阿里云
对象存储 OSS

开发指南

文档版本：20180930

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
	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。	 禁止： 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 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。	`swich` *`{stand | slave}`*

目录

法律声明.....I
通用约定.....I
1 使用前须知.....1
2 基本概念介绍.....2
3 访问域名和数据中心.....6
4 数据容灾.....10
4.1 同城区域冗余存储.....10
5 访问与控制.....12
5.1 OSS访问域名使用规则.....12
5.2 ACL验证流程.....14
5.3 绑定自定义域名.....17
5.4 访问控制.....18
5.5 教程示例：控制存储空间和文件夹的访问权限.....29
6 接入OSS.....49
6.1 基于OSS的移动开发.....49
6.2 快速开始.....52
7 管理存储空间.....54
7.1 创建存储空间.....54
7.2 设置WORM策略.....54
7.3 请求者付费模式.....55
7.4 设置存储空间读写权限（ACL）.....57
7.5 查看存储空间列表.....58
7.6 获取存储空间信息.....58
7.7 删除存储空间.....58
8 上传文件.....59
8.1 简单上传.....59
8.2 表单上传.....60
8.3 分片上传和断点续传.....63
8.4 追加上传.....66
8.5 授权给第三方上传.....67
8.6 上传回调.....68
8.7 RTMP推流上传.....69
9 下载文件.....73
9.1 简单下载.....73

9.2 断点续传下载 73
9.3 授权给第三方下载 74
9.4 选取内容（OSS Select） 75
10 安全管理 76
10.1 设置访问日志记录 76
10.2 设置防盗链 79
10.3 设置跨域访问 80
10.4 服务器端加密编码 81
10.5 设置安全令牌 84
11 静态网站托管 89
11.1 配置静态网站托管 89
11.2 教程示例：使用自定义域名设置静态网站托管 90
12 云端数据处理 97

1 使用前须知

如果您初次使用阿里云OSS，请参阅阿里云*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*开发人员指南	本文档为您讲解阿里云OSS服务的核心概念、所有功能介绍与操作步骤，以及如何使用API和SDK的有效示例。
阿里云*OSS*最佳实践	详细介绍阿里云OSS的各种使用场景与配置实践。
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* 控制台用户指南	阿里云OSS管理控制台可让您通过界面执行OSS的部分功能。本文档为您介绍了基于阿里云OSS管理控制台的所有操作。
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。
阿里云*OSS*迁移工具	您可能需要将您本地或第三方云存储服务上的文件同步到阿里云OSS上，阿里云为此提供了完善的迁移解决方案。

2 基本概念介绍

本部分将向您介绍本产品中涉及的几个基本概念，以便于您更好地理解对象存储 OSS 产品。

存储空间（Bucket）

存储空间是您用于存储对象（Object）的容器，所有的对象都必须隶属于某个存储空间。您可以设置和修改存储空间属性用来控制地域、访问权限、生命周期等，这些属性设置直接作用于该存储空间内所有对象，因此您可以通过灵活创建不同的存储空间来完成不同的管理功能。

- 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、数字和短横线（-）。
- 必须以小写字母或者数字开头和结尾。
- 长度必须在3－63字节之间。

对象/文件（Object）

对象是 OSS 存储数据的基本单元，也被称为 OSS 的文件。对象由元信息（Object Meta），用户数据（Data）和文件名（Key）组成。对象由存储空间内部唯一的 Key 来标识。对象元信息是一个键值对，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，对象信息不可变更。重复上传同名的对象会覆盖之前的对象，因此，OSS 不支持修改文件的部分内容等操作。

OSS 提供了追加上传功能，用户可以使用该功能不断地在Object尾部追加写入数据。

对象的命名规范如下：

- 使用UTF-8编码。
- 长度必须在1－1023字节之间。
- 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。

Region（地域）

Region 表示 OSS 的数据中心所在的地域，物理位置。用户可以根据费用、请求来源等综合选择数据存储的 Region。一般来说，距离用户更近的 Region 访问速度更快。详情请查看*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，其中 AccessKeySecret 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 介绍请参见访问控制。

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 上同样具有强一致性，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，并且数据的三份副本已经写成功。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。

强一致性方便了用户架构设计，可以使用跟传统存储设备同样的逻辑来使用OSS，修改立即可见，无需考虑最终一致性带来的各种问题。

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，和a.jpg 这种并没有本质的区别。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	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。

对比项	OSS	文件系统
	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。	

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（目的和源相同）	修改文件属性
CopyObject	复制文件
无	重命名文件

3 访问域名和数据中心

OSS开通Region和Endpoint对照表

经典网络情况下各地域Endpoint的内外网设置如下：

Region中文名称	Region英文表示	外网Endpoint	外网支持HTTPS	ECS访问的内网Endpoint	内网支持HTTPS
华东 1	oss-cn-hangzhou	oss-cn-hangzhou.aliyuncs.com	是	oss-cn-hangzhou-internal.aliyuncs.com	是
华东 2	oss-cn-shanghai	oss-cn-shanghai.aliyuncs.com	是	oss-cn-shanghai-internal.aliyuncs.com	是
华北 1	oss-cn-qingdao	oss-cn-qingdao.aliyuncs.com	是	oss-cn-qingdao-internal.aliyuncs.com	是
华北 2	oss-cn-beijing	oss-cn-beijing.aliyuncs.com	是	oss-cn-beijing-internal.aliyuncs.com	是
华北 3	oss-cn-zhangjiakou	oss-cn-zhangjiakou.aliyuncs.com	是	oss-cn-zhangjiakou-internal.aliyuncs.com	是
华北 5	oss-cn-huhehaote	oss-cn-huhehaote.aliyuncs.com	是	oss-cn-huhehaote-internal.aliyuncs.com	是
华南 1	oss-cn-shenzhen	oss-cn-shenzhen.aliyuncs.com	是	oss-cn-shenzhen-internal.aliyuncs.com	是
香港	oss-cn-hongkong	oss-cn-hongkong.aliyuncs.com	是	oss-cn-hongkong-internal.aliyuncs.com	是

Region中文名称	Region英文表示	外网Endpoint	外网支持HTTPS	ECS访问的内网Endpoint	内网支持HTTPS
美国西部 1（硅谷）	oss-us-west-1	oss-us-west-1.aliyuncs.com	是	oss-us-west-1-internal.aliyuncs.com	是
美国东部 1（弗吉尼亚）	oss-us-east-1	oss-us-east-1.aliyuncs.com	是	oss-us-east-1-internal.aliyuncs.com	是
亚太东南 1（新加坡）	oss-ap-southeast-1	oss-ap-southeast-1.aliyuncs.com	是	oss-ap-southeast-1-internal.aliyuncs.com	是
亚太东南 2（悉尼）	oss-ap-southeast-2	oss-ap-southeast-2.aliyuncs.com	是	oss-ap-southeast-2-internal.aliyuncs.com	是
亚太东南 3（吉隆坡）	oss-ap-southeast-3	oss-ap-southeast-3.aliyuncs.com	是	oss-ap-southeast-3-internal.aliyuncs.com	是
亚太东南 5 (雅加达)	oss-ap-southeast-5	oss-ap-southeast-5.aliyuncs.com	是	oss-ap-southeast-5-internal.aliyuncs.com	是
亚太东北 1（日本）	oss-ap-northeast-1	oss-ap-northeast-1.aliyuncs.com	是	oss-ap-northeast-1-internal.aliyuncs.com	是
亚太南部 1（孟买）	oss-ap-south-1	oss-ap-south-1.aliyuncs.com	是	oss-ap-south-1-internal.aliyuncs.com	是
欧洲中部 1（法兰克福）	oss-eu-central-1	oss-eu-central-1.aliyuncs.com	是	oss-eu-central-1-internal.aliyuncs.com	是
英国（伦敦）	oss-eu-west-1	oss-eu-west-1.aliyuncs.com	是	oss-eu-west-1-internal.aliyuncs.com	是

Region中文名称	Region英文表示	外网Endpoint	外网支持HTTPS	ECS访问的内网Endpoint	内网支持HTTPS
中东东部 1（迪拜）	oss-me-east-1	oss-me-east-1.aliyuncs.com	是	oss-me-east-1-internal.aliyuncs.com	是

说明：

- 在分享链接或者做自定义域名绑定（CNAME）的时候建议使用三级域名，即`Bucket + Endpoint`的形式。以华东2地域内名为oss-sample的Bucket为例，三级域名为oss-sample.oss-cn-shanghai.aliyuncs.com。
- 使用SDK时，请将 `http://` 或 `https://` + `Endpoint` 作为初始化的参数。以华东2的Endpoint为例，建议将初始化参数设置为`http://oss-cn-shanghai.aliyuncs.com` 或者 `https://oss-cn-shanghai.aliyuncs.com`，不建议将三级域名（即`http://bucket.oss-cn-shanghai.aliyuncs.com`）作为初始化参数。
- 原地址oss.aliyuncs.com 默认指向华东1 地域外网地址。原内网地址oss-internal.aliyuncs.com 默认指向华东 1 地域内网地址。

VPC网络下Region和Endpoint对照表

在VPC网络下的ECS访问OSS可以使用如下的Endpoint：

Region中文名称	Region英文表示	VPC网络Endpoint	支持HTTPS
华东 1	oss-cn-hangzhou	oss-cn-hangzhou-internal.aliyuncs.com	是
华东 2	oss-cn-shanghai	oss-cn-shanghai-internal.aliyuncs.com	是
华北 1	oss-cn-qingdao	oss-cn-qingdao-internal.aliyuncs.com	是
华北 2	oss-cn-beijing	oss-cn-beijing-internal.aliyuncs.com	是
华北 3	oss-cn-zhangjiakou	oss-cn-zhangjiakou-internal.aliyuncs.com	是
华北 5	oss-cn-huhehaote	oss-cn-huhehaote-internal.aliyuncs.com	是

Region中文名称	Region英文表示	VPC网络Endpoint	支持HTTPS
华南 1	oss-cn-shenzhen	oss-cn-shenzhen-internal.aliyuncs.com	是
香港	oss-cn-hongkong	oss-cn-hongkong-internal.aliyuncs.com	是
美国西部 1（硅谷）	oss-us-west-1	oss-us-west-1-internal.aliyuncs.com	是
美国东部 1（弗吉尼亚）	oss-us-east-1	oss-us-east-1-internal.aliyuncs.com	是
亚太东南 1（新加坡）	oss-ap-southeast-1	oss-ap-southeast-1-internal.aliyuncs.com	是
亚太东南 2（悉尼）	oss-ap-southeast-2	oss-ap-southeast-2-internal.aliyuncs.com	是
亚太东南 3（吉隆坡）	oss-ap-southeast-3	oss-ap-southeast-3-internal.aliyuncs.com	是
亚太东南 5（雅加达）	oss-ap-southeast-5	oss-ap-southeast-5-internal.aliyuncs.com	是
亚太东北 1（日本）	oss-ap-northeast-1	oss-ap-northeast-1-internal.aliyuncs.com	是
亚太南部 1 (孟买)	oss-ap-south-1	oss-ap-south-1-internal.aliyuncs.com	是
欧洲中部 1（法兰克福）	oss-eu-central-1	oss-eu-central-1-internal.aliyuncs.com	是
英国（伦敦）	oss-eu-west-1	oss-eu-west-1-internal.aliyuncs.com	是
中东东部 1（迪拜）	oss-me-east-1	oss-me-east-1-internal.aliyuncs.com	是

4 数据容灾

4.1 同城区域冗余存储

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3AZ能够提供99.9999999999%（12个9）的数据可靠性以及99.99%的数据可用性。

OSS的同城区域冗余存储能够提供机房级容灾能力。当断网、断电或者发生灾难事件导致某个机房不可用时，仍然能够确保继续提供强一致性的服务能力，整个故障切换过程用户无感知，业务不中断、数据不丢失，可以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。

支持的地域

目前OSS的同城区域冗余存储支持的地域如下，后续面向更多地域开放：

- 华北 2（北京）
- 华东 2（上海）
- 华东 1（杭州）
- 华南 1（深圳）

支持的存储类型

目前OSS的同城区域冗余存储支持标准存储类型、低频访问存储类型。这两种存储类型的各项对比指标详情如下：

对比指标	标准存储类型	低频访问存储类型
数据可靠性	99.9999999999%	99.9999999999%
服务设计的可用性	99.99%	99.99%
对象最小计量大小	按照对象实际大小计算	64KB
最短存储时间	无最短存储时间要求	30天
数据取回费用	不收取数据取回费用	按实际获取的数据收取，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

修改属性

目前仅支持在创建Bucket时，设置同城区域冗余存储属性。不支持针对现有的Bucket开启同城区域冗余存储属性。

对于现有的Bucket，可以利用迁移工具，如OSSImport、SDK等方式，将原有Bucket中的数据复制到开启了OSS同城区域冗余属性的Bucket中。

功能使用参考

控制台：创建存储空间

5 访问与控制

5.1 OSS访问域名使用规则

OSS域名构成规则

针对OSS的网络请求，除了GetService这个API以外，其他所有请求的域名都是带有指定Bucket信息的三级域名组成的。

访问域名规则：BucketName.Endpoint。其中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，需要不同的访问域名。Endpoint分内网和外网访问域名。例如华东1 Region的外网Endpoint是oss-cn-hangzhou.aliyuncs.com，内网Endpoint是oss-cn-hangzhou-internal.aliyuncs.com，Region和Endpoint对照表请参考访问域名和数据中心。

通过外网访问OSS服务

这里的外网指的是互联网。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。

外网访问OSS有如下两种方式：

- 访问方式1：在访问的时候以URL的形式来表示OSS的资源。OSS的URL构成如下：

  ```
  <Schema>://<Bucket>.<外网Endpoint>/<Object>
  ```

  - Schema：HTTP或者为HTTPS
  - Bucket：OSS存储空间
  - Endpoint：Bucket所在数据中心的访问域名，您需要填写外网Endpoint
  - Object：上传到OSS上的文件

  示例：如您的Region为华东1（oss-cn-hangzhou），Bucket名称为abc，Object名称为myfile/aaa.txt，那么您的外网访问地址为：

  ```
  abc.oss-cn-hangzhou.aliyuncs.com/myfile/aaa.txt
  ```

  您还可以直接将Object的URL放入HTML中使用，如下所示：

  ```
  <img src="https://abc.oss-cn-hangzhou.aliyuncs.com/mypng/aaa.png" />
  ```

- 访问方式2： 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的Bucket进行操作时，需要在对类实例化时设置Endpoint：

```
String accessKeyId = "<key>";
  String accessKeySecret = "<secret>";
  String endpoint = "oss-cn-hangzhou.aliyuncs.com";
  OSSClient client = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);
```

通过内网访问OSS服务

内网指的是阿里云产品之间的内网通信网络，例如您通过ECS云服务器访问OSS服务。内网产生的流入和流出流量均免费。

内网访问OSS有如下两种方式：

- 访问方式1：在访问的时候以URL的形式来表示OSS的资源。OSS的URL构成如下。

  ```
  <Schema>://<Bucket>.<内网Endpoint>/<Object>
  ```

  — Schema：HTTP或者为HTTPS

  — Bucket：OSS存储空间

  — Endpoint：Bucket所在数据中心的访问域名，您需要填写内网Endpoint

  — Object：上传到OSS上的文件

  示例：如您在Region为华东1，Bucket名称为abc ，Object名称为myfile/aaa.txt，那么您的内网访问地址为：

  ```
  abc.oss-cn-hangzhou-internal.aliyuncs.com/myfile/aaa.txt
  ```

- 访问方式2：通过ECS使用OSS SDK配置内网访问域名。

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

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

  说明：

  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。

例如您购买了华北2（oss-cn-beijing）的ECS，其OSS有两个Bucket：

- 其中一个Bucket为beijingres，Region为华北2，那么在华北2的ECS中可以使用`beijingres.oss-cn-beijing-internal.aliyuncs.com`来访问 beijingres 的资源。
- 另外一个Bucket为qingdaores，Region为华北1，那么在华北2的ECS用内网地址`qingdaores.oss-cn-qingdao-internal.aliyuncs.com`是无法访问OSS的，必须使用外网地址`qingdaores.oss-cn-qingdao.aliyuncs.com`。

5.2 ACL验证流程

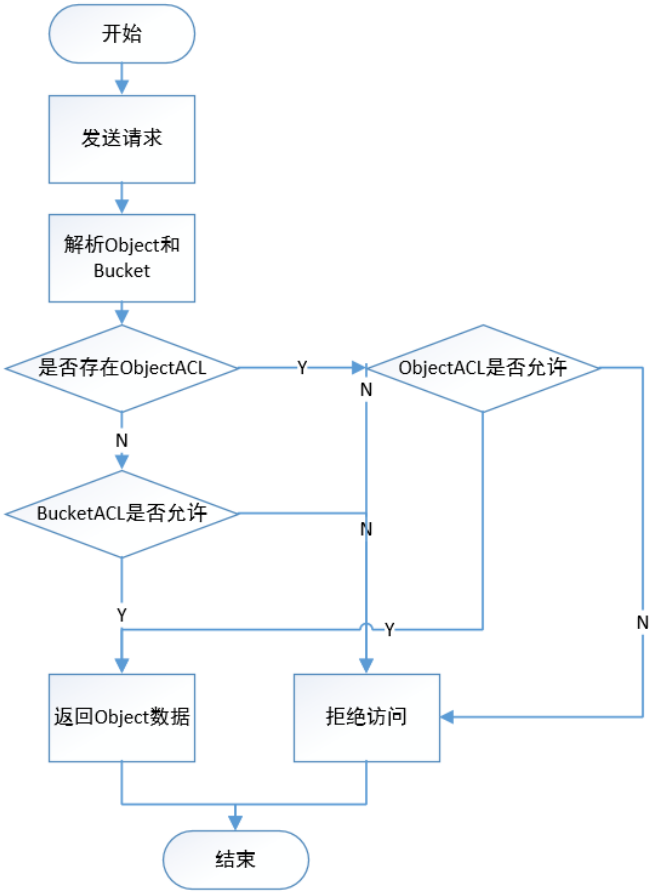
OSS访问的安全性

对OSS的HTTP请求可以根据是否携带身份验证信息分为两种请求。一种是带身份验证的请求，一种是不带身份验证的匿名请求。带身份验证的请求有如下两种情况：

- 请求头部中带Authorization，格式为OSS + AccessKeyId + 签名字符串。
- 请求的URL中带OSS AccessKeyId和Signature字段。

