的结束字节加1。
                "size": "" //Object大小。
                },
        "ossSchemaVersion": "", //此字段域的版本号，目前为“1.0”。
        "ruleId": "GetObject", //此事件匹配的规则ID。
        "region": "", //Bucket所在的地域。
        "requestParameters": {
            "sourceIPAddress": "" //请求的源IP地址。
            },
        "responseElements": {
            "requestId": "" //请求对应的Request ID。
            },
        "userIdentity": {
            "principalId": "" //请求发起者的UID。
            },
        "xVars": {  //OSS的回调功能（Callback）中的自定义参数。
            "x:callback-var1":"value1",
            "x:vallback-var2":"value2"
            }
        }
      }
  ]
}
```

消息通知示例：

```
{"events": [
      {
        "eventName": "ObjectDownloaded:GetObject",
        "eventSource": "acs:oss",
        "eventTime": "2016-07-01T11:17:30.000Z",
        "eventVersion": "1.0",
        "oss": {
            "bucket": {
                "arn": "acs:oss:cn-shenzhen:114895646818****:event-notification-test-shenzhen",
                "name": "event-notification-test-shenzhen",
                "ownerIdentity": "114895646818****"},
            "object": {
                "deltaSize": 0,
                "eTag": "0CC175B9C0F1B6468E1199E269772661",
                "key": "test",
                "readFrom": 0,
                "readTo": 1,
                "size": 1
            },
        "ossSchemaVersion": "1.0",
        "ruleId": "GetObjectRule",
        "region": "cn-shenzhen",
        "requestParameters": {
            "sourceIPAddress": "198.51.100.1"
            },
        "responseElements": {
            "requestId": "5FF16B65F05BC932307A3C3C"
            },
        "userIdentity": {
            "principalId": "114895646818****"
            },
        "xVars": {
            "x:callback-var1":"value1",
            "x:callback-var2":"value2"
            }
        }
      }
  ]
}
```


使用OSS控制台

1. 登录OSS管理控制台。
2. 单击左侧导航栏的**Bucket列表**，然后单击目标Bucket。
3. 选择左侧导航栏的**基础设置 > 事件通知**。
4. 单击**设置**，然后单击**创建规则**。
5. 在**创建规则**面板配置以下参数：

参数	说明
规则名称	设置事件通知规则的名称。 相同账号在同一地域下创建的规则名称不能重复。规则名称必须以英文字母开头，只能包含大小写字母、数字和短划线（-），长度不超过85个字符。