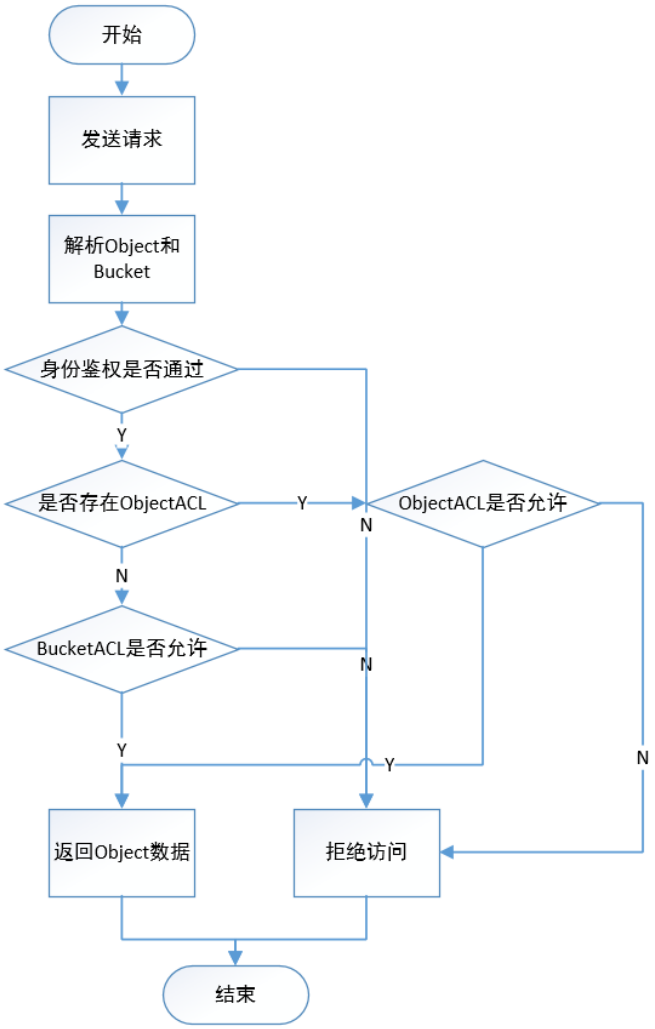
OSS访问验证流程

匿名请求的访问流程



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是否设置了ACL。
   - 如果没有设置ACL，那么继续4。
   - 如果设置了ACL，则判断Object的ACL是否允许匿名用户访问。
     - 允许则跳到5。
     - 不允许则拒绝请求，请求结束。
4. OSS判断Bucket的ACL是否允许匿名用户访问。
   - 允许则继续5。
   - 不允许则返回，请求结束。
5. 权限验证通过，返回Object的内容给用户。

带身份验证请求的访问流程



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，则返回，请求结束。
   - 如果获取成功，但请求者不被允许访问此资源，则返回，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，那么继续4。
4. OSS检查Object是否设置了ACL。
   - 如果Object没有设置ACL，那么继续5。
   - 如果Object设置了ACL，OSS判断Object的ACL是否允许此用户访问。
     - 允许则跳到6。

— 不允许则拒绝请求，请求结束。

5. OSS判断Bucket的ACL是否允许此用户访问。
   - 允许则继续6。
   - 不允许则返回，请求结束。
6. 权限验证通过，返回Object的内容给用户。

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。
- 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。
- 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。

关于AccessKey和身份验证的更多信息请参见访问控制。

5.3 绑定自定义域名

您的文件上传到OSS后，会自动生该文件的访问地址。您可以使用此地址访问OSS文件。如果您想要通过自定义域名访问OSS文件，需要将自定义域名访问绑定在文件所在的Bucket上，即CNAME。按照中国《互联网管理条例》的要求，所有需要开通这项功能的用户，必须提供工信部备案号，域名持有者身份证等有效资料，经由阿里云审批通过后才可以使用。在开通CNAME功能后，OSS将自动处理对该域名的访问请求。

应用场景

例如，用户A拥有一个域名为abc.com的网站，网站的网页中的链接为`http://img.abc.com/logo.png`。用户A此时需要将对网站中图片的请求迁移到OSS，并且不想修改任何网页的代码，也就是对外链接不变，CNAME功能特别适合这种场景。流程如下：

1. 用户A在OSS上创建一个名为abc-img的Bucket，并上传了其网站上的图片。
2. 用户A通过OSS控制台，提交将img.abc.com这个自定义的域名绑定在abc-img上的申请，并提供相应的材料。
3. 通过阿里云审核后，OSS后台会将img.abc.com做一个映射到abc-img（此处会做权限验证）。
4. 用户A在自己的域名服务器上，添加一条CNAME规则，将img.abc.com映射成abc-img.oss-cn-hangzhou.aliyuncs.com （即abc-img的OSS域名）。

5. `http://img.abc.com/logo.png`请求到达OSS后，OSS找到img.abc.com和abc-img的映射，转换成访问abc-img这个Bucket。也就是说，对`http://img.abc.com/logo.png`的访问，经过OSS后，实际上访问的是`http://abc-img.oss-cn-hangzhou.aliyuncs.com/logo.png`。

	CNAME绑定前	CNAME绑定后
流程对比	1. 访问`http://img.abc.com/logo.png`。 2. DNS解析到用户服务器 IP。 3. 访问用户服务器上的logo.png。	1. 访问`http://img.abc.com/logo.png` 。 2. DNS解析到abc-img.oss-cn-hangzhou.aliyuncs.com。 3. 访问OSS上 abc-img 里的 logo.png 。

功能使用参考

控制台：管理域名

5.4 访问控制

发送访问OSS的请求

您可以直接使用OSS提供的RESTful API接口访问或者使用对API接口进行完整封装的SDK开发包。而每一次向OSS的请求根据当前Bucket权限和操作不同要求用户进行身份验证或者直接匿名访问。对OSS的资源访问的分类如下：

- 按访问者的角色可分为拥有者访问和第三方用户访问。这里的拥有者指的是Bucket的Owner，也称为开发者。第三方用户是指访问Bucket里资源的用户。
- 按访问者的身份信息可分为匿名访问和带签名访问。对于OSS来说，如果请求中没有携带任何和身份相关的信息即为匿名访问。带签名访问指的是按照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。

AccessKey 类型

目前访问 OSS 使用的 AK（*AccessKey*）有三种类型，具体如下：

- 阿里云账号AccessKey

  阿里云账号AK特指Bucket拥有者的AK，每个阿里云账号提供的AccessKey对拥有的资源有完全的权限。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/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。

说明：

请避免直接使用阿里云账户的 AccessKey。

- RAM子账号AccessKey

  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AccessKey

  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。

对于用户来说可以直接使用OSS提供的SDK，配合不同类型的AccessKey即可实现不同的身份验证。

权限控制

针对存放在Bucket的Object的访问，OSS提供了多种权限控制，主要有：

- Bucket级别权限
- Object级别权限
- 账号级别权限（RAM）
- 临时账号权限（STS）

Bucket级别权限

- Bucket权限类型

  OSS提供ACL（Access Control List）权限控制方法，OSS ACL提供Bucket级别的权限访问控制，Bucket目前有三种访问权限：public-read-write，public-read和private，它们的含义如下：

权限值	中文名称	权限对访问者的限制
public-read-write	公共读写	任何人（包括匿名访问）都可以对该Bucket中的Object进行读/写/删除操作；所有这些操作产生的费用由该Bucket的Owner承担，请慎用该权限。
public-read	公共读，私有写	只有该Bucket的Owner或者授权对象可以对存放在其中的Object进行写/删除操作；任何人（包括匿名访问）可以对Object进行读操作。
private	私有读写	只有该Bucket的Owner或者授权对象可以对存放在其中的Object进行读/写/删除操作；其他人在未经授权的情况下无法访问该Bucket内的Object。

- Bucket权限设定和读取方法

  功能使用参考：

- API：*Put BucketACL*
- SDK: Java SDK - 设置*Bucket ACL*
- 控制台：创建*Bucket*权限设置
- API：*Get BucketACL*
- SDK：Java SDK - 获取*Bucket ACL*

Object级别权限

- Object权限类型

  OSS ACL也提供Object级别的权限访问控制。目前Object有四种访问权限：private, public-read, public-read-write, default。Put Object ACL操作通过Put请求中的“x-oss-object-acl”头来设置，这个操作只有Bucket Owner有权限执行。

权限值	中文名称	权限对访问者的限制
public-read-write	公共读写	该ACL表明某个Object是公共读写资源，即所有用户拥有对该Object的读写权限。
public-read	公共读，私有写	该ACL表明某个Object是公共读资源，即非Object Owner只有该Object的读权限，而Object Owner拥有该Object的读写权限。
private	私有读写	该ACL表明某个Object是私有资源，即只有该Object的Owner拥有该Object的读写权限，其他的用户没有权限操作该Object。
default	默认权限	该ACL表明某个Object是遵循Bucket读写权限的资源，即Bucket是什么权限，Object就是什么权限。

说明：

- 如果没有设置Object的权限，即Object的ACL为default，Object的权限和Bucket权限一致。

- 如果设置了Object的权限，Object的权限大于Bucket权限。举个例子，如果设置了Object的权限是public-read，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。

- Object权限设定和读取方法

  功能使用参考：

  - API：*Put Object ACL*
  - SDK：Java SDK - *ObjectACL* 中设定Object ACL
  - API：*Get Object ACL*
  - SDK：Java SDK - *ObjectACL* 中读取Object ACL

账号级别权限（**RAM**）

- 使用场景

  如果您购买了云资源，您的组织里有多个用户需要使用这些云资源，这些用户只能共享使用您的云账号AccessKey。这里有两个问题：

  - 您的密钥由多人共享，泄露的风险很高。
  - 您无法控制特定用户能访问哪些资源（比如Bucket）的权限。

  解决方法：在您的阿里云账号下面，通过RAM可以创建具有自己AccessKey的子用户。您的阿里云账号被称为主账号，创建出来的账号被称为子账号，使用子账号的AccessKey只能使用主账号授权的操作和资源。

- 具体实现

  有关RAM详情，请参考*RAM用户手册*。

  对于授权中需要的Policy的配置方式可以参考本章最后一节：RAM和STS授权策略（Policy）配置。

临时账号权限（**STS**）

- 使用场景

  对于您本地身份系统所管理的用户，比如您的App的用户、您的企业本地账号、第三方App，也有直接访问OSS资源的可能，将这部分用户称为联盟用户。此外，用户还可以是您创建的能访问您的阿里云资源的应用程序。

  对于这部分联盟用户，通过阿里云STS (Security Token Service) 服务为阿里云账号（或RAM用户）提供短期访问权限管理。您不需要透露云账号（或RAM用户）的长期密钥（如登录密码、

AccessKey），只需要生成一个短期访问凭证给联盟用户使用即可。这个凭证的访问权限及有效期限都可以由您自定义。您不需要关心权限撤销问题，访问凭证过期后会自动失效。

用户通过STS生成的凭证包括安全令牌(SecurityToken)、临时访问密钥(AccessKeyId, AccessKeySecret)。使用AccessKey方法与您在使用阿里云账户或RAM用户AccessKey发送请求时的方法相同。此外还需要注意的是在每个向OSS发送的请求中必须携带安全令牌。

- 具体实现

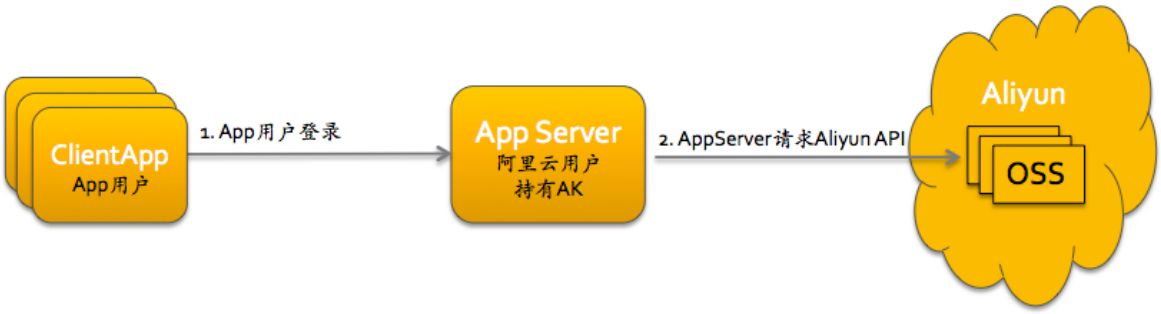
  STS安全令牌、角色管理和使用相关内容详情，请参考RAM用户指南中的角色管理。关键是调用STS服务接口*AssumeRole*来获取有效访问凭证即可，也可以直接使用STS SDK来调用该方法。

RAM和STS应用场景实践

对于不同的应用场景，涉及到的访问身份验证方式可能存在差异。下面以几种典型的应用场景来说明访问身份验证中几种使用方式。

以一个移动App举例。假设您是一个移动App开发者，打算使用阿里云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，防止一个App用户获取到其它App用户的数据。

- 方式一：使用AppServer来做数据中转和数据隔离

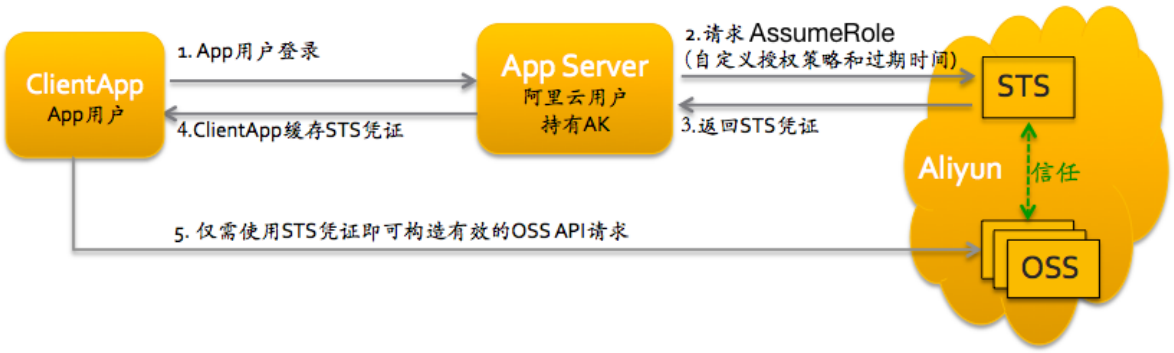


如上图所示，您需要开发一个AppServer。只有AppServer能访问云服务，ClientApp的每次读写数据都需要通过AppServer，AppServer来保证不同用户数据的隔离访问。

对于该种使用方式，使用阿里云账号或者RAM账号提供的密钥来进行签名验证访问。建议您尽量不要直接使用阿里云账号（主账号）的密钥访问OSS，避免出现安全问题。

- 方式二：使用STS让用户直接访问OSS

  STS方案描述如下图所示：



方案的详细描述如下：

1. App用户登录。App用户和云账号无关，它是App的终端用户，AppServer支持App用户登录。对于每个有效的App用户来说，需要AppServer能定义出每个App用户的最小访问权限。
2. AppServer请求STS服务获取一个安全令牌（SecurityToken）。在调用STS之前，AppServer需要确定App用户的最小访问权限（用Policy语法描述）以及授权的过期时间。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。
3. STS返回给AppServer一个有效的访问凭证，包括一个安全令牌（SecurityToken）、临时访问密钥（AccessKeyId, AccessKeySecret）以及过期时间。
4. AppServer将访问凭证返回给ClientApp。ClientApp可以缓存这个凭证。当凭证失效时，ClientApp需要向AppServer申请新的有效访问凭证。比如，访问凭证有效期为1小时，那么ClientApp可以每30分钟向AppServer请求更新访问凭证。
5. ClientApp使用本地缓存的访问凭证去请求Aliyun Service API。云服务会感知STS访问凭证，并会依赖STS服务来验证访问凭证，正确响应用户请求。

RAM和STS授权策略（Policy）配置

对于RAM或者STS授权中使用Policy，详细规则如下。

- 示例

  先看下面的一个Policy示例：

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": [
                "oss:GetBucketAcl",
                "oss:ListObjects"
            ],
            "Resource": [
                "acs:oss:*:1775305056529849: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:*:1775305056529849:mybucket/file*"
            ],
            "Effect": "Allow",
            "Condition": {
                "IpAddress": {
                    "acs:SourceIp": "192.168.0.1"
                }
            }
        }
    ]
}
```

这是一个授权的Policy，用户用这样的一个Policy通过RAM或STS服务向其他用户授权。Policy当中有一个Statement（一条Policy当中可以有多条Statement）。Statement里面规定了相应的Action、Resource、Effect和Condition。

这条Policy把用户自己名下的`mybucket`和`mybucket/file*`这些资源授权给相应的用户，并且支持GetBucketAcl、GetBucket、PutObject、GetObject和DeleteObject这几种操作。Condition中的条件表示UserAgent为java-sdk，源IP为192.168.0.1的时候鉴权才能通过，被授权的用户才能访问相关的资源。Prefix这个Condition是在GetBucket（ListObjects）的时候起作用的，关于这个字段的解释详见OSS的API文档。

- 配置细则
  - Version

    Version定义了Policy的版本，本文档中sw2q的配置方式，设置为1。
  - Statement

    通过Statement描述授权语义，其中可以根据业务场景包含多条语义，每条包含对Action、Effect、Resource和Condition的描述。每次请求系统会逐条依次匹配检查，所有匹配成功的Statement会根据Effect的设置不同分为通过（Allow）、禁止（Deny），其中禁止（Deny

）的优先。如果匹配成功的都为通过，该条请求即鉴权通过。如果匹配成功有一条禁止，或者没有任何条目匹配成功，该条请求被禁止访问。

- Action

Action分为三大类：Service级别操作，对应的是GetService操作，用来列出所有属于该用户的Bucket列表。

- Bucket级别操作，对应类似于oss:PutBucketAcl、oss:GetBucketLocation之类的操作，操作的对象是Bucket，它们的名称和相应的接口名称一一对应。
- Object级别操作，分为oss:GetObject、oss:PutObject、oss:DeleteObject和oss:AbortMultipartUpload，操作对象是Object。

如想授权某一类的Object的操作，可以选择这几种的一种或几种。另外，所有的Action前面都必须加上`oss:`，如上面例子所示。Action是一个列表，可以有多个Action。具体的Action和API接口的对应关系如下：

- Service级别

API	Action
GetService（ListBuckets）	oss:ListBuckets

- Bucket级别

API	Action
PutBucket	oss:PutBucket
GetBucket（ListObjects）	oss:ListObjects
PutBucketAcl	oss:PutBucketAcl
DeleteBucket	oss:DeleteBucket
GetBucketLocation	oss:GetBucketLocation
GetBucketAcl	oss:GetBucketAcl
GetBucketLogging	oss:GetBucketLogging
PutBucketLogging	oss:PutBucketLogging
DeleteBucketLogging	oss:DeleteBucketLogging
GetBucketWebsite	oss:GetBucketWebsite
PutBucketWebsite	oss:PutBucketWebsite
DeleteBucketWebsite	oss:DeleteBucketWebsite

API	Action
GetBucketReferer	oss:GetBucketReferer
PutBucketReferer	oss:PutBucketReferer
GetBucketLifecycle	oss:GetBucketLifecycle
PutBucketLifecycle	oss:PutBucketLifecycle
DeleteBucketLifecycle	oss:DeleteBucketLifecycle
ListMultipartUploads	oss:ListMultipartUploads
PutBucketCors	oss:PutBucketCors
GetBucketCors	oss:GetBucketCors
DeleteBucketCors	oss:DeleteBucketCors
PutBucketReplication	oss:PutBucketReplication
GetBucketReplication	oss:GetBucketReplication
DeleteBucketReplication	oss:DeleteBucketReplication
GetBucketReplicationLocation	oss:GetBucketReplicationLocation
GetBucketReplicationProgress	oss:GetBucketReplicationProgress

■ Object级别

API	Action
GetObject	oss:GetObject
HeadObject	oss:GetObject
PutObject	oss:PutObject
PostObject	oss:PutObject
InitiateMultipartUpload	oss:PutObject
UploadPart	oss:PutObject
CompleteMultipart	oss:PutObject
DeleteObject	oss:DeleteObject
DeleteMultipartObjects	oss:DeleteObject
AbortMultipartUpload	oss:AbortMultipartUpload
ListParts	oss:ListParts
CopyObject	oss:GetObject,oss:PutObject
UploadPartCopy	oss:GetObject,oss:PutObject

API	Action
AppendObject	oss:PutObject
GetObjectAcl	oss:GetObjectAcl
PutObjectAcl	oss:PutObjectAcl

- Resource

  Resource指代的是OSS上面的某个具体的资源或者某些资源（支持*通配），resource的规则是`acs:oss:{region}:{bucket_owner}:{bucket_name}/{object_name}`。对于所有Bucket级别的操作来说不需要最后的斜杠和{object_name}，即`acs:oss:{region}:{bucket_owner}:{bucket_name}`。Resource也是一个列表，可以有多个Resource。其中的region字段暂时不做支持，设置为*。

- Effect

  Effect代表本条的Statement的授权的结果，分为Allow和Deny，分别指代通过和禁止。多条Statement同时匹配成功时，禁止（Deny）的优先级更高。

  例如，期望禁止用户对某一目录进行删除，但对于其他文件有全部权限：

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:*"
      ],
      "Resource": [
        "acs:oss:*:*:bucketname"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "oss:DeleteObject"
      ],
      "Resource": [
        "acs:oss:*:*:bucketname/index/*",
      ]
    }
  ]
}
```