事件类型	为目标Object配置事件类型。例如，您希望接收到目标Object通过拷贝操作创建或覆盖文件的事件通知，请将事件类型配置为CopyObject。 您可以为目标Object配置一条事件通知规则，并在规则中指定多个事件类型。您还可以为目标Object配置多条事件通知规则。配置多条规则时，有如下注意事项： ○ 如果多条规则涉及的目标Object相同，则事件类型不允许相同。例如，规则A针对前缀 `images` 配置了CopyObject事件，如果规则B涉及 `images` 前缀下任意Object时，则事件类型不能包含CopyObject。 ○ 如果多条规则涉及的目标Object不同，则事件类型可以相同也可以不同。例如，规则A针对前缀为 `images` 、后缀为 `.png` 的Object配置了PutObject事件，如果规则B涉及的目标Object前缀为 `log` 、后缀为 `.jpg` ，则事件类型可以包含PutObject或者DeleteObject。 **注意** 对开启了版本控制的Bucket执行Object删除操作时，如果您未指定版本ID，不会触发DeleteObject或者DeleteObjects事件通知。原因是未指定版本ID的Object删除行为默认不会删除任意版本Object，而是将当前版本Object转为历史版本Object，并添加删除标记。 有关事件类型对应Object操作的更多信息，请参见事件类型。

参数	说明
资源描述	设置事件通知涉及的目标Object。 ○ 通过**全名**匹配目标Object ■ 如果要匹配Bucket根目录下名为exampleobject.txt的目标Object，则填写为*exampleobject.txt*。 ■ 如果要匹配Bucket根目录下destdir目录中名为myphoto.jpg的目标Object，则填写为*destdir/myphoto.jpg*。 ○ 通过**前后缀**匹配目标Object ■ 如果要匹配Bucket内的所有Object，则前缀和后缀均置空。 ■ 如果要匹配Bucket根目录下examplefolder目录中的所有Object，则前缀填写为*examplefolder/*，后缀置空。 ■ 如果要匹配Bucket内所有JPG格式的Object，则前缀置空，后缀填写为*.jpg*。 ■ 如需匹配Bucket根目录下*examplefolder*目录中所有MP3格式的Object，则前缀填写为*examplefolder/*，后缀填写为*.mp3*。 您可以单击**添加**按钮，创建最多5条**资源描述**。
接收终端	设置事件的接收终端，支持**HTTP**和**队列**。 ○ **HTTP**：填写接收事件通知的HttpEndpoint地址，例如 `http://198.51.100.1:8080` 。搭建HttpEndpoint的具体操作，请参见主题使用手册和HttpEndpoint。 ○ **队列**：填写您在MNS中创建的队列名称。创建队列的具体操作，请参见创建队列。 您可以单击**添加**按钮，创建最多5个**接收终端**。

6. 单击**确定**。

   以上步骤配置完成后，事件通知规则约10分钟后生效。

常见问题

- 问题描述：某个Bucket的事件通知规则中配置了DeleteObject以及DeleteObjects事件类型，但是当该Bucket出现Object删除行为时，并未触发事件通知。
- 问题原因：该Bucket开启了版本控制并且执行了未指定版本ID的Object删除操作。未指定版本ID的Object删除行为默认不会删除任意版本Object，而是将当前版本Object转为历史版本Object，并添加删除标记。因此，该场景下并不会触发DeleteObject或者DeleteObjects事件通知。

更多参考

您可以在事件通知规则中自定义您关注的Object，当这些Object发生指定事件时，您可以通过消息服务指定的接收终端，及时收到Object的事件通知。具体操作，请参见结合消息服务实现OSS事件通知的教程示例。

12.2. 教程示例：结合消息服务实现OSS事件通知

您可以通过OSS管理控制台配置事件通知规则，自定义您关注的文件（Object），当这些Object发生指定事件时，您可以通过消息服务指定的接收终端，及时收到Object的事件通知。

场景描述

某企业在华东1（杭州）地域创建了名为srcbucket的存储空间（Bucket），Bucket内包含了不同日期持续生成的以log为前缀的日志文件log/date1.txt、log/date2.txt和log/date3.txt等，以及以周为单位收集的以destdir为前缀的客户案例图片文件destdir/photo1.jpg、destdir/photo2.jpg等。