- Condition

  Condition代表Policy授权的一些条件，上面的示例里面可以设置对于acs:UserAgent的检查、acs:SourceIp的检查、还有oss:Prefix这项用来在GetBucket的时候对资源进行限制。

OSS支持的Condition如下：

condition	功能	合法取值
acs:SourceIp	指定ip网段	普通的ip，支持*通配
acs:UserAgent	指定http useragent头	字符串
acs:CurrentTime	指定合法的访问时间	ISO8601格式
acs:SecureTransport	是否是https协议	“true”或者”false”
oss:Prefix	用作ListObjects时的prefix	合法的object name

更多示例

针对具体场景更多的授权策略配置示例，可以参考教程示例#控制存储空间和文件夹的访问权限和*OSS*授权常见问题。

Policy在线图形化便捷配置工具，请单击这里。

最佳实践

- *RAM*和*STS*使用指南

5.5 教程示例：控制存储空间和文件夹的访问权限

本教程示例详细演示了如何控制用户对 OSS 存储空间和文件夹的访问。在示例中，我们首先创建一个存储空间和文件夹，然后使用阿里云主账号创建访问管理 (RAM) 用户，并为这些用户授予对所创建 OSS 存储空间及文件夹的增量权限。

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。用户根据其唯一的key（对象名）检索对象的内容。例如，名为 **example-company** 的存储空间有三个文件夹：**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。注意，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：
> 当控制台列出**example-company**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。控制台的响应与文件系统类似，会显示文件夹列表。上例说明，存储空间**example-company**有三个对象，其key分别为`Development/Alibaba Cloud.pdf`、`Development/ProjectA.docx`及`Development/ProjectB.docx`。

控制台通过对象的key推断逻辑层次结构。当您创建对象的逻辑层次结构时，您可以管理对个别文件夹的访问，如本教程后面描述的那样。

在本教程开始之前，您还需要知道“根级”存储空间内容的概念。假设**example-company**存储空间包含以下对象：

- Development/Alibaba Cloud.pdf
- Development/ProjectA.docx
- Development/ProjectB.docx
- Marketing/data2016.xlsx
- Marketing/data2016.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（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

在授予权限之前，我们需要清楚，当用户单击某个存储空间的名字时控制台向 OSS 发送的是什么请求、OSS 返回的是什么响应，以及控制台如何解析该响应。

当用户单击某个存储空间名时，控制台会将*GetBucket* 请求发送至 OSS。此请求包括以下参数：

- `prefix`，其值为空字符串。
- `delimiter`，其值为`/`。

请求示例如下所示：

```
GET /?prefix=&delimiter=/ HTTP/1.1
Host: example-company.oss-cn-hangzhou.aliyuncs.com
```

```
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:DNrnx7xHk3sgysx7I8U9
I9IY1vY=
```

OSS 返回的响应包括ListBucketResult元素：

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 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>example-company</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>
```

由于 oss-dg.pdf 不包含/分隔符，因此 OSS 在<Contents/>元素中返回该key。存储空间 example-company 中的所有其他key都包含/分隔符，因此 OSS 会将这些key分组，并为每个前缀值 Development/、Marketing/ 和 Private/ 返回一个<CommonPrefixes/>元素。该元素是一个字符串，包含从这些key的第一个字符开始到第一次出现指定的/分隔符之间的字符。

控制台会解析此结果并显示如下的根级别项：

现在，如果用户单击**Development**文件夹，控制台会将*GetBucket*请求发送至 OSS。此请求包括以下参数：

- `prefix`，其值为`Development/`。
- `delimiter`，其值为`/`。

请求示例如下所示：

```
GET /?prefix=Development/&delimiter=/ HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:DNrnx7xHk3sgysx7I8U9
I9IY1vY=
```

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: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 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>example-company</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>
</ListBucketResult>
```

控制台会解析此结果并显示如下的key：

教程示例

本教程示例如下所示：

- 创建一个存储空间example-company，然后向其中添加三个文件夹（Development、Marketing 和 Private）。
- 您有Anne和Leo两个用户。您希望Anne只能访问**Development**文件夹而Leo则只能访问**Marketing**文件夹，并且希望将**Private**文件夹保持私有。在教程示例中，通过创建访问控制 (RAM) 用户（Anne和Leo）来管理访问权限，并授予他们必要的权限。
- RAM 还支持创建用户组并授予适用于组中所有用户的组级别权限。这有助于更好地管理权限。在本示例中，Anne和Leo都需要一些公共权限。因此，您还要创建一个名为Staff 的组，然后将

Anne和Leo添加到该组中。首先，您需要给该组分配策略授予权限。然后，将策略分配给特定用户，添加特定用户的权限。

> 说明：
> 本教程示例使用example-company作为存储空间名、使用Anne和Leo作为RAM用户名并使用Staff作为组名。由于阿里云OSS要求存储空间名全局唯一，所以您需要用自己的存储空间名称替换本教程中的存储空间名。

示例准备

本示例使用阿里云主账号创建RAM用户。最初，这些用户没有任何权限。您将逐步授予这些用户执行特定 OSS 操作的权限。为了测试这些权限，您需要使用每个用户的RAM账号登录到控制台。当您作为主账号所有者逐步授予权限并作为RAM用户测试权限时，您需要每次使用不同账号进行登录和注销。您可以使用一个浏览器来执行此测试。如果您可以使用两个不同的浏览器，则该测试过程用时将会缩短：一个浏览器用于使用主账号连接到阿里云控制台，另一个浏览器用于使用RAM 账号进行连接。

要使用您的主账号登录到阿里云控制台。 RAM用户不能使用相同的链接登录。他们必须使用RAM用户登录链接。作为主账号所有者，您可以向RAM用户提供此链接。

> 说明：
> 有关 RAM 的详细信息，请参见使用 *RAM* 用户账号登录。

为 RAM 用户提供登录链接

1. 使用主账号登录*RAM*控制台。
2. 在左侧导航栏中，单击**概览**。
3. 在 RAM用户登录链接 后找到URL。您将向RAM用户提供此URL，以便其使用RAM用户名和密码登录控制台。

步骤 1：创建存储空间

在此步骤中，您可以使用主账号登录到OSS控制台、创建存储空间、将文件夹（Development、Marketing、Private）添加到存储空间中，并在每个文件夹中上传一个或两个示例文档。

1. 使用主账号登录 *OSS* 控制台。
2. 创建名为**example-company** 的存储空间。

   有关详细过程，请参见OSS 控制台用户指南中的创建存储空间 。

3. 将一个文件上传到存储空间中。

   本示例假设您将文件 oss-dg.pdf 上传到存储空间的根级别。您可以用不同的文件名上传自己的文件。

   有关详细过程，请参见 OSS 控制台用户指南中的上传文件。

4. 添加名为 Development、Marketing 和 Private 的三个文件夹。

   有关详细过程，请参见 OSS 控制台用户指南 中的创建文件夹。

5. 将一个或两个文件上传到每个文件夹中。

   本例假设您将具有以下对象键的对象上传到存储空间中：

   - Development/Alibaba Cloud.pdf
   - Development/ProjectA.docx
   - Development/ProjectB.docx
   - Marketing/data2016.xlsx
   - Marketing/data2016.xlsx
   - Private/2017/images.zip
   - Private/2017/promote.pptx
   - oss-dg.pdf

步骤 2：创建 RAM 用户和组

在此步骤中，您使用 RAM 控制台将两个 RAM 用户 Anne 和 Leo 添加到主账号中。您还将创建一个名为 Staff 的组，然后将这两个用户添加到该组中。

> 说明：
> 在此步骤中，不要分配任何授予这些用户权限的策略。在以下步骤中，您将逐步为其授予权限。

有关创建 RAM 用户的详细过程，请参见 RAM 快速入门中的创建 *RAM* 用户。请为每个 RAM 用户创建登录密码。

有关创建组的详细过程，请参见 RAM 用户指南中的创建组。

步骤 3：确认 RAM 用户没有任何权限

如果您使用两个浏览器，现在可以在另一个浏览器中使用其中一个 RAM 用户账号登录到控制台。

1. 打开 RAM 用户登录链接，并用 Anne 或 Leo 的账号登录到 RAM 控制台。

2. 打开 OSS 控制台。

   您发现控制台中没有任何存储空间，这意味着 Anne 不具有对存储空间 example-company 的任何权限。

步骤 4：授予组级别权限

我们希望 Anne 和 Leo 都能执行以下操作：

- 列出主账号所拥有的所有存储空间。

  为此，Anne 和 Leo 必须具有执行 oss:ListBuckets 操作的权限。

- 列出 example-company 存储空间中的根级别项、文件夹和对象。

  为此，Anne 和 Leo 必须具有对 example-company 存储空间执行 oss:ListObjects 操作的权限。

步骤 4.1.授予列出所有存储空间的权限

在此步骤中，创建一个授予用户最低权限的策略。凭借最低权限，用户可列出主账号所拥有的所有存储空间。您还将此策略分配给 Staff 组，以便授予获得主账号拥有的存储空间列表的组权限。

1. 使用主账号登录*RAM*控制台。
2. 创建策略 **AllowGroupToSeeBucketListInConsole**。
   a. 在左侧导航窗格中，单击策略管理，然后单击新建授权策略。
   b. 单击空白模板。
   c. 在授权策略名称 字段中，输入**AllowGroupToSeeBucketListInConsole**。
   d. 在策略内容 字段中，复制并粘贴以下策略。

```
{
"Version": "1",
"Statement": [
 {
   "Effect": "Allow",
   "Action": [
     "oss:ListBuckets"
   ],
   "Resource": [
     "acs:oss:*:*:*"
   ]
 }
]
}
```

说明：

策略为 JSON 文档。在该策略中，Statement 是一个对象数组，每个对象使用名/值对的集合来描述权限。前面的策略描述了一个特定的权限。Effect 元素值决定是允许还是拒绝特定的权限。Action 指定访问权限的类型。在本策略中，oss:ListBuckets 是预定义的 OSS 操作，可返回经过身份验证的发送者所拥有的所有储存空间的列表。

3. 将 **AllowGroupToSeeBucketListInConsole** 策略分配给 Staff 组。

   有关分配策略的详细过程，请参见 RAM 快速入门 中将策略分配给 *RAM* 用户的将策略分配给 **RAM** 组。

   可以将策略分配给 RAM 控制台中的 RAM 用户和组。在本例中，我们将策略分配给组，因为我们希望 Anne 和 Leo 都能够列出这些存储空间。

4. 测试权限。

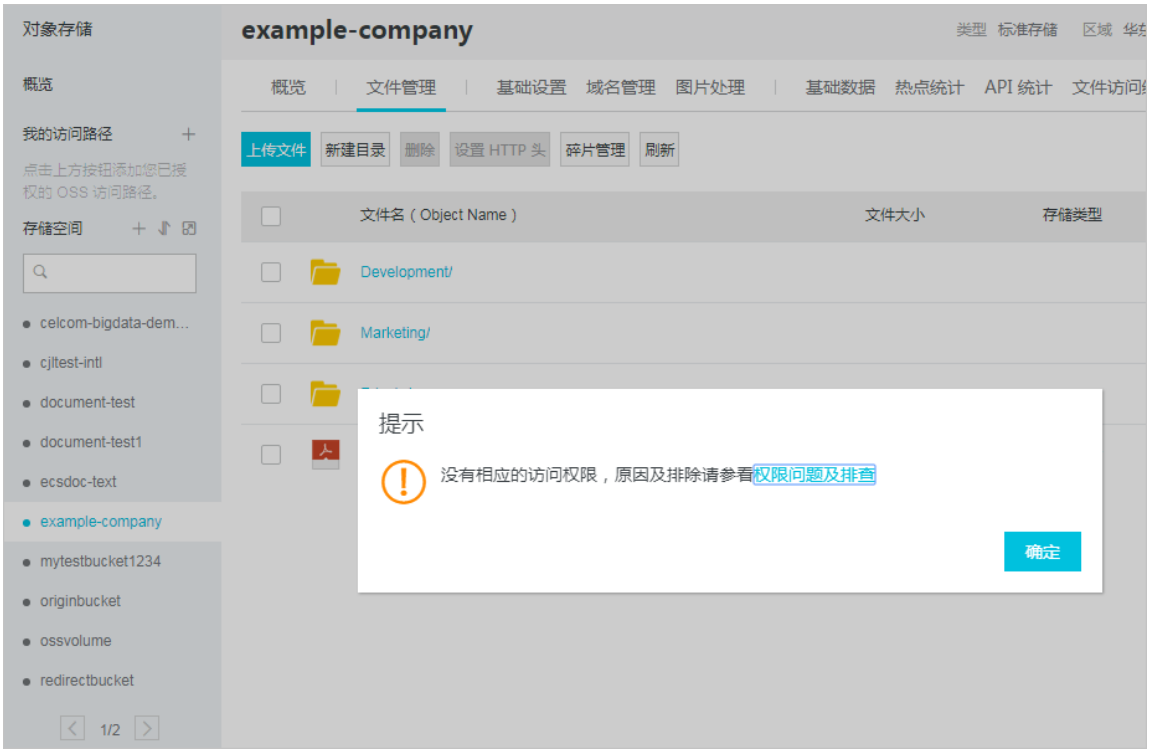
   a. 打开 RAM 用户登录链接，并用 Anne 或 Leo 的账号登录到 RAM 控制台。

   b. 打开 OSS 控制台。

      控制台列出所有存储空间。

   c. 单击 example-company 存储空间，然后单击文件选项卡。

      此时将显示一个消息框，表明您没有相应的访问权限。



步骤 4.2.授予列出存储空间根级内容的权限

在此步骤中，您授予权限，允许所有用户列出存储空间 example-company 中的所有项目。当用户在 OSS 控制台中单击 example-company 时，能够看到存储空间中的根级别项。

1. 使用主账号登录*RAM*控制台。
2. 用以下策略取代分配给 Staff 组的现有策略**AllowGroupToSeeBucketListInConsole**，该策略还允许 oss:ListObjects 操作。请用您的存储空间名替换策略资源中的 example-company。

   有关详细过程，请参见 RAM 用户指南中授权策略的修改自定义授权策略部分。注意，您最多可对 RAM 策略进行五次修改。如果超过了五次，则需要删除该策略并创建一个新的策略，然后再次将新策略分配给 Staff 组。

```
{
   "Version": "1",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "oss:ListBuckets",
         "oss:GetBucketAcl"
       ],
       "Resource": [
         "acs:oss:*:*:*"
       ],
       "Condition": {}
     },
     {
       "Effect": "Allow",
```

```
        "Action": [
          "oss:ListObjects"
        ],
        "Resource": [
          "acs:oss:*:*:example-company"
        ],
        "Condition": {
          "StringLike": {
            "oss:Prefix": [
              ""
            ],
            "oss:Delimiter": [
              "/"
            ]
          }
        }
      }
    ]
}
```

说明：

- 要列出存储空间内容，用户需要调用 oss:ListObjects 操作的权限。为了确保用户仅看到根级内容，我们添加了一个条件：用户必须在请求中指定一个空前缀，也就是说，他们不能单击任何根级文件夹。我们还通过要求用户请求包含分隔符参数和值 / 来添加需要文件夹样式访问的条件。
- 当用户登录到 OSS 控制台时，控制台检查用户的身份是否有访问 OSS 服务的权限。要在控制台中支持存储空间操作，我们还需要添加 oss:GetBucketAcl 操作。

3. 测试更新的权限。

   a. 打开 RAM 用户登录链接，并用 Anne 或 Leo 的账号登录到 RAM 控制台。

   b. 打开 OSS 控制台。

      控制台列出所有存储空间。

   c. 单击 example-company 存储空间，然后单击文件选项卡。

      控制台列出所有根级别项。

d. 单击任何文件夹或对象**oss-dg.pdf**。

此时将显示一个消息框，表明您没有相应的访问权限。

组策略摘要

添加组策略的最终结果是授予 RAM 用户 Anne 和 Leo 以下最低权限：

- 列出主账号所拥有的所有存储空间。
- 查看 example-company 存储空间中的根级别项。

然而，他们可以进行的操作仍然有限。在以下部分中，我们将授予用户以下特定权限：

- 允许 Anne 在 Development 文件夹中获取和放入对象。
- 允许 Bob 在 Finance 文件夹中获取和放入对象。

对于用户特定的权限，您需要将策略分配给特定用户，而非分配给组。以下部分授予 Anne 在 Development 文件夹中操作的权限。您可以重复这些步骤，授予 Leo 在 Finance 文件夹中进行类似操作的权限。

步骤 5：授予 RAM 用户 Anne 特定权限

在此步骤中，我们向 Anne 授予额外的权限，使她可以看到 Development 文件夹的内容，并将对象放入文件夹中。

步骤 5.1.授予 RAM 用户 Anne 权限以列出 Development 文件夹内容

若要 Anne 能够列出 Development 文件夹内容，您必须为其分配策略。该策略必须能够授予其对 example-company 存储空间执行 oss:ListObjects 操作的权限，还必须包括要求用户在请求中指定前缀 `Development/` 的条件。

1. 使用主账号登录*RAM*控制台。
2. 创建策略**AllowListBucketsIfSpecificPrefixIsIncluded**，授予 RAM 用户 Anne 权限以列出 Development 文件夹内容。
   a. 在左侧导航窗格中，单击策略管理，然后单击新建授权策略。
   b. 单击空白模板。
   c. 在授权策略名称字段中，输入**AllowListBucketsIfSpecificPrefixIsIncluded**。
   d. 在策略内容 字段中，复制并粘贴以下策略。

```
{
"Version": "1",
"Statement": [
 {
   "Effect": "Allow",
   "Action": [
     "oss:ListObjects"
   ],
   "Resource": [
     "acs:oss:*:*:example-company"
   ],
   "Condition": {
     "StringLike": {
       "oss:Prefix": [
         "Development/*"
       ]
     }
   }
 }
]
}
```

3. 将策略分配给 RAM 用户 Anne。

   有关分配策略的详细过程，请参见 RAM 快速入门中的将策略分配给 *RAM* 用户
4. 测试 Anne 的权限。
   a. 打开 RAM 用户登录链接，并用 Anne 的账号登录到 RAM 控制台。

b. 打开 OSS 控制台，控制台列出所有存储空间。

c. 单击 example-company 存储空间，然后单击文件选项卡，控制台列出所有根级别项。

d. 单击 Development/ 文件夹。控制台列出文件夹中的对象。

步骤 5.2 授予 RAM 用户 Anne 在 Development 文件夹中获取和放入对象的权限。

若要 Anne 能够在 Development 文件夹中获取和放入对象，您必须授予她调用 oss:GetObject 和 oss:PutObject 操作的权限，包括用户必须在请求中指定前缀 Development/ 的条件。

1. 使用主账号登录*RAM*控制台。
2. 用以下策略取代您在之前步骤中创建的策略**AllowListBucketsIfSpecificPrefixIsIncluded**。

   有关详细过程，请参见 RAM 用户指南中授权策略的修改自定义授权策略部分。注意，您最多可对 RAM 策略进行五次修改。如果超过了五次，则需要删除该策略并创建一个新的策略，然后再次将新策略分配给 Staff 组。

```
{
   "Version": "1",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "oss:ListObjects"
       ],
       "Resource": [
         "acs:oss:*:*:example-company"
       ],
       "Condition": {
         "StringLike": {
           "oss:Prefix": [
             "Development/*"
           ]
         }
       }
     },
     {
       "Effect": "Allow",
       "Action": [
         "oss:GetObject",
         "oss:PutObject",
         "oss:GetObjectAcl"
       ],
       "Resource": [
         "acs:oss:*:*:example-company/Development/*"
       ],
       "Condition": {}
     }
   ]
 }
```

 说明：

当用户登录到 OSS 控制台时，控制台检查用户的身份是否有访问 OSS 服务的权限。要在控制台中支持存储空间操作，我们还需要添加 oss:GetObjectAcl 操作。

3. 测试更新的策略。

   a. 打开 RAM 用户登录链接，并用 Anne 的账号登录到 RAM 控制台。

   b. 打开 OSS 控制台。

      控制台列出所有存储空间。

   c. 在 OSS 控制台中，确认 Anne 现在可以在 Development 文件夹中添加对象并下载对象。

步骤 5.3 显式拒绝 RAM 用户 Anne 访问存储空间中任何其他文件夹的权限

RAM 用户 Anne 现在可以在 example-company 存储空间中列出根级内容，并将对象放入 Development 文件夹中。如果要严格限制访问权限，您可以显式拒绝 Anne 对存储空间中任何其他文件夹的访问。如果有授予 Anne 访问存储空间中任何其他文件夹的其他策略，则此显式策略将替代这些权限。

您可以将以下语句添加到 RAM 用户 Anne 的策略**AllowListBucketsIfSpecificPrefixIsIncluded**，以要求 Anne 发送到 OSS 的所有请求包含前缀参数，该参数的值可以是 Development/* 或空字符串。

```
{
      "Effect": "Deny",
      "Action": [
        "oss:ListObjects"
      ],
      "Resource": [
        "acs:oss:*:*:example-company"
      ],
      "Condition": {
        "StringNotLike": {
          "oss:Prefix": [
            "Development/*",
            ""
          ]
        }
      }
    }
```

按照之前的步骤更新您为 RAM 用户 Anne 创建的策略**AllowListBucketsIfSpecificPrefixIsIncluded**。复制并粘贴以下策略以替换现有策略。

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "oss:ListObjects"
      ],
      "Resource": [
        "acs:oss:*:*:example-company"
      ],
      "Condition": {
        "StringLike": {
          "oss:Prefix": [
            "Development/*"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:GetObject",
        "oss:PutObject",
        "oss:GetObjectAcl"
      ],
      "Resource": [
        "acs:oss:*:*:example-company/Development/*"
      ],
      "Condition": {}
    },
    {
      "Effect": "Deny",
      "Action": [
        "oss:ListObjects"
      ],
      "Resource": [
        "acs:oss:*:*:example-company"
      ],
      "Condition": {
        "StringNotLike": {
          "oss:Prefix": [
            "Development/*",
            ""
          ]
        }
      }
    }
  ]
}
```

步骤 **6**：授予 **RAM** 用户 **Leo** 特定权限

现在，您希望授予 Leo 访问 Marketing 文件夹的权限。请遵循之前用于向 Anne 授予权限的步骤，但应将 Development 文件夹替换为 Marketing 文件夹。有关详细过程，请参见步骤 5：授予 RAM 用户 Anne 特定权限。

步骤 **7**：确保 **Private** 文件夹安全

在本例中，您仅拥有两个用户。您在组级别授予两个用户所有所需的最小权限，只有当您真正需要单个用户级别上的权限时，才授予用户级别权限。此方法有助于最大限度地减少管理权限的工作量。随着用户数量的增加，我们希望确保不意外地授予用户对 Private 文件夹的权限。因此，我们

需要添加一个显式拒绝访问 Private 文件夹的策略。显式拒绝策略会取代任何其他权限。若要确保 Private 文件夹保持私有，可以向组策略添加以下两个拒绝语句：

- 添加以下语句以显式拒绝对 Private 文件夹 (example-company/Private/*) 中的资源执行任何操作。

```
{
    "Effect": "Deny",
    "Action": [
      "oss:*"
    ],
    "Resource": [
      "acs:oss:*:*:example-company/Private/*"
    ],
    "Condition": {}
  }
```

- 您还要在请求指定了 Private/ prefix 时拒绝执行 ListObjects 操作的权限。在控制台中，如果 Anne 或 Leo 单击 Private 文件夹，则此策略将导致 OSS 返回错误响应。

```
{
    "Effect": "Deny",
    "Action": [
      "oss:ListObjects"
    ],
    "Resource": [
      "acs:oss:*:*:*"
    ],
    "Condition": {
      "StringLike": {
        "oss:Prefix": [
          "Private/"
        ]
      }
    }
  }
```

- 用包含前述拒绝语句的更新策略取代 Staff 组策略**AllowGroupToSeeBucketListInConsole**。在应用更新策略后，组中的任何用户都不能访问您的存储空间中的 Private 文件夹。

  1. 使用主账号登录*RAM*控制台。
  2. 用以下策略取代分配给 Staff 组的现有策略**AllowGroupToSeeBucketListInConsole**。请用您的存储空间名替换策略资源中的 example-company。

```
{
"Version": "1",
"Statement": [
 {
   "Effect": "Allow",
   "Action": [
     "oss:ListBuckets",
     "oss:GetBucketAcl"
   ],
```