```
srcbucket
    └── log/
        ├── date1.txt
        ├── date2.txt
        ├── date3.txt
        ├── .......
    └── destdir/
        ├── photo1.jpg
        ├── photo2.jpg
        ├── ......
```

该企业子公司需要将srcbucket内以上持续生成的日志以及客户案例图片文件、以及这些文件在任意时间内产生的变化（例如文件的增、删、改操作）实时同步至母公司位于英国（伦敦）地域下名为destbucket的存储空间，并希望公司全员能第一时间内了解srcbucket以及destbucket内与前缀log以及destdir匹配的文件的变化情况。

为实现以上需求，您需要为srcbucket配置跨区域复制规则，同时还需要为srcbucket以及destbucket配置事件通知。

步骤一：创建队列

1. 登录消息服务MNS管理控制台。
2. 在左侧导航栏，单击**队列列表**。
3. 在顶部导航栏，选择**华东1（杭州）**地域。
4. 在**队列列表**页面，单击**创建队列**。
5. 在**创建队列**面板，队列名称设置为*myqueue1*，其他参数保持默认配置。
6. 单击**确定**。
7. 重复上述步骤在**英国（伦敦）**地域创建用于接收目标存储空间destbucket事件通知的队列*myqueue2*。

步骤二：为srcucket配置跨区域复制

1.
2. 单击左侧导航栏的**Bucket列表**，然后单击srcbucket。
3. 在左侧导航栏，选择**冗余与容错 > 跨区域复制**。
4. 在**跨区域复制**区域，单击**设置**
5. 单击**跨区域复制**，在**跨区域复制**面板配置以下参数。

参数	说明
目标地域	选择**英国（伦敦）**。
目标Bucket	选择**destbucket**。
加速类型	如果要提升中国内地srcbucket与非中国内地的destbucket之间跨区域复制时的传输速度，建议选择**传输加速**。开启传输加速功能，OSS还会额外收取传输加速费用。计费方式，请参见传输加速费用。
数据同步对象	选择**指定文件名前缀进行同步**，并添加前缀*destdir/*以及*log/*。
数据同步策略	选择**增/删/改同步**。
同步历史数据	选择**同步**。

6. 单击**确定**。

步骤三：为Bucket配置事件通知

> **注意** 因步骤二中跨区域复制规则中指定了同步历史数据，且需要同步的历史数据较多，会触发大量的消息。如果您不希望同步历史数据过程中触发消息，建议待历史数据同步完成后再开启事件通知。

1. 为源存储空间srcbucket配置事件通知。
   i. 登录OSS管理控制台。
   ii. 单击左侧导航栏的**Bucket列表**，然后单击srcbucket。
   iii. 选择左侧导航栏的**基础设置 > 事件通知**。
   iv. 单击**设置**，然后单击**创建规则**。
   v. 在**创建规则**面板按如下说明配置各项参数，然后单击**确定**。

参数	说明
规则名称	将事件通知规则名称设置*notification1*。
事件类型	选择**PutObject**、**CopyObject**、**DeleteObject**和**DeleteObjects**。
资源描述	选择**前后缀**，并依次添加前缀*log/*以及*destdir/*。
接收终端	选择**队列**，并填写步骤一中创建的队列名称*myqueue1*。

2. 为目标存储空间destbucket配置事件通知。
   i. 单击左侧导航栏的**Bucket列表**，然后单击destbucket。
   ii. 在**创建规则**面板按如下说明配置各项参数，然后单击**确定**。

参数	说明
规则名称	将事件通知规则名称设置*notification2*。
事件类型	选择**ObjectReplication:ObjectCreated**、**ObjectReplication:ObjectRemoved**和**ObjectReplication:ObjectModified**。
资源描述	选择**前后缀**，并依次添加前缀*log/*以及*destdir/*。
接收终端	选择**队列**，并填写步骤一中创建的队列名称*myqueue2*。

以上步骤配置完成后，事件通知规则约10分钟后生效。

步骤四：接收消息

当触发了事件匹配规则时，消息服务将自动创建主题（Topic），Topic名称格式为 `mns-en-topics-[Product]-[RuleName]-[Timestamp]` ，例如 `mns-en-topics-oss-notification1-96828124450125` 。此时，您需要为该Topic创建订阅，并在订阅规则中指定接收端地址。

1. 为Topic创建订阅。