```
    "Resource": [
      "acs:oss:*:*:*"
    ],
    "Condition": {}
  },
  {
    "Effect": "Allow",
    "Action": [
      "oss:ListObjects"
    ],
    "Resource": [
      "acs:oss:*:*:example-company"
    ],
    "Condition": {
      "StringLike": {
        "oss:Prefix": [
           ""
         ],
        "oss:Delimiter": [
          "/"
        ]
      }
    }
  },
  {
    "Effect": "Deny",
    "Action": [
      "oss:*"
    ],
    "Resource": [
      "acs:oss:*:*:example-company/Private/*"
    ],
    "Condition": {}
  },
  {
    "Effect": "Deny",
    "Action": [
      "oss:ListObjects"
    ],
    "Resource": [
      "acs:oss:*:*:*"
    ],
    "Condition": {
      "StringLike": {
        "oss:Prefix": [
          "Private/"
        ]
      }
    }
  }
]
}
```

清理

要进行清理，您需要在 RAM 控制台中删除用户 Anne 和 Leo。

有关详细过程，请参见 RAM 用户指南中用户的删除**RAM**用户部分。

为了确保您不再因存储而继续被收取费用，您还需要删除为本示例创建的对象和存储空间。

6 接入OSS

6.1 基于OSS的移动开发

开发架构图

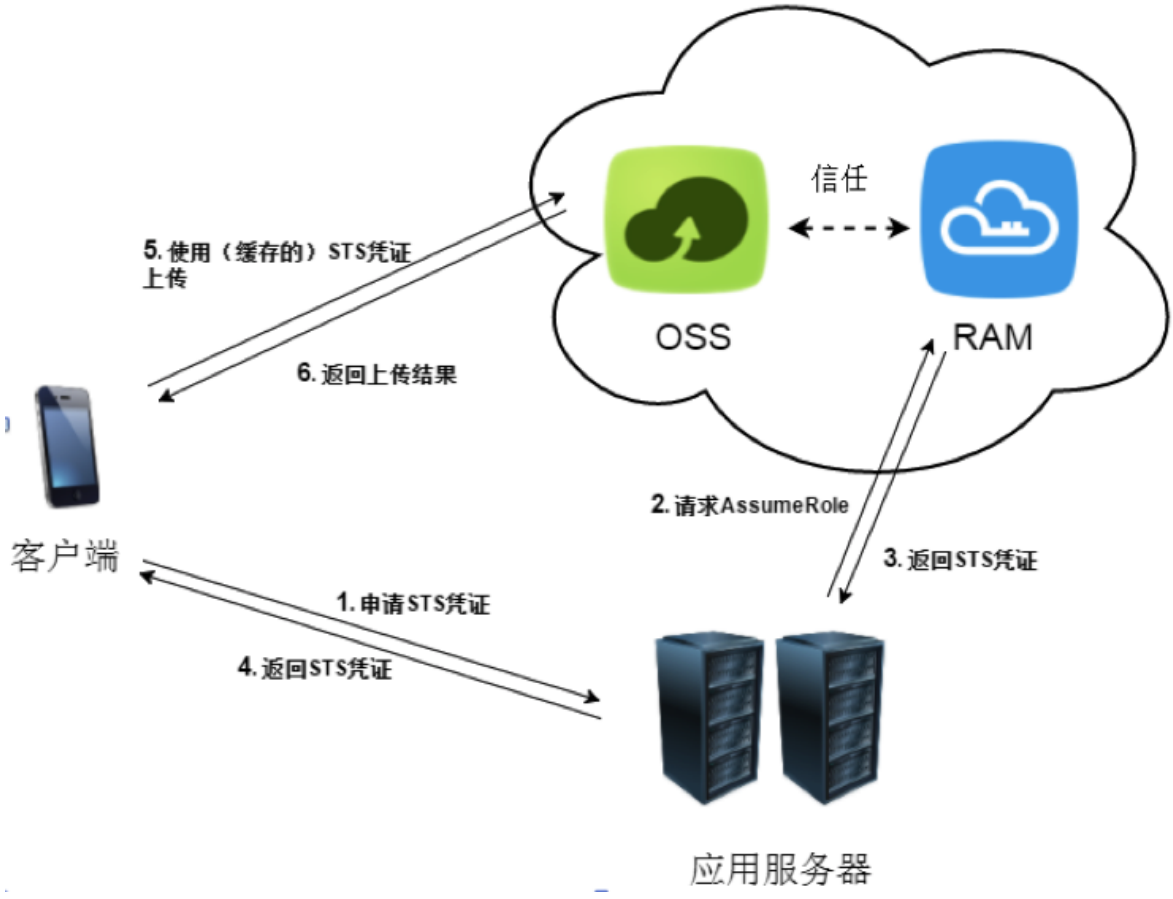
典型的基于OSS的移动开发有四个组件：

- OSS：提供上传、下载、上传回调等功能。
- 开发者的移动客户端（app或者网页应用），简称客户端：通过开发者提供的服务，间接使用OSS。
- 应用服务器：客户端交互的服务器，也是开发者的业务服务器。
- 阿里云STS：颁发临时凭证。

开发业务流程

- 临时凭证授权的上传

  如图所示：



具体步骤如下：

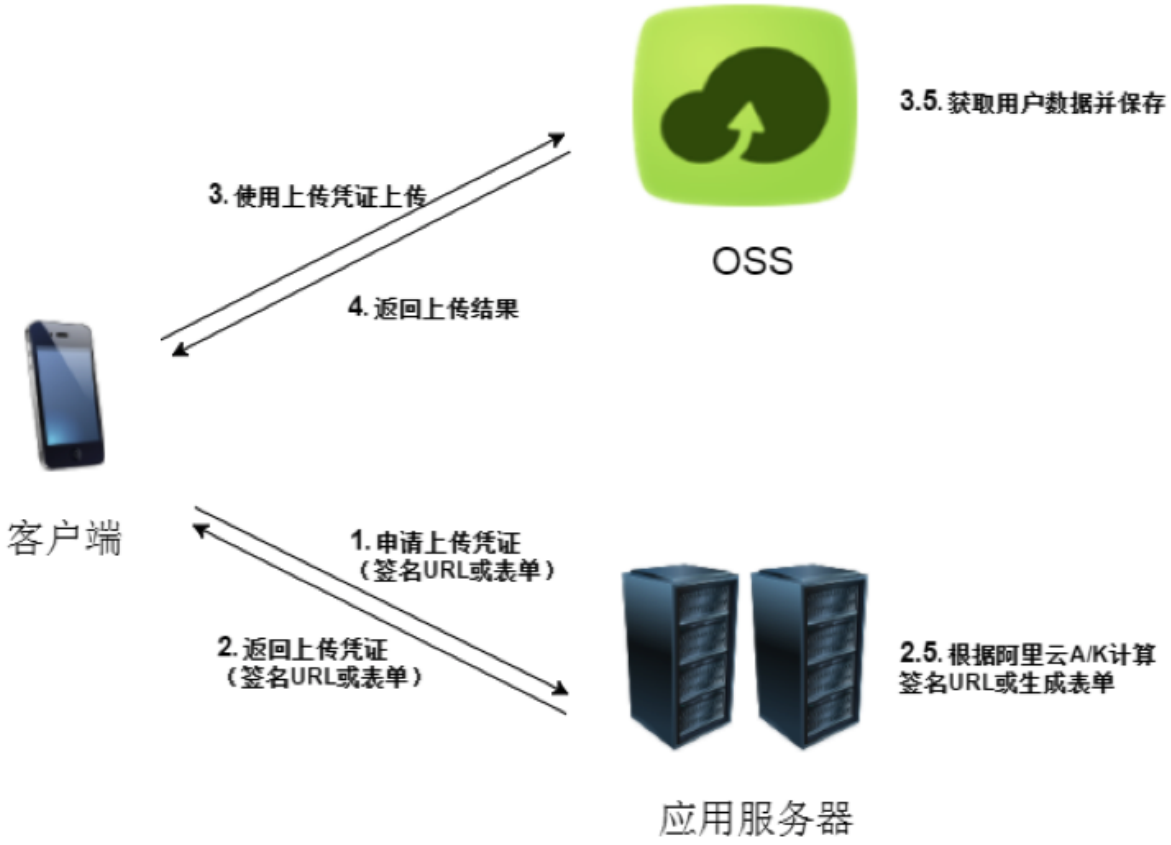
1. 客户端向应用服务器发出上传文件到OSS的请求。
2. 应用服务器向STS服务器请求一次，获取临时凭证。
3. 应用服务器回复客户端，将临时凭证返回给客户端。
4. 客户端获取上传到OSS的授权（STS的AccessKey以及Token），调用OSS提供的移动端SDK上传。
5. 客户端成功上传数据到OSS。如果没有设置回调，则流程结束。如果设置了回调功能，OSS会调用相关的接口。

> **说明：**
>
> - 客户端不需要每次都向应用服务器请求授权，在第一次授权完成之后可以缓存STS返回的临时凭证直到超过失效时间。
> - STS提供了强大的权限控制功能，可以将客户端的访问权限限制到Object级别，从而实现不同客户端在OSS端上传的Object的完全隔离，极大提高了安全性。

更多信息请参见授权给第三方上传。

- 签名URL授权的上传和表单上传

  如图所示：



具体步骤如下：

1. 客户端向应用服务器发出上传文件到OSS的请求。
2. 应用服务器回复客户端，将上传凭证（签名URL或者表单）返回给客户端。
3. 客户端获取上传到OSS的授权（签名URL或者表单），调用OSS提供的移动端SDK上传或者直接表单上传。
4. 客户端成功上传数据到OSS。如果没有设置回调，则流程结束。如果设置了回调功能，OSS会调用相关的接口。

更多信息请参见授权给第三方上传。

- 临时凭证授权的下载

  跟临时凭证授权的上传过程类似：

  1. 客户端向应用服务器发出下载OSS文件的请求。
  2. 应用服务器向STS服务器请求一次，获取临时凭证。
  3. 应用服务器回复客户端，将临时凭证返回给客户端。
  4. 客户端获取下载OSS文件的授权（STS的AccessKey以及Token），调用OSS提供的移动端SDK下载。

5. 客户端成功从OSS下载文件。

> 说明：
>
> - 和上传类似，客户端对于临时凭证也可以进行缓存从而提高访问速度。
> - STS同样也提供了精确到Object的下载权限控制，和上传权限控制结合在一起可以实现各移动端在OSS上存储空间的完全隔离。

- 签名URL授权的下载

  跟签名URL授权的上传类似：

  1. 客户端向应用服务器发出下载OSS文件的请求。
  2. 应用服务器回复客户端，将签名URL返回给客户端。
  3. 客户端获取下载OSS文件的授权（签名URL），调用OSS提供的移动端SDK下载。
  4. 客户端成功从OSS下载文件。

  > 说明：
  > 客户端不能存储开发者的AccessKey，只能获取应用服务器签名的URL或者是通过STS颁发的临时凭证（也就是STS的AccessKey和Token）。

最佳实践

- 快速搭建移动应用直传服务
- 快速搭建移动应用上传回调服务
- 权限控制
- *RAM和STS使用指南*

功能使用参考

- *Android SDK上传文件*
- *iOS SDK上传文件*

6.2 快速开始

1. 登录*OSS管理控制台*，开通OSS。
2. 创建一个Bucket。
3. 上传和下载文件。

具体请参见开始使用阿里云 *OSS*。

快速了解OSS的上传下载

开始使用SDK之前，请先参考开发人员指南关于上传下载文件的功能介绍。

OSS是使用RESTful API来操作的，所有的请求都是标准的HTTP请求。

- OSS提供了多种类型的上传文件的方法，如使用单次PUT请求完成的简单上传，使用网页表单直接上传的表单上传，用于大文件上传的分片上传，以及适用于视频监控等领域的追加上传。
- 同样也可以使用多种方法从OSS下载文件，如简单下载和下载大文件的断点续传下载。

基于SDK快速开始

1. 开通OSS后，从控制台上获取AccessKeyId和AccessKeySecret。
2. 下载各种开发语言SDK。
3. 根据SDK的文档描述，完成上传、下载文件等操作。

具体请参见*SDK*文档。

7 管理存储空间

7.1 创建存储空间

您可以选择在已有的地域创建存储空间。同时需要注意有下列限制：

- 同一用户创建的存储空间总数不能超过 30 个。
- 每个存储空间的名字全局唯一，否则会创建失败。
- 存储空间的名称需要符合命名规范。
- 存储空间一旦创建成功，名称和所处地域不能修改。

OSS 提供 ACL（Access Control List）权限控制方法，您可以在创建存储空间的时候设置相应的存储空间权限（ACL），也可以在创建之后修改 ACL。如果不设置 ACL，默认值为私有读写。更多信息，请参见设置存储空间读写权限。

功能使用参考

- 控制台：创建存储空间
- API：*PutBucket*

7.2 设置WORM策略

WORM（一次写入，多次读取）策略用于指定Bucket内文件的保护周期。在保护周期内，任何人都不能对文件进行修改和删除操作。

> 说明：
> 目前，OSS仅支持针对Bucket级别设置WORM策略。

规则说明

目前，OSS只允许添加一条基于时间的 WORM 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三个文件。随后，在2014年7月1日创建了保护周期为5年的WORM策略。有关这三个文件的具体上传时间以及对应的WORM规则到期日期如下：

文件名称	上传时间	WORM规则到期日期
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

文件名称	上传时间	WORM规则到期日期
file3.txt	2018年9月30日	2023年9月29日

您还可以指定基于时间的WORM策略的生效规则、修改规则以及删除规则。

- 生效规则

  当基于时间的WORM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WORM策略24小时内：若该WORM策略未提交锁定，则Bucket所有者以及授权用户可以修改或者删除该策略。若该WORM策略已提交锁定，则不允许修改和删除该策略，但是您可以延长保护周期。
  - 启动 WORM策略24小时后：若超过24小时该WORM策略未提交锁定，则该策略自动失效。

- 修改规则

  针对基于时间的WORM策略的InProgress和 Locked两种状态，提供了两种修改规则：

  - 处于InProgress状态时，只能删除规则。
  - 处于Locked状态时，无法修改和删除规则，且只允许延长保护时间。

- 删除规则

  基于时间的WORM策略是Bucket的一种Metadata属性。当删除某个Bucket时，该Bucket对应的WORM策略以及访问策略也会被删除。

功能使用参考

控制台：设置*WORM*策略

7.3 请求者付费模式

通常OSS Bucket拥有者将支付该Bucket的存储费用、流量费用和请求费用。但是，Bucket拥有者可以将Bucket配置为开启请求者付费模式。使用请求者付费模式时，请求方 (而不是Bucket拥有者) 将支付流量费用和请求费用。而Bucket拥有者将始终支付存储费用。

通常情况下，当您想共享数据，而又不希望产生读取数据等其他操作带来的费用时，您可以配置Bucket开启请求者付费模式。

说明：

请求者付费模式目前只在华南1（深圳）开放。

使用案例

- 共享大型数据集 (如邮政编码目录、参考数据、地理空间信息或网络爬取数据)。所有OSS的认证客户都能访问该数据，但请求产生的流量费用和请求费用由请求者支付。

  配置步骤如下：

  1. 将Bucket开启请求者付费模式。
  2. 将Bucket ACL设置为公共读。

- 您将数据交付给客户或合作伙伴。您的账号以及Bucket Policy里授权的合作伙伴账号可以访问该Bucket，但请求产生的流量费用和请求费用由请求者支付。

  配置步骤如下：

  1. 将Bucket开启请求者付费模式。
  2. 将Bucket ACL设置为私有。
  3. 利用Bucket Policy授权，您的合作伙伴的阿里云账号可以访问该Bucket。

请求方式说明

- 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。

- 申请方需携带x-oss-request-payer信息

  如果您在Bucket上启用了请求者付费模式，请求方必须在其请求中包含 x-oss-request-payer: requseter (在 POST、GET 和 HEAD 请求的Head信息中)，以表明请求方知道请求和数据下载将产生费用。否则，请求方无法通过验证。

  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。

费用详解

请求者付费模式下，请求者支付流量费用和请求费用，Bucket拥有者支付存储费用。但是在以下情况下，请求会失败（返回HTTP 403错误），将对Bucket拥有者收取请求费用：

- 请求者未在请求中 (GET、HEAD 或 POST) 包含参数 x-oss-request-payer，或未在请求中将其作为参数 (REST) 。

- 请求身份验证失败。
- 请求是匿名请求 。

功能使用参考

控制台：设置请求者付费模式

7.4 设置存储空间读写权限（ACL）

除了在创建存储空间的时候能够对存储空间的 ACL 进行设置，也可以之后根据自己的业务需求对存储空间的ACL进行修改。这个操作只有该存储空间的创建者有权限执行。目前存储空间有三种访问权限:

权限值	中文名称	权限对访问者的限制
public-read-write	公共读写	任何人（包括匿名访问）都可以对该存储空间中的文件进行读写操作，所有这些操作产生的费用由该存储空间的拥有者承担，请慎用该权限。
public-read	公共读，私有写	只有该存储空间的拥有者或者授权对象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）可以对该存储空间中的文件进行读操作。
private	私有读写	只有存储空间的拥有者或者授权对象可以对该存储空间内的文件进行读写操作，其他人在未经授权的情况下无法访问该存储空间内的文件。

详细解释请参见访问权限。

功能使用参考

设置存储空间 ACL

- 控制台：设置访问权限
- API：*PutBucketACL*

获取存储空间 ACL

- 控制台：登录后可以在存储空间属性中查看
- API：*GetBucketACL*

7.5 查看存储空间列表

查看您创建的所有存储空间列表。

功能使用参考

- 控制台：进入控制台后默认显示您创建的存储空间列表
- API：*GetService*

相关参考链接

- 创建*Bucket*

7.6 获取存储空间信息

您可以获取存储空间所属的地域，即数据中心的物理位置信息。返回的 Location 字段显示存储空间所在的地域信息，比如华东 1（杭州）的 Location 字段信息显示为 oss-cn-hangzhou。请参见 OSS 提供的访问域名。

功能使用参考

- 控制台：进入控制台后在存储空间属性中直接显示地域信息
- API：*Get Bucket Location*

7.7 删除存储空间

您可以删除您创建的存储空间。如果存储空间不为空（存储空间中有文件或者是尚未完成的分片上传），则存储空间无法删除，必须删除存储空间中的所有文件和未完成的分片文件后，存储空间才能成功删除。如果想删除存储空间内部所有的文件，推荐使用 生命周期管理。

功能使用参考

- API：*Delete Bucket*
- 控制台：删除存储空间

8 上传文件

8.1 简单上传

适用场景

简单上传指的是使用OSS API中的PutObject方法上传单个Object。简单上传适用于一次HTTP请求交互即可完成上传的场景，比如小文件的上传。

上传文件时设置Object Meta

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，比如可以设定Content-Type等标准HTTP头，也可以设定自定义信息。具体请参考设置文件元信息。

上传限制

- 大小限制：Object的大小不能超过5GB。
- 命名限制
  - 使用UTF-8编码。
  - 长度必须在1－1023字节之间。
  - 不能以正斜线（/）或者反斜线（\）字符开头。

大文件上传

因为使用的是单次HTTP请求，Object过大会导致上传时间长。在这段时间出现网络原因造成超时或者链接断开等错误的时候，上传容易失败，可以考虑断点续传上传（分片上传）。当Object大于5GB时，只能使用断点续传上传，具体参考断点续传上传。

上传的安全及授权

为了防止第三方未经授权往开发者的Bucket上传数据，OSS提供了Bucket和Object级别的访问权限控制，详情请参见访问控制。为了授权给第三方上传，OSS除了Bucket和Object级别的访问权限外，还提供了账号级别的授权，请参见授权给第三方上传。

上传后续操作

在文件上传到OSS上后，开发者可以使用上传回调来向指定的应用服务器发起回调请求，进行下一步操作。如果上传的是图片需要处理，可以使用图片处理。如果上传的是音频或者视频文件可以使用媒体处理。

功能使用参考

- 控制台：上传文件
- API：*PutObject*

最佳实践

- *RAM*和*STS*使用指南
- *Web*端直传实践及上传回调

8.2 表单上传

适用场景

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5GB。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：

	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。

上传限制

- 大小限制：使用表单上传时，Object不能超过5GB。
- 命名限制：
  - 使用UTF-8编码。
  - 长度必须在1－1023字节之间。
  - 不能以正斜线（/）或者反斜线（\）字符开头。

优势

- 从流程上来说，少了一步转发，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，压力都在OSS上，由OSS来保障服务质量。

安全及授权

- 为了授权给第三方上传，使用的是PostPolicy方法，详情请参见*PostObject*。

使用表单上传的基本步骤

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名字等，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。具体请参考*Post Policy*。

   例如如下Policy，网站用户能上传的过期时间是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），能上传的文件最大为104857600字节。

   ```
   以Python代码为例子，Policy是json格式的字符串。
    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。

```
用法：
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
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。
- 假如指定生成的HTML文件为post.html，打开post.html，选择文件上传。在这个例子中如果成功则会跳转到OSS的主页面。

功能使用参考

- API：*PostObject*
- Java SDK：表单上传

最佳实践

- *Web*端直传实践
- 跨域资源共享*#CORS#*

8.3 分片上传和断点续传

适用场景

当使用简单上传（PutObject）功能来上传较大的文件到OSS的时候，如果上传的过程中出现了网络错误，那么此次上传失败。重试必须从文件起始位置上传。针对这种情况，OSS提供了分片上传（Multipart Upload）来达到断点续传的效果。顾名思义，分片上传就是将要上传的文件分成多个数据块（OSS里又称之为Part）来分别上传，上传完成之后再调用OSS的接口将这些Part组合成一个Object。

相对于其他的上传方式，分片上传适用于以下场景：

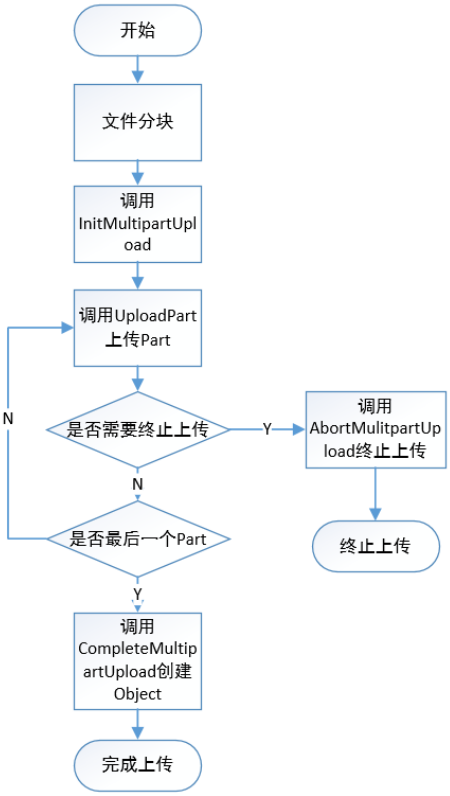
- 恶劣的网络环境：如手机端，当出现上传失败的时候，可以对失败的Part进行独立的重试，而不需要重新上传其他的Part。
- 断点续传：中途暂停之后，可以从上次上传完成的Part的位置继续上传。
- 加速上传：要上传到OSS的本地文件很大的时候，可以并行上传多个Part以加快上传。
- 流式上传：可以在需要上传的文件大小还不确定的情况下开始上传。这种场景在视频监控等行业应用中比较常见。

分片上传流程

分片上传的基本流程如下：

1. 将要上传的文件按照一定的大小分片。
2. 初始化一个分片上传任务（*InitiateMultipartUpload*）。

3. 逐个或并行上传分片（*UploadPart*）。
4. 完成上传（*CompleteMultipartUpload*）。



该过程需注意以下几点：

- 除了最后一块Part，其他Part的大小不能小于100KB，否则会导致调用*CompleteMultipartUpload*接口时失败。
- 要上传的文件切分成Part之后，文件顺序是通过上传过程中指定的partNumber来确定的，实际执行中并没有顺序要求，因此可以实现并发上传。具体的并发个数并不是越多速度越快，要结合用户自身的网络情况和设备负载综合考虑。
- 默认情况下，已经上传但还没有调用*CompleteMultipartUpload*的Part是不会自动回收的，因此如果要终止上传并删除占用的空间请调用*AbortMultipartUpload*。如果需要自动回收上传的Part，请参考Object生命周期管理。

断点续传

因为已经上传的Part的生命周期是永久的，因此很容易可以实现断点续传的功能。

在使用分片上传的过程中，如果系统意外崩溃，可以在重启的时候通过*ListMultipartUploads*和*ListParts*两个接口来获取某个Object上的所有的分片上传任务和每个分片

上传任务中上传成功的Part列表。这样就可以从最后一块成功上传的Part开始继续上传，从而达到断点续传的效果。暂停和恢复上传实现原理也是一样的。

断点续传功能在移动设备和大文件上传中的优势尤为明显。

上传限制

- 大小限制：在这种上传方式下，Object的大小是由Part来决定的，最大支持10000块Part。每块Part最小100KB（最后一块可以比100KB小），最大5GB。Object的大小不能超过48.8TB。
- 命名限制
  - 使用UTF-8编码。
  - 长度必须在1－1023字节之间。
  - 不能以正斜线（/）或者反斜线（\）字符开头。

上传的安全及授权

为了防止第三方往开发者的Bucket未经授权上传，OSS提供了Bucket和Object级别的访问权限控制，详细解释见访问控制。为了授权给第三方上传，OSS除了Bucket和Object级别的访问权限外，还提供了账号级别的授权，见上传安全之授权第三方。

上传后续操作

- 在文件上传到OSS上后，开发者可以使用上传后回调来向指定的应用服务器发起回调请求，进行下一步操作。
- 如果上传的是图片，可以使用图片服务进行后续处理。
- 如果上传的是音频或者视频文件，可以使用媒体转码进行后续处理。

功能使用参考

- 分片上传API：
  - *MultipartUpload*
  - *InitiateMultipartUpload*
  - *UploadPart*
  - *UploadPartCopy*
  - *CompleteMultipartUpload*
  - *AbortMultipartUpload*
  - *ListMultipartUploads*

- *ListParts*

最佳实践

- *RAM和STS使用指南*
- *Web端直传实践*

8.4 追加上传

适用场景

之前提到的上传方式，比如简单上传，表单上传，断点续传上传等，创建的Object都是Normal类型，这种Object在上传结束之后内容就是固定的，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，这也是OSS和普通文件系统使用的一个重大区别。

正因为这种特性，在很多应用场景下会很不方便，典型比如视频监控、视频直播领域等，视频数据在实时的不断产生。如果使用其他上传方式，只能将视频流按照一定规律切分成小块然后不断的上传新的Object。这种方式在实际使用上存在很明显的缺点：

- 软件架构比较复杂，需要考虑文件分块等细节问题。
- 需要有位置保存元数据，比如已经生成的Object列表等，然后每次请求都重复读取元数据来判断是否有新的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切分的比较小的话，延时比较低，但是众多Object会导致管理起来很复杂。如果Object切分的比较大的话，数据的延时又会很高。

为了简化这种场景下的开发成本，OSS提供了追加上传（Append Object）的方式在一个Object后面直接追加内容的功能。通过这种方式操作的Object的类型为Appendable Object，而其他的方式上传的Object类型为Normal Object。每次追加上传的数据都能够即时可读。

如果使用追加上传，那么上述场景的架构就变得很简单。视频数据产生之后即时地通过追加上传到同一个Object，而客户端只需要定时获取该Object的长度与上次读取的长度进行对比，如果发现有新的数据可读，那么就触发一次读操作来获取新上传的数据部分即可。通过这种方式可以很大的简化架构，增强扩展性。

不仅在视频场景，在日志追加上传的场景下，追加上传也能发挥作用。

上传限制

- 大小限制：在这种上传方式下，Object不能超过5GB。
- 命名限制：
  - 使用UTF-8编码。
  - 长度必须在1 - 1023字节之间。
  - 不能以正斜线（/）或者反斜线（\）字符开头。
- 文件类型：只有通过追加上传创建的文件才可以后续继续被追加上传。也就是说，其他通过简单上传、表单上传、分片上传得到的文件，不能在这些文件后面追加上传新的内容。
- 后续操作限制：通过追加上传的文件，不能被复制，可以修改文件本身的meta信息。

上传的安全及授权

为了防止第三方往开发者的Bucket未经授权上传，OSS提供了Bucket和Object级别的访问权限控制，详细解释见访问控制。为了授权给第三方上传，OSS除了Bucket和Object级别的访问权限外，还提供了账号级别的授权，见上传安全之授权第三方。

上传后续操作

如果上传的是图片需要处理，可以使用图片处理。如果上传的是音频或者视频文件也可以使用媒体处理。

功能使用参考

- API：*AppendObject*

> **说明：**
> 追加上传不支持上传回调操作。

最佳实践

- *RAM和STS使用指南*

8.5 授权给第三方上传

适用场景

在典型的C/S系统架构中，服务器端负责接收并处理客户端的请求。那么考虑一个使用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，一次从客户端到服务器端，一次从服务器端到

OSS。当访问量很大的时候，服务器端需要有足够的带宽资源来满足多个客户端的同时上传的需求，这对架构的伸缩性提出了挑战。

为了解决这种场景带来的挑战，OSS提供了授权给第三方上传的功能。使用这个功能，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，节省了自建服务器的成本，并且充分利用了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。

目前授权上传可以由两种实现方式：URL签名和临时访问凭证。

URL签名

URL签名是授权访问的一种方式，即在请求的URL中带OSS AccessKeyId和Signature字段，这样用户就可以直接使用该URL来进行上传。每个URL签名中携带有过期时间以保证安全。具体的做法可以参考在*URL*中包含签名。

临时访问凭证

临时访问凭证是通过阿里云Security Token Service(STS)来实现授权的一种方式。其实现请参见*STS Java SDK*。临时访问凭证的流程如下：

1. 客户端向服务器端发起获得授权的请求。服务器端先验证客户端的合法性。如果是合法客户端，那么服务器端会使用自己的AccessKey来向STS发起一个请求授权的请求，具体可以参考访问控制。
2. 服务器端获取临时凭证之后返回给客户端。
3. 客户端使用获取的临时凭证来发起向OSS的上传请求，更详细的请求构造可以参考临时授权访问。客户端可以缓存该凭证用来上传，直到凭证失效再向服务器端请求新的凭证。

最佳实践

- *RAM*和*STS*使用指南
- *Web*端直传实践及上传回调

8.6 上传回调

适用场景

OSS在上传文件完成的时候可以提供回调（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，即能实现回调。现在支持CallBack的API 接口有：PutObject、PostObject、CompleteMultipartUpload。

上传回调的一种典型应用场景是与授权第三方上传同时使用，客户端在上传文件到OSS的时候指定到服务器端的回调，当客户端的上传任务在OSS执行完毕之后，OSS会向应用服务器端主动发起HTTP请求进行回调，这样服务器端就可以及时得到上传完成的通知从而可以完成诸如数据库修改等操作，当回调请求接收到服务器端的响应之后OSS才会将状态返回给客户端。

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，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的时候根据应用逻辑的需要指定这些参数。您可以通过使用自定义参数来携带一些和应用逻辑相关的信息，比如发起请求的用户id等。具体使用自定义参数的方法可以参考*Callback*。

通过适当的使用上传回调机制，能很好的降低客户端的逻辑复杂度和网络消耗。流程如下：

说明：

- 上传回调支持的地域包括中国大陆地域、香港、亚太东南1、亚太东南2、美东、美西、亚太东北1、欧洲中部1、中东东部1。
- 目前只有简单上传（PutObject）、表单上传（PostObject）、分片上传完成（Complete Multipart Upload）操作支持上传回调功能。

功能使用参考

- API：*Callback*
- iOS SDK上传后回调通知

最佳实践

- *Web*端直传实践及上传回调
- 快速搭建移动应用上传回调服务

8.7 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。

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。
- 必须包含视频流，且视频流格式为H264。
- 音频流是可选的，并且只支持AAC格式，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。

下面分别介绍如何推送音视频流到OSS，以及如何点播和直播播放。

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），那么可以直接使用得到的推流地址进行推流；否则需要进行签名操作。

  以Python SDK为例，获取未签名以及签名推流地址的代码如下：

```
from oss2 import *
from oss2.models import *
host = "oss-cn-hangzhou.aliyuncs.com" #just for example
accessid = "your-access-id"
accesskey = "your-access-key"
bucket_name = "your-bucket"
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
signed_publish_url = bucket.sign_rtmp_url("test-channel", "playlist.
m3u8", 3600)
```

  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=LGarWrijh8HjKWg6&playlistName=t.
m3u8&Expires=1472201595&Signature=bjKraZTTyzz9%2FpYoomDx4Wgh%2FlM%3D
"
```

- 使用ffmpeg进行推流

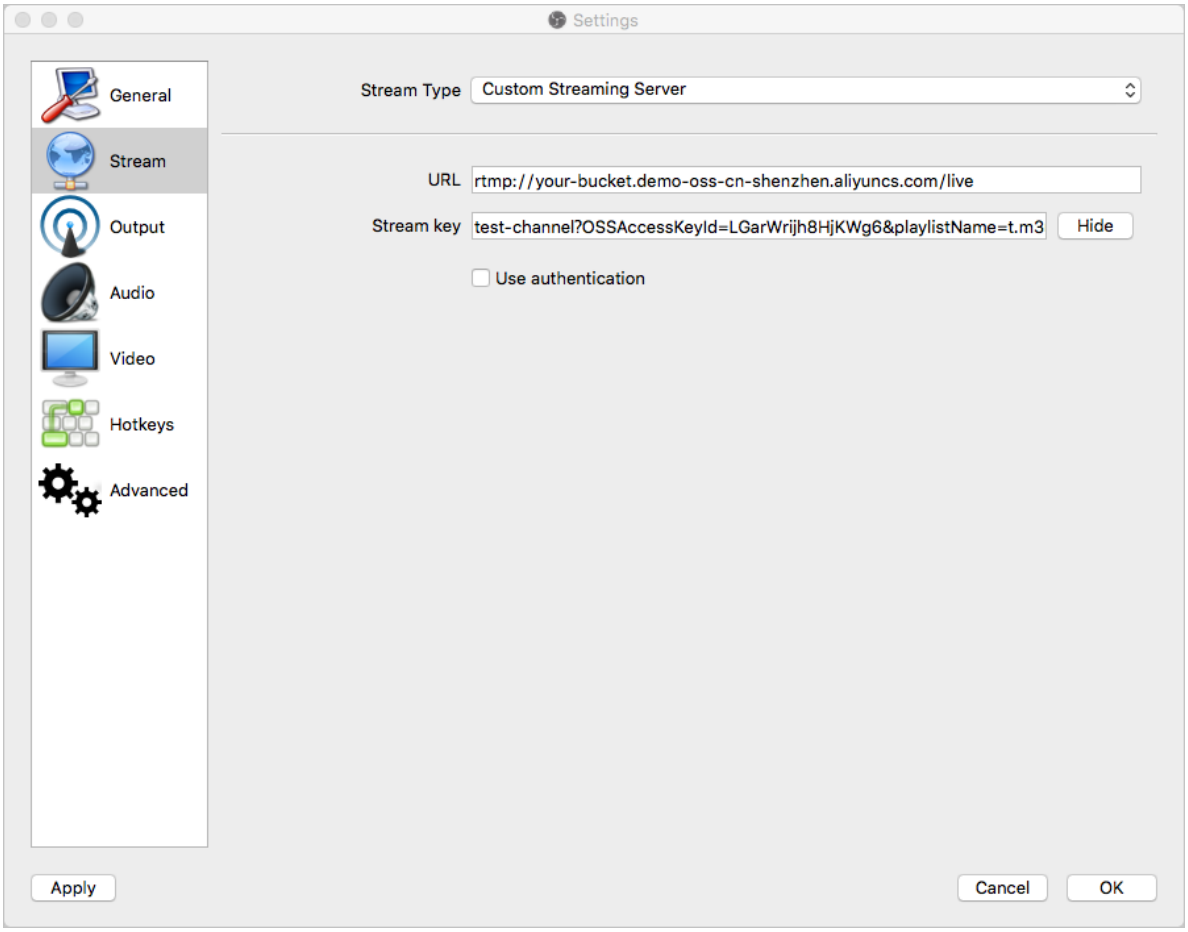
可以使用ffmpeg推送本地的视频文件到OSS，命令如下：

```
ffmpeg -i 1.flv -c copy -f flv "rtmp://your-bucket.oss-cn-hangzhou
.aliyuncs.com/live/test-channel?OSSAccessKeyId=LGarWrijh8HjKWg6&
Expires=1472199095&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：



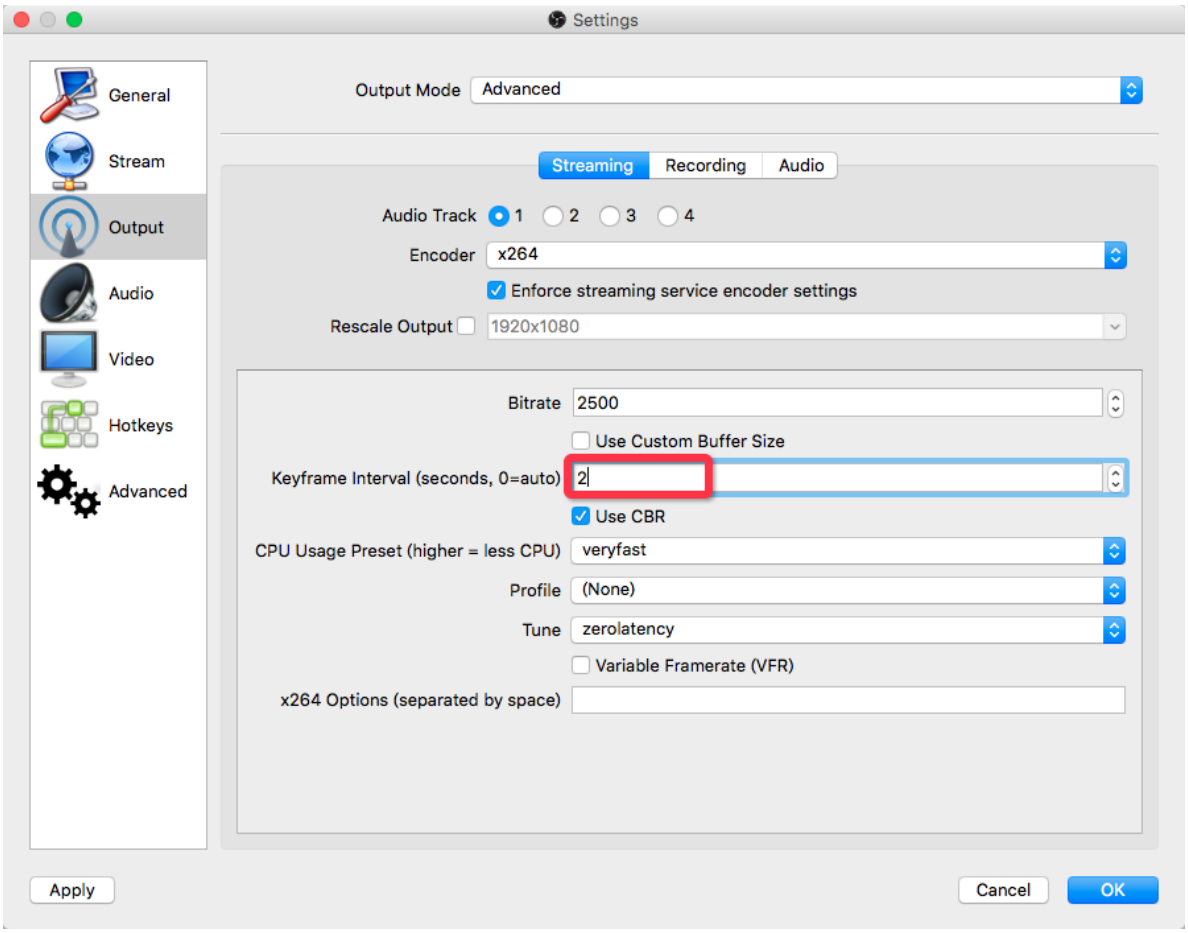
播放推送到OSS的音视频数据

- 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播放器进行播放。

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总是以直播的方式更新m3u8文件。所以对于点播的场景，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，降低码率。

9 下载文件

9.1 简单下载

简单下载即下载已经上传的文件（Object），Object下载是使用HTTP的GET请求来完成的。Object的URL生成规则请参考*OSS*的访问。如果需要使用自定义域名来访问Object，请参考自定义域名访问*OSS*。

当用户需要访问某个Object的时候，有两种情况：

- 该Object没有匿名读的权限，用户拥有对应的AccessKey，那么可以使用AccessKey对GET请求进行签名来访问。
- 该Object拥有匿名读的权限，那么所有的用户都可以直接使用GET请求来进行访问。

具体的Object和Bucket的访问权限控制请参考访问控制。

如果希望将私有Bucket的Object提供给第三方进行下载，请参考授权给第三方下载。

如果需要实现断点下载请参考断点续传下载。

功能使用参考

- API：*GetObject*
- Java SDK：下载文件
- 控制台：获取文件访问地址

最佳实践

- *RAM*和*STS*使用指南

9.2 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，重启的时候也可以从上次完成的位置开始继续下载。

和简单上传类似，您也需要对该Object有读权限。通过设置参数Range来支持断点续传，对于比较大的Object建议使用该功能。Range的定义可参考HTTP RFC。如果在请求头中使用Range参数，则返回消息中会包含整个文件的长度和此次返回的范围。例如：Content-Range: bytes 0－9/44，表示整个文件长度为44，此次返回的范围为0－9。如果不在范围内，则传送整个文件，并且不在结果中提及Content-Range，返回码为206。

功能使用参考

- API：*GetObject*

9.3 授权给第三方下载

当您希望将私有Bucket内部的Object授权给第三方下载的时候，不应该直接将AccessKey提供给下载者，而应该使用以下两种方法。

URL签名

OSS提供了签名下载的方法。您可以在URL中加入签名信息，把该URL转给第三方实现授权访问。第三方用户只需要使用HTTP的GET请求访问此URL即可下载Object。

- 实现方式

  URL中包含签名示例如下：

  ```
  http://<bucket>.<region>.aliyuncs.com/<object>?OSSAccessKeyId=<user access_key_id>&Expires=<unix time>&Signature=<signature_string>
  ```

  在URL中实现签名，必须至少包含Signature、Expires、OSSAccessKeyId三个参数。

  — OSSAccessKeyId：您的AccessKeyId。

  — Expires：您期望URL过期的时间。

  — Signature：您签名的字符串，具体请参见在*URL*中包含签名。

  > 说明：
  > 此链接需要进行URL编码。

- 功能使用参考

  — API：*Get Object*

  — 控制台：获取文件访问地址

  > 说明：
  > 在控制台中只有当Bucket处于私有读写权限的时候，获取的访问地址才是URL中签名的形式。否则为不带签名的URL形式。

临时访问凭证

- 实现方式

第三方用户向app服务器请求，获取了STS颁发的AccessKeyId、AccessKeySecret以及STS Token。第三方用户以STS AccessKeyId、AccessKeySecret以及STS Token去请求开发者的Object资源。

- 功能使用参考
  - API：临时访问凭证
  - 控制台：获取文件访问地址

最佳实践

- *RAM和STS使用指南*

9.4 选取内容（OSS Select）

通过 OSS Select，您可以使用简单的结构化查询语言 (SQL) 语句选取 OSS 中文件的内容，仅获取需要的数据。使用 OSS Select 可以减少从 OSS 传输的数据量，从而提高数据获取效率，节约时间。

限制

- 文件格式：OSS Select 支持编码为 UTF-8 且符合 RFC 4180 标准的 CSV 文件（包括TSV 等类 CSV 文件）。文件的行列分隔符及 Quote 字符都可以自由定义。
- 加密方式：OSS Select 支持 OSS 完全托管和通过 KMS 托管主密钥进行加密的文件。
- 地域：OSS Select 目前只在华南1（深圳）开放。

支持的 SQL 语句

您可以在请求中将 SQL 表达式传递给 OSS。有关 OSS Select 支持的 SQL 语句和限制，请参考*SelectObject*。

功能使用参考

- 控制台：选取内容
- API：*SelectObject*
- OSS SDK（目前 Java 和 Python 的 SDK 已支持 OSS Select）
- 阿里云*DataLakeAnalytics*

10 安全管理

10.1 设置访问日志记录

您可以开启存储空间的访问日志记录功能。当一个Bucket（源Bucket，Source Bucket）开启访问日志记录功能后，OSS自动将访问这个Bucket的请求日志，以小时为单位，按照固定的命名规则，生成一个Object写入您指定的Bucket（目标Bucket，Target Bucket）。

日志记录Object命名规则

```
<TargetPrefix><SourceBucket>YYYY-mm-DD-HH-MM-SS-UniqueString
```

命名规则中，

- `TargetPrefix`由用户指定，表示存储访问日志记录的Object名字前缀，可以为空。
- `YYYY-mm-DD-HH-MM-SS`分别是该Object被创建时的阿拉伯数字的年、月、日、小时、分钟和秒（注意位数）。
- `UniqueString`为OSS系统生成的字符串（UUID），用于唯一标识该Log文件。

存储OSS访问日志的Object的名称例子如下：

```
MyLog-oss-example2017-09-10-04-00-00-0000
```

上例中，

- `MyLog-`是用户指定的Object前缀。
- `oss-example`是源Bucket的名称。
- `2017-09-10-04-00-00`是该Object的创建时间。
- `0000` 是OSS系统生成的字符串。

Log文件格式

Log文件的格式组成：以下名称从左至右，以空格分隔。

名 称	例 子	含 义
Remote IP	119.140.142.11	请求发起的IP地址（Proxy代理或用户防火墙可能会屏蔽该字段）
Reserved	-	保留字段

名 称	例 子	含 义
Reserved	-	保留字段
Time	[02/May/2012:00:00:04 +0800]	OSS收到请求的时间
Request-URI	“GET /aliyun-logo.png HTTP/1.1”	用户请求的URI（包括query-string）
HTTP Status	200	OSS返回的HTTP状态码
SentBytes	5576	用户从OSS下载的流量
RequestTime (ms)	71	完成本次请求的时间（毫秒）
Referer	`http://www.aliyun.com/product/oss`	请求的HTTP Referer
User-Agent	curl/7.15.5	HTTP的User-Agent头
HostName	oss-example.oss-cn-hangzhou.aliyuncs.com	请求访问域名
Request ID	505B01695037C2AF032593A4	用于唯一标识该请求的UUID
LoggingFlag	true	是否开启了访问日志功能
Requester Aliyun ID	1657136103983691	请求者的阿里云ID；匿名访问为“-”
Operation	GetObject	请求类型
Bucket	oss-example	请求访问的Bucket名字
Key	/aliyun-logo.png	用户请求的Key
ObjectSize	5576	Object大小
Server Cost Time (ms)	17	OSS服务器处理本次请求所花的时间（毫秒）
Error Code	NoSuchBucket	OSS返回的错误码
Request Length	302	用户请求的长度（Byte）
UserID	1657136103983691	Bucket拥有者ID
Delta DataSize	280	Bucket大小的变化量；若没有变化为`-`
Sync Request	-	是否是CDN回源请求；若不是为`-`

名 称	例 子	含 义
Reserved	-	保留字段

细节分析

- 源Bucket和目标Bucket可以是同一个Bucket，也可以是不同的Bucket，但必须属于同一个账号下的同一数据中心内。您也可以将多个源Bucket的Log都保存在同一个目标Bucket内（建议指定不同的TargetPrefix）。
- OSS以小时为单位生成Bucket访问的Log文件，但并不表示这个小时的所有请求都记录在这个小时的Log文件内，也有可能出现在上一个或者下一个Log文件中。
- OSS生成一个Bucket访问的Log文件，算作一次PUT操作，并记录其占用的空间，但不会记录产生的流量。Log生成后，您可以按照普通的Object来操作这些Log文件。
- OSS会忽略掉所有以`x-`开头的query-string参数，但这个query-string会被记录在访问Log中。如果您想从海量的访问日志中标识一个特殊的请求，可以在URL中添加一个`x-`开头的query-string参数。例如：