   i. 登录消息服务MNS管理控制台。
   ii. 在左侧导航栏，单击**主题列表**。
   iii. 在顶部导航栏，选择**华东1（杭州）**地域。
   iv. 在自动创建的Topic右侧的操作栏下，单击**查看订阅**。
   v. 在目标Topic订阅页面，单击**订阅管理**，然后单击**创建订阅**。
   vi. 在**创建订阅**页面，为源存储空间srcbucket创建订阅。**名称**输入*mysubscription1*，**推送类型**选择**队列**，**接收端地址**输入*myqueue1*，其他参数保留默认配置。
   vii. 单击**确定**。
   viii. 重复上述步骤在**英国（伦敦）**地域为目标存储空间destbucket创建订阅*mysubscription2*，推送类型选择**队列**，接收端地址指定为*myqueue2*，其他参数保留默认配置。
2. 接收通知。
   i. 在左侧导航栏，单击**队列列表**。
   ii. 选择目标队列*myqueue1*右侧操作栏下的**更多 > 收发消息**。
   iii. 在**接收消息**区域，单击右上角的**接收消息**。
   此时，您将接收到源存储空间srcbucket内与前缀log以及destdir匹配的文件的增、删、改操作的事件通知。

iv. 重复上述步骤为目标存储空间dest bucket对应的队列*myqueue2*配置接收通知。配置完成后，您将接收到dest bucket内由于跨区域复制规则生成、覆盖或者删除文件的事件通知。

当您不再需要接收相关事件通知时，请及时删除相应的事件通知规则。但是，事件通知规则删除后，不会同步删除自动创建的Topic。为避免产生不必要的费用，请及时删除不再使用的Topic。

13.数据湖接入

13.1. OSS-HDFS服务概述

OSS-HDFS服务（JindoFS服务）是一款云原生数据湖存储产品。基于统一的元数据管理能力，在完全兼容HDFS文件系统接口的同时，提供充分的POSIX能力支持，能更好地满足大数据和AI等领域的数据湖计算场景。

功能优势

通过OSS-HDFS服务，无需对现有的Hadoop、Spark大数据分析应用做任何修改。通过简单的配置即可像在原生HDFS中那样管理和访问数据，同时获得OSS无限容量、弹性扩展、更高的安全性、可靠性和可用性支撑。

作为云原生数据湖基础，OSS-HDFS在满足EB级数据分析、亿级文件管理服务、TB级吞吐量的同时，全面融合大数据存储生态，除提供对象存储扁平命名空间之外，还提供了分层命名空间服务。分层命名空间支持将对象组织到一个目录层次结构中进行管理，并能通过统一元数据管理能力进行内部自动转换。对Hadoop用户而言，无需做数据复制或转换就可以实现像访问本地HDFS一样高效的数据访问，极大提升整体作业性能，降低了维护成本。

服务特性

OSS-HDFS服务支持的特性如下：

- OSS-HDFS服务完全兼容HDFS接口，同时支持目录层级的操作，您只需集成JindoSDK，即可为Apache Hadoop的计算分析应用（例如MapReduce、Hive、Spark、Flink等）提供了访问HDFS服务的能力，像使用Hadoop分布式文件系统（HDFS）一样管理和访问数据。具体操作，请参见OSS-HDFS服务快速入门。

HDFS兼容访问

- OSS-HDFS服务可以通过JindoFuse提供POSIX支持，允许您将OSS-HDFS服务上的文件挂载到本地文件系统中，能够像操作本地文件系统一样操作JindoFS服务中的文件。具体操作，请参见使用JindoFuse访问OSS-HDFS服务。

POSIX能力支持

- 使用自建Hadoop集群，严重依赖硬件资源，难以实现资源的弹性伸缩。例如Hadoop集群规模超过数百台，文件数达到4亿左右时，NameNode基本达到瓶颈。随着元数据规模的上涨，其QPS存在下降的趋势。

  OSS-HDFS服务是专为多租户、海量数据存储服务设计。元数据管理能力可弹性扩容，可支持更高的并发度、吞吐量和低时延。即使超过10亿文件数依然可以保持服务稳定，提供高性能、高可用的服务。同时采用统一元数据管理能力，可轻松应对超大文件规模，并支持多种分层分级策略，系统的资源利用率更高，更节约成本，能充分满足业务体量快速变化的需求。

高性能、高弹性、低成本

- OSS-HDFS服务中的数据存储在OSS上，而OSS作为阿里巴巴全集团数据存储的核心基础设施，多年支撑双11业务高峰，历经高可用与高可靠的严苛考验。OSS特性如下：
  - 服务可用性（或业务连续性）不低于99.995%。
  - 数据设计持久性不低于99.9999999999%（12个9）。
  - 规模自动扩展，不影响对外服务。
  - 数据自动多重冗余备份。

数据持久性和服务可用性

应用场景

OSS-HDFS服务提供全面的大数据和AI生态支持，其主要应用场景如下：

- OSS-HDFS服务原生支持文件、目录语义和操作，支持目录原子性、毫秒级rename操作，适用于开源Hive、Spark离线数仓。在ETL场景下相较于OSS标准存储类型Bucket，OSS-HDFS服务具有更大的性能优势。

Hive、Spark离线数仓

- OSS-HDFS服务提供append、truncate、flush、pwrite等基础文件操作。通过JindoFuse充分支持POSIX，可以在ClickHouse这类OLAP场景中替换本地磁盘来实现存储与计算分离方案。同时，得益于缓存系统进行加速，达到较优性价比。

OLAP

- OSS-HDFS服务提供append、truncate、flush、pwrite等基础文件操作。通过JindoFuse充分支持POSIX，无缝对接AI生态和已有的Python训练、推理程序。

AI 训练、推理

- OSS-HDFS服务原生支持文件、目录语义和操作，并支持flush操作，可用于替代HDFS用做HBase存储与计算分离方案。相比HBase结合OSS标准存储类型Bucket的方案，HBase结合OSS-HDFS服务不依赖HDFS来存放WAL日志，大幅简化整体方案架构。

HBase存储与计算分离

- OSS-HDFS服务高效支持flush和truncate操作，可无缝替代HDFS在Flink实时计算应用场景下用做Sink、Checkpoint存储方案。

实时计算

- OSS-HDFS服务作为新一代云原生数据湖存储，支持IDC HDFS平迁上云，优化HDFS使用体验，同时享受弹性伸缩、按需付费的成本效益，大幅优化存储成本。JindoDistCp工具支持将HDFS文件数据（包括文件属性等元数据）无缝迁入OSS-HDFS 服务，并基于HDFS Checksum提供快速比对。

数据迁移

功能特性

下表列举了OSS-HDFS在不同场景下的功能特性支持情况：

场景	支持特性
	原生支持文件、目录语义和操作
	添加文件、目录权限
	目录原子性、毫秒级rename

场景	支持特性
Hive、Spark数仓	通过setTimes设置时间
	扩展属性（XAttrs）
	ACL
	本地读缓存加速
替代HDFS	快照（Snapshot）
	文件flush、sync、truncate以及append操作
	校验和Checksum
	HDFS回收站自动清理
POSIX能力	文件随机写
	文件truncate、append、flush操作

13.2. OSS-HDFS服务快速入门

OSS-HDFS服务（JindoFS服务）完全兼容HDFS接口，同时支持目录层级的操作。JindoSDK为Apache Hadoop的计算分析应用（例如MapReduce、Hive、Spark、Flink等）提供了访问HDFS服务的能力。

前提条件

仅华东1（杭州）、华东2（上海）、华南1（深圳）、华北2（北京）和华北 3（张家口）地域支持在创建Bucket时开启HDFS服务，请联系技术支持申请试用。具体操作，请参见创建Bucket。

使用限制

- 仅支持在创建Bucket时开启HDFS服务，且开启后不支持关闭，请谨慎操作。
- 归档以及冷归档存储类型Bucket不支持开启HDFS服务。

步骤一：授权访问

1. 授权服务端管理已开通HDFS服务的Bucket。

   首次使用HDFS功能时，需要先在RAM控制台完成以下授权，以便OSS服务账号能够管理Bucket中的数据。

   i. 创建名为AliyunOSSDlsDefaultRole的角色。

      a. 登录RAM控制台。

      b. 在左侧导航栏，选择**身份管理 > 角色**。

      c. 单击**创建角色**，选择可信实体类型为**阿里云服务**，单击**下一步**。

      d. 角色类型选择**普通服务角色**，**角色名称**填写为*AliyunOSSDlsDefaultRole*，**选择受信服务**为**对象存储**。

      e. 单击**完成**。角色创建完成后，单击**关闭**。

   ii. 新建名为AliyunOSSDlsRolePolicy的自定义权限策略。

      a. 在左侧导航栏，选择**权限管理 > 权限策略**。

      b. 单击**创建权限策略**。

      c. 在**新建自定义权限策略**页面，填写**策略名称**为*AliyunOSSDlsRolePolicy*，配置模式选择**脚本配置**，并在**策略内容**中填写以下权限策略。

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "oss:ListObjects",
      "Resource": [
        "acs:oss:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "oss:*",
      "Resource": [
        "acs:oss:*:*:*/.dlsdata",
        "acs:oss:*:*:*/.dlsdata*"
      ]
    }
  ]
}
```