  ```
  http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png
  ```

  ```
  http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png?x-user=admin
  ```

  OSS处理上面两个请求，结果是一样的。但是在访问Log中，您可以通过搜索`x-user=admin`，方便地定位出经过标记的这个请求。
- OSS的Log中的任何一个字段，都可能出现`-`，用于表示未知数据或对于当前请求该字段无效。
- 根据需求，OSS的Log格式将来会在尾部添加一些字段，请开发者开发Log处理工具时考虑兼容性的问题。

功能使用参考

- 控制台：设置日志
- API：*PutBucketLogging*，*DeleteBucketLogging*，*GetBucketLogging*
- Java SDK：设置访问日志
- Java SDK：设置访问日志

10.2 设置防盗链

为了防止您在OSS上的数据被其他人盗链而产生额外费用，您可以设置防盗链功能，包括以下参数：

- Referer白名单。仅允许指定的域名访问OSS资源。
- 是否允许空Referer。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。

例如，对于一个名为oss-example的Bucket，设置其Referer白名单为`http://www.aliyun.com/`。则所有Referer为`http://www.aliyun.com/`的请求才能访问oss-example这个Bucket中的Object。

细节分析

- Referer验证
  - 用户只有通过签名URL或者匿名访问Object时，才会做防盗链验证。请求的Header中有`Authorization`字段的，不会做防盗链验证。
  - Bucket的三种权限（private，public-read，public-read-write）都会做防盗链验证。
- Referer配置
  - 一个Bucket可以支持多个Referer参数。通过控制台设置时使用回车作为换行符分隔，通过API设置时使用英文逗号（,）分隔。
  - Referer参数支持通配符星号（*）和问号（?）。
- Referer效果
  - Referer白名单为空时，不会检查Referer字段是否为空（否则所有的请求都会被拒绝）。
  - 如果Referer白名单不为空，且不允许Referer字段为空，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。
  - 如果白名单不为空，但允许Referer字段为空，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。

通配符详解

- 星号（*）：可以使用星号代替0个或多个字符。如果正在查找以AEW开头的一个文件，但不记得文件名其余部分，可以输入`AEW*`，查找以AEW开头的所有文件类型的文件，如AEWT.txt、AEWU.EXE、AEWI.dll等。要缩小范围可以输入`AEW*.txt`，查找以AEW开头并以.txt为扩展名的文件，如AEWIP.txt、AEWDF.txt。

- 问号（？）：可以使用问号代替一个字符。例如输入`love?`查找以love开头的一个字符结尾的文件，如lovey、lovei等。要缩小范围可以输入`love?.doc`，查找以love开头的一个字符结尾并以.doc为扩展名的文件，如lovey.doc、loveh.doc。

功能使用参考

- 控制台：设置防盗链
- API：*PutBucketReferer*

10.3 设置跨域访问

跨域访问，或者说JavaScript的跨域访问问题，是浏览器出于安全考虑而设置的一个限制，即同源策略。当来自于A网站的页面中的JavaScript代码希望访问B网站的时候，浏览器会拒绝该访问，因为A、B两个网站是属于不同的域。

在实际应用中，经常会有跨域访问的需求，比如用户的网站www.a.com，后端使用了OSS。在网页中提供了使用JavaScript实现的上传功能，但是在该页面中，只能向www.a.com发送请求，向其他网站发送的请求都会被浏览器拒绝。这样就导致用户上传的数据必须从www.a.com中转。如果设置了跨域访问的话，用户就可以直接上传到OSS而无需从www.a.com中转。

跨域资源共享的实现

跨域资源共享（Cross-Origin Resource Sharing），简称CORS，是HTML5提供的标准跨域解决方案，OSS支持CORS标准来实现跨域访问。具体的CORS规则可以参考*W3C CORS规范*。其实现如下：

1. CORS通过HTTP请求中附带Origin的Header来表明自己来源域，比如上面那个例子，Origin的Header就是www.a.com。
2. 服务器端接收到这个请求之后，会根据一定的规则判断是否允许该来源域的请求。如果允许，服务器在返回的响应中会附带上Access-Control-Allow-Origin这个Header，内容为www.a.com来表示允许该次跨域访问。如果服务器允许所有的跨域请求，将Access-Control-Allow-Origin的Header设置为*即可。
3. 浏览器根据是否返回了对应的Header来决定该跨域请求是否成功，如果没有附加对应的Header，浏览器将会拦截该请求。如果是非简单请求，浏览器会先发送一个OPTIONS请求来获取服务器的CORS配置，如果服务器不支持接下来的操作，浏览器也会拦截接下来的请求。

OSS提供了CORS规则的配置，从而根据需求允许或者拒绝相应的跨域请求。该规则是配置在Bucket级别的。详情可以参考*PutBucketCORS*。

细节分析

- CORS相关的Header附加等都是浏览器自动完成的，用户不需要有任何额外的操作。CORS操作也只在浏览器环境下有意义。
- CORS请求的通过与否和OSS的身份验证是完全独立的，即OSS的CORS规则仅仅是用来决定是否附加CORS相关的Header的一个规则。是否拦截该请求完全由浏览器决定。
- 使用跨域请求的时候需要关注浏览器是否开启了Cache功能。当运行在同一个浏览器上分别来源于www.a.com和www.b.com的两个页面都同时请求同一个跨域资源的时候，如果www.a.com的请求先到达服务器，服务器将资源带上Access-Control-Allow-Origin的Header为www.a.com返回给用户。这个时候www.b.com又发起了请求，浏览器会将Cache的上一次请求返回给用户，此时Header的内容和CORS的要求不匹配，就会导致后面的请求失败。

功能使用参考

- 控制台：跨域资源共享
- API：跨域资源共享

10.4 服务器端加密编码

本文介绍如何使用OSS的服务端加密功能，对持久化在OSS上的数据进行加密保护。

OSS支持在服务器端对上传的数据进行加密编码（Server-Side Encryption）：用户上传数据时，OSS对收到的用户数据进行加密编码，然后再将得到的加密数据持久化保存下来；用户下载数据时，OSS自动对保存的加密数据进行解密并把原始数据返回给用户，并在返回的HTTP请求Header中，声明该数据进行了服务器端加密。

OSS服务端加密主要应用场景

OSS通过服务端加密机制，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。针对不同的应用场景，OSS有如下几种服务端加密方式：

- 使用KMS托管密钥进行加解密

  用户上传文件时，可以使用指定的CMK ID或者默认KMS托管的CMK进行加解密操作。这种场景适合于大量的数据加解密。用户数据无需通过网络发送到KMS服务端进行加解密，这是一种低成本的加解密方式。
- 使用OSS完全托管加密

基于OSS完全托管的加密方式，是Object的一种属性。用户创建Object时，只需在put object 请求中携带x-oss-server-side-encryption的HTTP Header，并指定其值为AES256，即可以实现该Object的服务器端加密存储。该方式适合于批量数据的加解密。

服务端加密选项

OSS服务端加密有以下选项，用户可根据需要（密钥管理策略）进行选择：

- 使用由OSS完全托管的服务端加密功能：数据加密密钥的生成和管理，由OSS负责，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使用行业标准的强加密算法AES-256（即256位高级加密标准）。

服务端加密-OSS完全托管

OSS的服务器端加密是Object的一个属性。用户创建一个Object的时候，只需要在Put Object的请求中携带`x-oss-server-side-encryption`的HTTP Header，并指定其值为`AES256`，即可以实现该Object的服务器端加密存储。

以下操作支持在请求中携带这些HTTP Header：

- PutObject：简单上传
- CopyObject：复制Object
- InitiateMultipartUpload：分片上传

服务端加密-KMS托管主密钥

当用户创建Object时，在请求中携带`x-oss-server-side-encryption`的HTTP Header，并指定其值为`KMS`，即可实现在服务端加密存储该Object（密钥管理使用KMS服务）。OSS通过KMS服务，采用信封加密机制、AES-256加密算法等强安全方法，对用户存储在OSS的对象进行加密保护。

KMS是阿里云提供的一款安全、易用的管理类服务。用户无需花费大量成本来保护密钥的保密性、完整性和可用性，借助密钥管理服务，用户可以安全、便捷的使用密钥，专注于开发加解密功能场景。用户可以通过KMS控制台中查看和管理KMS密钥。

以下操作支持在请求中携带这些HTTP Header

- PutObject：简单上传
- CopyObject：复制Object
- InitiateMultipartUpload：分片上传

说明：

- 使用该模式，首次向某地域中的Bucket添加加密对象时，将自动为您在该地域创建一个默认KMS密钥（用作用户主密钥CMK）。此密钥将用于服务端加密（KMS模式）。
- 您可以在KMS控制台查看该密钥信息，请参见*KMS*使用文档。

服务端加密API使用

操作请求

以下API支持请求携带`x-oss-server-side-encryption`头：

- PutObject：简单上传
- CopyObject：复制Object
- InitiateMultipartUpload：分片上传

HTTP Header格式如下：

名称	描述	示例
x-oss-server-side-encryption	指定服务端加密的模式 有效值：AES256、KMS	`x-oss-server-side-encryption:KMS`表示使用服务端加密模式为KMS托管主密钥

说明：

- 其他OSS收到的请求中，如果出现`x-oss-server-side-encryption`头，OSS会直接返回HTTP状态码400，并在消息体内注明错误码为InvalidArgument。
- 如果您指定的`x-oss-server-side-encryption`头不为有效值，OSS会直接返回HTTP状态码400，并在消息体内注明错误码为`InvalidEncryptionAlgorithmError`。

操作响应

通过服务器端加密存储的Object，以下API请求中OSS会返回`x-oss-server-side-encryption`头：

- PutObject
- CopyObject
- InitiateMultipartUpload

- UploadPart
- CompleteMultipartUpload
- GetObject
- HeadObject

Meta信息

通过服务器端加密-KMS托管主密钥模式存储的Object，其Meta信息会增加以下字段：

名称	描述	示例
x-oss-server-side-encryption	指定服务器端加密的模式	`x-oss-server-side-encryption: KMS`
x-oss-server-side-encryption-key-id	该Object加密所用的用户KMS key的ID	`x-oss-server-side-encryption-key-id: 72779642-7d88-4a0f-8d1f-1081a9cc7afb`

相关**API**

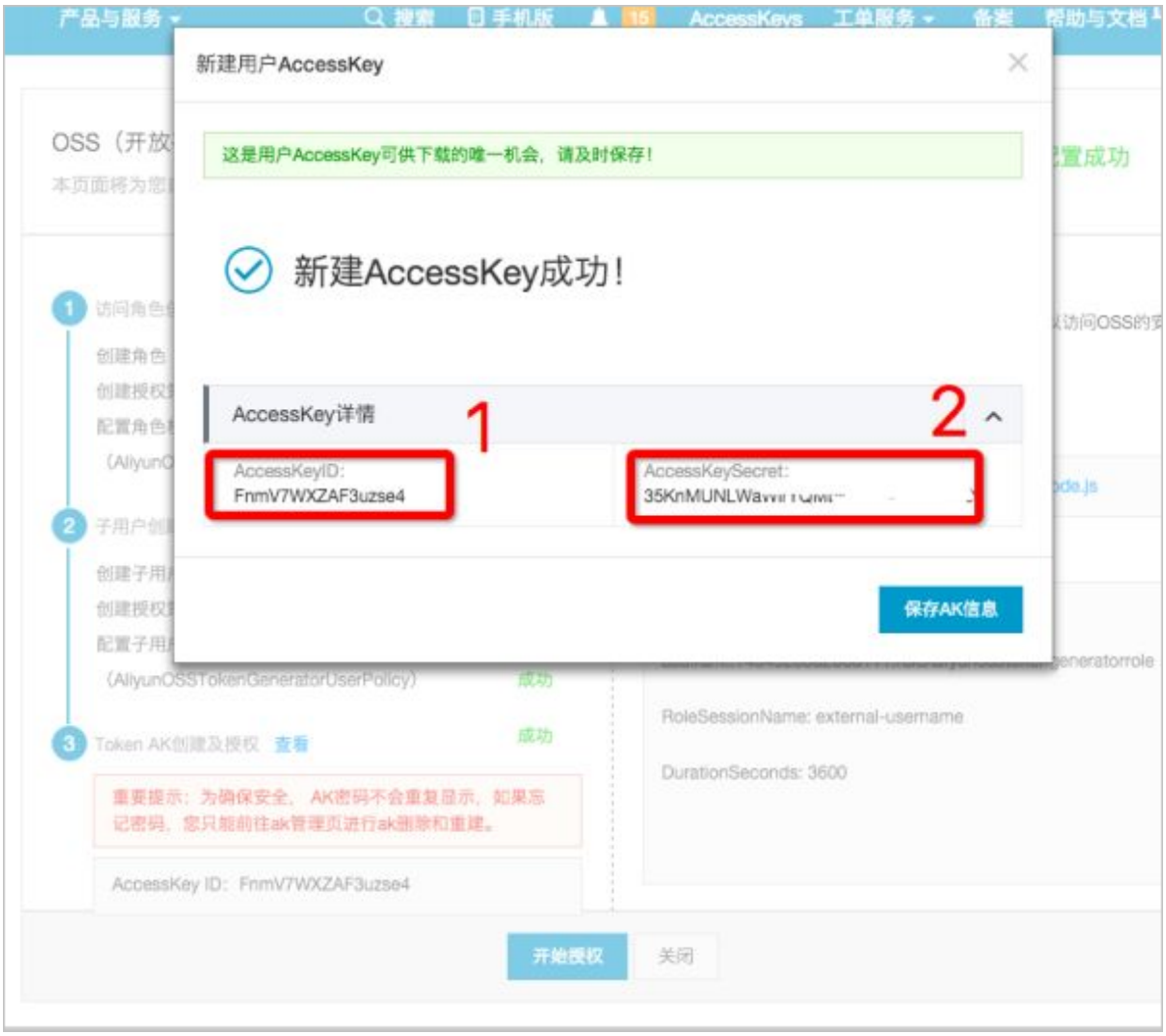
- *AppendObject*
- *PutObject*
- *CopyObject*
- *PostObject*

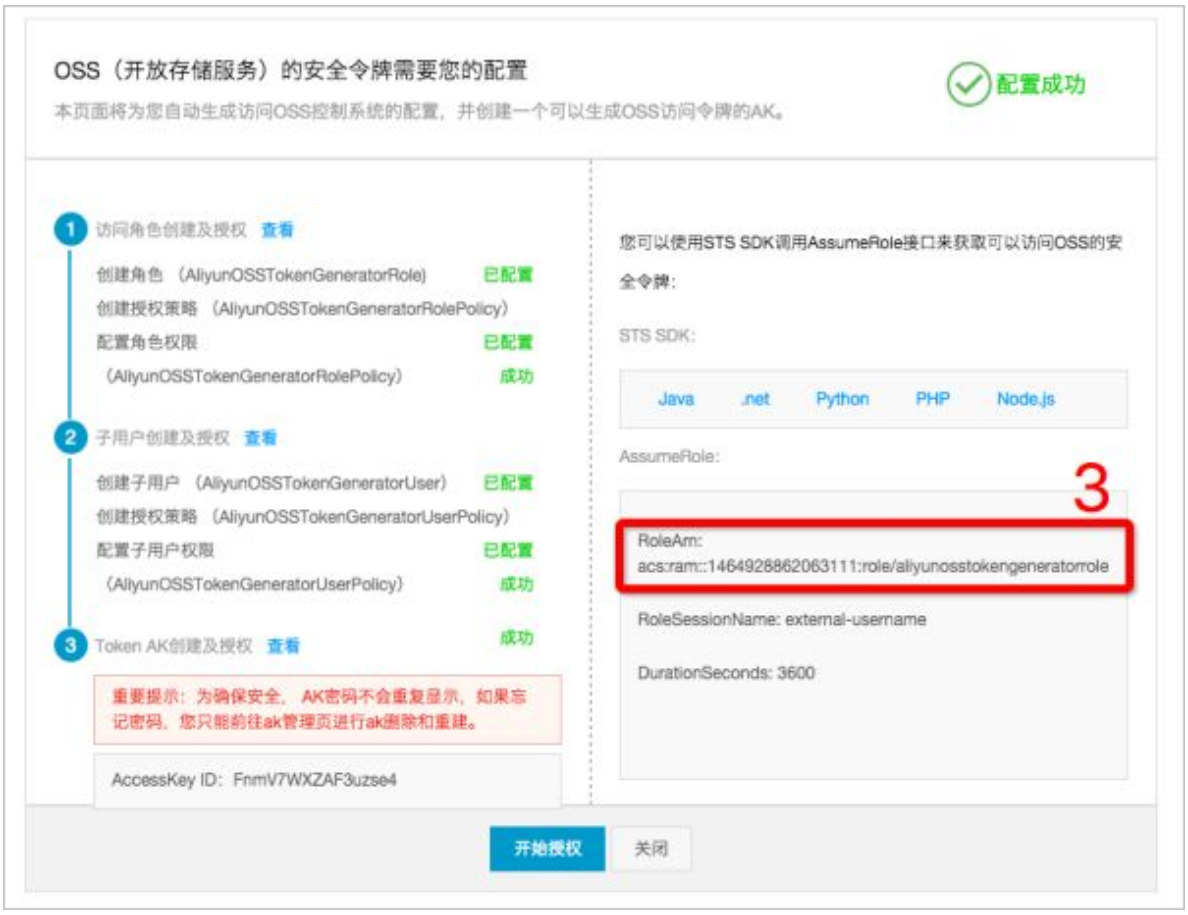
10.5 设置安全令牌

操作步骤

1. 登录*OSS管理控制台*，在OSS概览页中找到基础配置区域，单击安全令牌。
2. 进入到安全令牌快捷配置页面。

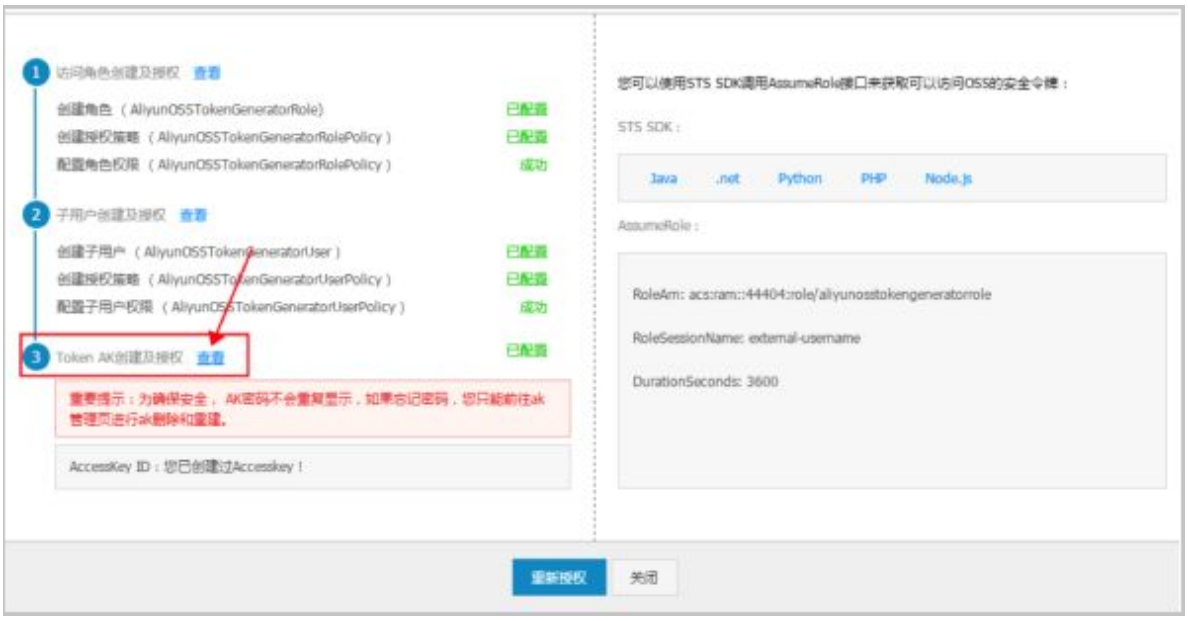
   如果没有开通RAM，会弹出开通的对话框，直接单击开通，并进行实名验证。完成后跳到本页面。单击开始授权。
3. 系统进行自动授权，请务必保存下图中三个红框内的参数。单击保存**AK**信息后，对话框会关闭，STS的开通完成。


新建用户AccessKey
这是用户AccessKey可供下载的唯一机会，请及时保存！
新建AccessKey成功！
AccessKey详情
1
2
AccessKeyID:
FnmV7WXZAF3uzse4
AccessKeySecret:
保存AK信息
RoleSessionName: external-username
DurationSeconds: 3600
重要提示：为确保安全，AK密码不会重复显示，如果忘记密码，您只能前往ak管理页进行ak删除和重建。
AccessKey ID: FnmV7WXZAF3uzse4
开始授权
关闭



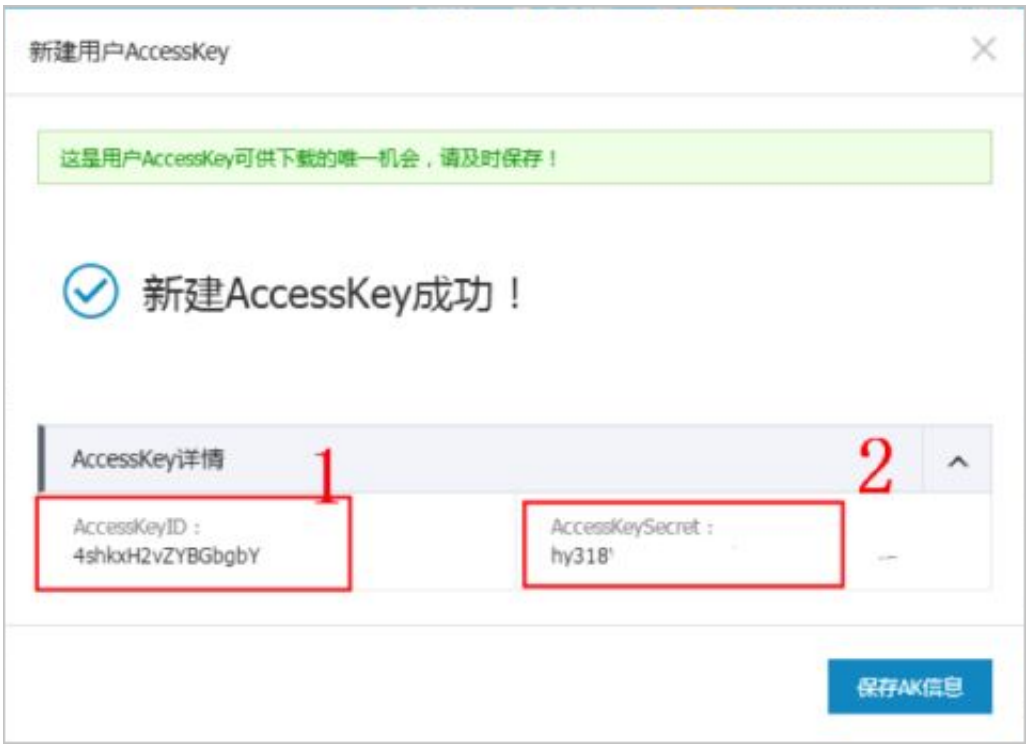
4. 如果您之前已经创建了AccessKeyId/AccessKeySecret， 打开的页面如下：

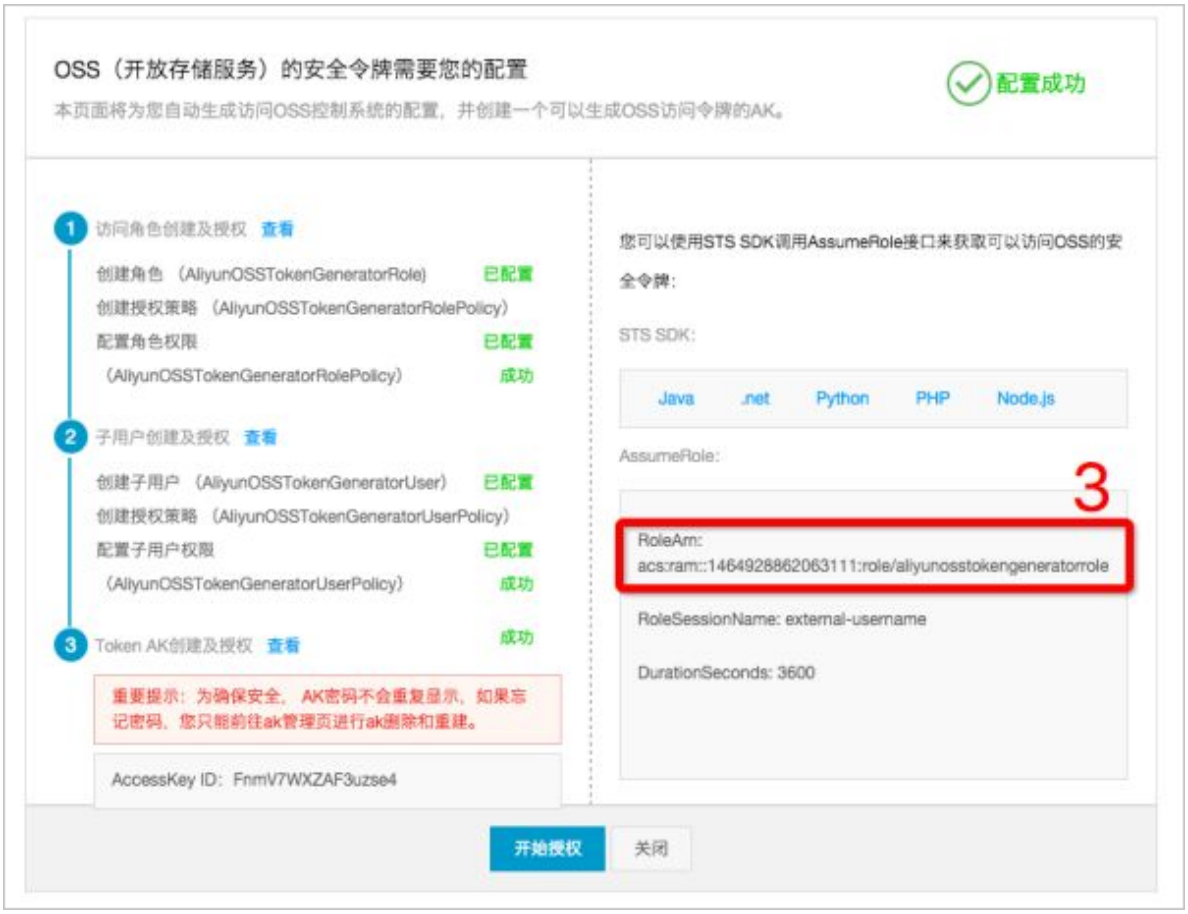
您可以单击如下图所示的查看。



5. 单击如下图所示的创建**AccessKey**。

记下如下参数1、2、3。





保存这三个参数后，STS的开通已经完成了。

11 静态网站托管

11.1 配置静态网站托管

OSS支持静态网站托管。您可以通过*OSS* 控制台将自己的存储空间配置成静态网站托管模式。

配置生效后，假如这个Bucket在杭州，那么这个静态网站的访问域名为：