      d. 单击**确定**。

iii. 为角色授予自定义权限策略。

a. 在左侧导航栏，选择**身份管理 > 角色**。

b. 单击RAM角色*AliyunOSSDlsDefaultRole*右侧的**精确授权**。

c. 在**添加权限**页面，选择权限类型为**自定义策略**，输入已创建的自定义权限策略名称*AliyunOSSDlsRolePolicy*。

d. 单击**确定**。

2. 授权RAM用户访问已开通HDFS服务的Bucket。

i. 创建RAM用户。具体操作，请参见创建RAM用户。

ii. 创建自定义策略，策略内容如下：

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "oss:*",
            "Resource": [
                "acs:oss:*:*:*/.dlsdata",
                "acs:oss:*:*:*/.dlsdata*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:GetBucketInfo",
                "oss:PostDataLakeStorageFileOperation"
            ],
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

创建自定义策略的具体操作，请参见创建自定义权限策略。

iii. 为RAM用户授权已创建的自定义策略。具体步骤，请参见为RAM用户授权。

如果您使用服务角色（例如EMR服务角色AliyunEMRDefaultRole）访问已开通HDFS服务的Bucket，请参见上述为RAM用户授权的步骤完成服务角色授权。

步骤二：下载和安装JAR包

下载最新版本的JindoSDK JAR包。下载地址，请参见GitHub。

步骤三：配置环境变量

1. 配置 `JINDOSDK_HOME` 。

解压下载的安装包，以安装包内容解压在*/usr/lib/jindosdk-4.0.0*目录为例：

```
export JINDOSDK_HOME=/usr/lib/jindosdk-4.0.0
```

2. 配置 `HADOOP_CLASSPATH` 。

执行以下命令配置 `HADOOP_CLASSPATH` ：

```
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:${JINDOSDK_HOME}/lib/*
```

注意 请将安装目录和环境变量部署到所有所需节点上。

步骤四：配置JindoFS服务实现类及AccessKey

1. 将JindoSDK DLS实现类配置到Hadoop的core-site.xml中。

```
<configuration>
    <property>
        <name>fs.AbstractFileSystem.oss.impl</name>
        <value>com.aliyun.jindodata.oss.JindoOSS</value>
    </property>
    <property>
        <name>fs.oss.impl</name>
        <value>com.aliyun.jindodata.oss.JindoOssFileSystem</value>
    </property>
</configuration>
```

2. 将已开启HDFS服务的Bucket对应的accessKeyId、accessKeySecret预先配置在Hadoop的core-site.xml中。

```
<configuration>
    <property>
        <name>fs.oss.accessKeyId</name>
        <value>xxx</value>
    </property>
    <property>
        <name>fs.oss.accessKeySecret</name>
        <value>xxx</value>
    </property>
</configuration>
```

步骤五：配置JindoFS服务Endpoint

使用JindoFS服务访问OSS Bucket时需要配置Endpoint。推荐访问路径格式为 `oss://<Bucket>.<Endpoint>/<Object>` ，例如 `oss://examplebucket.cn-shanghai.oss-dls.aliyuncs.com/exampleobject.txt` 。配置完成后，JindoSDK会根据访问路径中的Endpoint访问对应的JindoFS服务接口。

您还可以通过其他方式配置JindoFS服务Endpoint，且不同方式配置的Endpoint存在生效优先级。更多信息，请参见配置Endpoint的其他方式。

步骤六：使用HDFS Shell访问OSS

以下列举了通过HDFS Shell命令对OSS执行的几种常见操作。

- 将本地根目录下的examplefile.txt文件上传至目标存储空间examplebucket，示例如下：

```
hdfs dfs -put examplefile.txt oss://examplebucket.<Endpoint>/
```

上传文件

- 在目标存储空间examplebucket下创建名为*dir/*的目录，示例如下：

```
hdfs dfs -mkdir oss://examplebucket.<Endpoint>/dir/
```

新建目录

- 查看目标存储空间examplebucket下的文件或目录信息，示例如下：

```
hdfs dfs -ls oss://examplebucket.<Endpoint>/
```

查看文件或目录信息

- 查看目标存储空间examplebucket下所有文件或目录的大小，示例如下：

```
hdfs dfs -du oss://examplebucket.<Endpoint>/
```

查看文件或目录大小

- 例如，您需要查看examplebucket下名为localfile.txt的文件内容，示例如下：

```
hdfs dfs -cat oss://examplebucket.<Endpoint>/localfile.txt
```

> **注意** 查看文件内容时，文件内容将以纯文本形式打印到屏幕上。如果文件内容进行了特定格式的编码，请使用HDFS的Java API读取并解码文件内容。

查看文件内容

- 例如，将目标存储空间examplebucket下根目录subdir1拷贝到目录subdir2下，且根目录subdir1所在的位置、根目录下的文件和子目录结构和内容保持不变，示例如下：

```
hdfs dfs -cp oss://examplebucket.<Endpoint>/subdir1  oss://examplebucket.<Endpoint>/subdir2/subdir2
```

拷贝目录或文件

- 例如，将目标存储空间根目录srcdir及其包含的文件或者子目录移动至另一个根目录destdir下，示例如下：

```
hdfs dfs -mv oss://examplebucket.<Endpoint>/srcdir  oss://examplebucket.<Endpoint>/destdir
```

移动目录或文件

- 例如，将目标存储空间examplebucket下的exampleobject.txt下载到本地根目录文件夹/tmp，示例如下：

```
hdfs dfs -get oss://examplebucket.<Endpoint>/exampleobject.txt  /tmp/
```

下载文件

```
hdfs dfs -rm oss://<bucket>.<Endpoint>/<path>
```

例如，删除目标存储空间examplebucket下*destfolder/*目录及其目录下的所有文件，示例如下：

```
hdfs dfs -rm oss://examplebucket.<Endpoint>/destfolder/
```

删除目录或文件

（可选）步骤七：性能调优

您可以结合实际业务需求，将以下配置项添加到Hadoop的core-site.xml中。仅JindoSDK 4.0及以上版本支持以下配置项。

```
<configuration>
    <property>
          <!-- 客户端写入的临时文件目录，可配置多个，每个临时文件目录需以逗号隔开。多用户环境需配置可读写权限 -->
        <name>fs.oss.tmp.data.dirs</name>
        <value>/tmp/</value>
    </property>
    <property>
          <!-- 访问OSS失败重试次数 -->
        <name>fs.oss.retry.count</name>
        <value>5</value>
    </property>
    <property>
          <!-- 请求OSS超时时间（毫秒） -->
        <name>fs.oss.timeout.millisecond</name>
        <value>30000</value>
    </property>
    <property>
          <!-- 连接OSS超时时间（毫秒） -->
        <name>fs.oss.connection.timeout.millisecond</name>
        <value>3000</value>
    </property>
    <property>
          <!-- OSS单个文件并发上传线程数 -->
        <name>fs.oss.upload.thread.concurrency</name>
        <value>5</value>
    </property>
    <property>
          <!-- OSS并发上传任务队列大小 -->
        <name>fs.oss.upload.queue.size</name>
        <value>5</value>
    </property>
    <property>
          <!-- 进程内OSS最大并发上传任务数 -->
        <name>fs.oss.upload.max.pending.tasks.per.stream</name>
        <value>16</value>
    </property>
    <property>
          <!-- OSS并发下载任务队列大小 -->
        <name>fs.oss.download.queue.size</name>
        <value>5</value>
    </property>
    <property>
          <!-- 进程内OSS最大并发下载任务数 -->
        <name>fs.oss.download.thread.concurrency</name>
        <value>16</value>
    </property>
    <property>
          <!-- 预读OSS的buffer大小 -->
        <name>fs.oss.read.readahead.buffer.size</name>
        <value>1048576</value>
    </property>
    <property>
          <!-- 同时预读OSS的buffer个数 -->
        <name>fs.oss.read.readahead.buffer.count</name>
        <value>4</value>
    </property>
</configuration>
```

附录：配置Endpoint的其他方式

除上述提到的在访问路径中指定Endpoint的方式以外，您还可以通过以下两种方式配置JindoFS服务的Endpoint：

- Bucket级别的Endpoint

  如果使用 `oss://<Bucket>/<Object>` 格式的访问路径，即访问路径中未设置Endpoint。此时，您可以在Hadoop的配置文件*core-site.xml*中设置Bucket级别的Endpoint，从而指向JindoFS服务的Endpoint。

```
<configuration>
    <property>
        <!-- 以下examplebucket为开通HDFS服务的Bucket名称，其他Bucket请根据实际情况替换。 -->
        <name>fs.oss.bucket.examplebucket.endpoint</name>
        <!-- 以下以杭州地域为例，其他地域请根据实际情况替换。 -->
        <value>cn-hangzhou.oss-dls.aliyuncs.com</value>
    </property>
</configuration>
```

- 全局默认Endpoint

  如果使用 `oss://<Bucket>/<Object>` 格式的访问路径，且访问路径中未设置Bucket级别的Endpoint，则默认使用全局Endpoint的方式访问JindoFS服务。在Hadoop的配置文件*core-site.xml*中设置全局默认Endpoint的方式如下：

```
<configuration>
    <property>
        <name>fs.oss.endpoint</name>
        <!-- 以下以杭州地域为例，其他地域请根据实际情况替换。 -->
        <value>cn-hangzhou.oss-dls.aliyuncs.com</value>
    </property>
</configuration>
```

说明 通过不同方式配置Endpoint后，Endpoint生效优先级为 `访问路径中的Endpoint>Bucket级别的Endpoint>全局默认Endpoint` 。

13.3. OSS-HDFS服务快照功能

OSS-HDFS服务（JindoFS服务）的快照功能在使用方式上与HDFS的快照功能完全兼容，同时支持目录层级的操作。本文介绍JindoFS服务快照功能的常见操作。

前提条件

OSS Bucket已开通HDFS服务。关于为Bucket开启HDFS服务的操作步骤，请参见创建存储空间。

开启快照功能

假设您拥有名为examplebucket的Bucket，且该Bucket已开启HDFS服务，并在该Bucket下创建了名为exampledir的目录。当您需要为该目录开启快照功能时，请通过JindoSDK的shell命令行工具执行如下命令：

```
jindo dlsadmin -allowSnapshot -dlsUri oss://examplebucket/exampledir
```

创建快照

在examplebucket下的exampledir目录开启快照功能后，在该目录下新建了子目录dir1以及dir2以及文件file1和file2：

```
# 新建子目录dir1。
hdfs dfs -mkdir oss://examplebucket/exampledir/dir1
# 新建子目录dir2。
hdfs dfs -mkdir oss://examplebucket/exampledir/dir2
# 新建文件file1。
hdfs dfs -touchz oss:oss://examplebucket/exampledir/file1
# 新建文件file2。
hdfs dfs -touchz oss://examplebucket/exampledir/file2
```

此时，您可以为exampledir目录创建快照，用于记录当前该目录下的子目录及文件结构。通过HDFS的shell命令行工具执行以下命令创建快照，并将快照命名为S1：

```
hdfs dfs -createSnapshot oss://examplebucket/exampledir S1
```

重命名快照

例如，您需要将已创建的快照S1重命名为S2，请通过HDFS的shell命令行工具执行以下命令：

```
hdfs dfs -renameSnapshot oss://examplebucket/exampledir S1 S2
```

访问快照中的目录和文件

当您需要访问examplebucket根目录exampledir的子目录dir1，请通过HDFS的shell命令行工具执行以下命令：

```
hdfs dfs -ls oss://examplebucket/exampledir/dir1
```

考虑到此前您已为examplebucket根目录exampledir创建了快照S1，因此访问快照S1的行为等同于访问根目录exampledir。请通过HDFS的shell命令行工具执行以下命令访问快照S1中的目录和文件：

```
hdfs dfs -ls oss://examplebucket/exampledir/.snapshot/S1/dir1
```

通过快照恢复数据

快照功能通常用于数据备份和恢复。通过快照功能，可以及时恢复误删除的数据。假设您误删除了examplebucket根目录exampledir下的文件dir1：

```
hdfs dfs -rm -r oss://examplebucket/exampledir/dir1
```

考虑到此前您已为examplebucket根目录exampledir创建了快照S1，此时您可以通过HDFS的shell命令行工具执行以下命令恢复误删除数据：

```
hdfs dfs -cp oss://examplebucket/exampledir/.snapshot/S1/dir1  oss://examplebucket/exampledir
```

完成数据恢复后，您可以通过以下命令查看误删除的文件夹或者文件：

```
hdfs dfs  -ls oss://examplebucket/exampledir/dir1
```

删除快照

例如，您不再需要保留examplebucket根目录exampledir创建的快照S1或者重命名后的快照S2，请通过HDFS的shell命令行工具执行如下命令删除快照S1以及S2。

- 删除快照S1

```
hdfs dfs -deleteSnapshot oss://examplebucket/exampledir S1
```

- 删除快照S2

```
hdfs dfs -deleteSnapshot oss://examplebucket/exampledir S2
```

关闭快照功能

当您不再需要使用快照功能时，请通过jindoSDK的shell命令行工具执行如下命令关闭快照功能：

```
jindo dlsadmin -disallowSnapshot -dlsUri oss://examplebucket/exampledir
```

注意 关闭快照功能前，请确保已删除目标路径下的所有快照。否则，关闭快照功能将会报错。