```
http://<Bucket>.oss-cn-hangzhou.aliyuncs.com/
```

为了使您更方便地管理在OSS上托管的静态网站，OSS支持两种功能：

- 静态页面支持（Index Document Support）

  静态页是指当用户直接访问静态网站根域名时，OSS返回的默认静态页（相当于网站的index.html）。如果您为一个Bucket配置了静态网站托管模式，就必须指定一个静态页。
- 错误页面支持（Error Document Support）

  错误页面是指在用户访问该静态网站时，如果遇到HTTP 4XX错误时（最典型的是404 NOT FOUND错误），OSS返回给用户的错误页面。通过指定错误页面，您可以为您的用户提供恰当的出错提示。

例如：设置索引页面为index.html，错误页面为error.html，Bucket为oss-sample，Endpoint为oss-cn-hangzhou.aliyuncs.com，那么：

- 用户访问`http://oss-sample.oss-cn-hangzhou.aliyuncs.com/`和`http://oss-sample.oss-cn-hangzhou.aliyuncs.com/directory/`的时候，相当于访问`http://oss-sample.oss-cn-hangzhou.aliyuncs.com/index.html`。
- 用户访问`http://oss-sample.oss-cn-hangzhou.aliyuncs.com/object`的时候，如果object不存在，OSS会返回`http://oss-sample.oss-cn-hangzhou.aliyuncs.com/error.html`。

细节分析

- 所谓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。OSS不支持涉及到需要服务器端处理的内容，例如PHP，JSP，APS.NET等。
- 如果您想使用自己的域名来访问基于Bucket的静态网站，可以通过绑定自定义域名*CNAME*来实现。
- 将一个Bucket设置成静态网站托管模式时：

索引页面是必须指定的，错误页面是可选的。

指定的索引页面和错误页面必须是该Bucket内的Object。

- 将一个Bucket设置成静态网站托管模式后：

  对静态网站根域名的匿名访问，OSS将返回索引页面；对静态网站根域名的签名访问，OSS将返回GetBucket结果。

  访问静态网站根域名或者访问不存在的Object时，OSS会返回给用户设定的Object，对此返回的流量和请求将会计费。

功能使用参考

- 控制台：静态网站托管
- API：*PutBucketWebsite*
- Java SDK：静态网站托管

11.2 教程示例：使用自定义域名设置静态网站托管

假设您要在阿里云 OSS 上托管静态网站。您注册了域名（例如，examplewebsite.com）并且想要从 OSS 内容响应对 `http://examplewebsite.com` 和 `http://www.examplewebsite.com` 的请求。不论您是已经有了要在 OSS 上托管的静态网站，还是要从头开始创建，都可以通过以下示例了解如何在阿里云 OSS 上托管网站。

准备工作

本教程涉及到以下服务：

- 域名注册

  如果您没有注册域名（如`examplewebsite.com`），请选择一个注册商进行注册。阿里云也提供域名注册服务。详情请参考阿里云域名服务。

- 阿里云 OSS

  您可以使用阿里云 OSS 创建 bucket，上传示例网页，配置权限允许所有人查看内容，然后配置 bucket 的网站托管功能。在本示例中，因为要允许对 `http://examplewebsite.com` 和 `http://www.examplewebsite.com` 的请求，所以需要创建两个 bucket。您可以仅在一个 bucket 中托管内容，再配置另一个 bucket 将请求重定向到托管内容的 bucket。

- 云解析

您可以将阿里云解析配置为域名解析服务 (DNS) 提供商。在本示例中，您可以将域名添加到云解析并定义 CNAME 记录，这样能够使用您的域名而不是 OSS 分配的访问域名来访问 OSS bucket。

- 在本示例中，我们使用阿里云解析。但您可以通过大多数注册商定义指向 OSS bucket 的 CNAME 记录。

> 说明：
>
> 本教程使用 `examplewebsite.com` 作为域名。请使用您注册的域名替换此域名。

步骤 1：注册域名

如果您已有注册域名，可以跳过该步骤。如果您从未托管过网站，请先注册一个域名，例如 `examplewebsite.com`。您可以使用阿里云域名服务注册一个域名。

详情请参考购买域名。

步骤 2：创建和配置 **bucket** 并上传数据

您可以创建两个 bucket，以同时支持来自根域名（如 `examplewebsite.com`）和子域名（如 `http://www.examplewebsite.com`）的请求。一个 bucket 用于存储内容，另一个 bucket 用于将请求重定向到存储内容的 bucket。

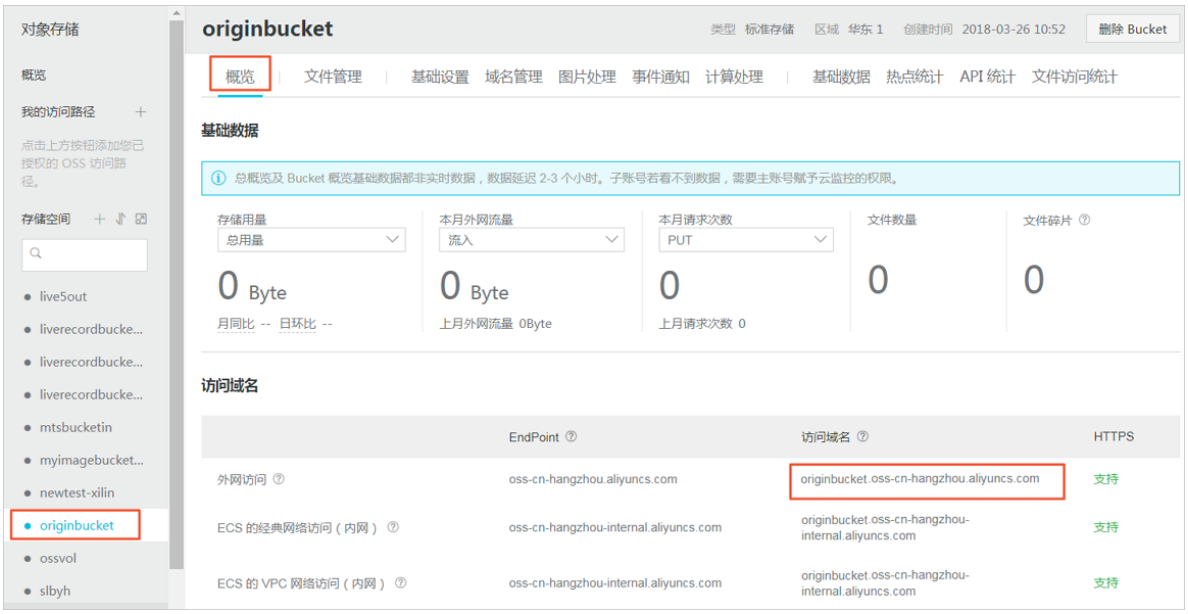
步骤 2.1：创建两个 bucket

在该步骤中，使用阿里云帐户登录 OSS 控制台，并创建以下两个 bucket：

- **originbucket**：用于存储内容
- **redirectbucket**：用于将请求重定向到**originbucket**

1. 登录 *OSS* 控制台。
2. 创建两个 bucket（例如**originbucket**和**redirectbucket**）。将两个 bucket 的读写权限设置为公共读取，以便所有人都可以查看 bucket 的内容。

   详细流程，请参考创建 *bucket*。
3. 记下**originbucket**和**redirectbucket**的访问域名。在后续步骤中将会用到它们。您可以在 bucket 的概览页面上找到 bucket 的访问域名，如下图所示。



4. 将网站数据上传到 **originbucket**。

   您将在根域 bucket **originbucket** 外托管内容，并且将针对子域 bucket **redirectbucket** 的请求重定向到根域 bucket **originbucket**。您可以在任一 bucket 中存储内容。

   本示例将内容托管在**originbucket** bucket 中。内容可以是任何类型的文件，例如文本文件、照片和视频。如果您尚未创建网站，您在本示例中仅需要两个文件。一个文件用作网站首页，另一个文件用作网站的错误页面。

   例如，使用以下 HTML 创建一个名为 index.html 的文件，并将其上传到 bucket。在后续步骤中，将使用此文件名作为网站的默认首页。

```
<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>
```

   您可以使用以下 HTML 创建另一个名为error.html 的文件，并将其上传到bucket。此文件用作网站的 404 错误页面。在后续步骤中，将使用此文件名作为网站的默认404 页面。

```
<html>
<head>
   <title>Hello OSS!</title>
   <meta charset="utf-8">
</head>
<body>
```

```
    <p>这是 404 错误页面</p>
</body>
</html>
```

步骤 2.2：配置 bucket 的网站托管功能

在配置 bucket 的网站托管功能后，可以通过 OSS 分配的访问域名访问该网站。

在此步骤中，将 originbucket 配置为网站。

1. 登录 *OSS* 控制台。
2. 从 bucket 名称列表中，选择 `originbucket`。
3. 单击**基础设置**页签，找到**静态页面**区域。
4. 单击**设置**，然后输入以下信息：
   - **默认首页**：索引页面（相当于网站的 index.html）。只能使用已存储在 bucket 中的 HTML 文件。在本示例中，输入*`index.html`* 。
   - **默认 404 页**：当访问错误路径时，返回默认的 404 页面。只能使用已存储在 bucket 中的 HTML 和图像文件。在本示例中，输入 *`error.html`*。
5. 单击**保存**。

步骤 2.3：配置索引页面的重定向功能

配置 **originbucket**的默认首页和错误页面之后，您还需要配置**redirectbucket** 的默认首页。

参考以下步骤配置索引页面的重定向功能：

1. 登录 *OSS* 控制台。
2. 从 bucket 名称列表中，选择 `redirectbucket`。
3. 单击**基础设置**页签，找到**静态页面**区域。
4. 单击**设置**，然后在**默认首页**文本框中输入 *`index.html`*。
5. 单击**保存**。

步骤 3：将域名绑定到 **OSS bucket**

现在，您已有了根域名`examplewebsite.com`和OSS bucket**originbucket**，接下来您需要将域名绑定到 OSS bucket，以便能够使用您的域名而不是 OSS 分配的域名来访问 OSS bucket。

在本示例中，在将域名`examplewebsite.com`绑定到OSS bucket**originbucket**之前，必须使用 OSS 分配的域名 originbucket.oss-cn-beijing.aliyuncs.com 访问 bucket originbucket。在绑定域名 `examplewebsite.com` 后，可以使用此 `examplewebsite.com` 来访问 OSS bucket。

同样，还需要将子域名`www.examplewebsite.com`绑定到OSS bucket**redirectbucket**，以便能够使用`www.examplewebsite.com`而不是 OSS 分配的域名 originbucket.oss-cn-beijing.aliyuncs.com 来访问 OSS bucket。

参考以下步骤将根域名 `examplewebsite.com`绑定到OSS bucket**originbucket**：

1. 登录 *OSS* 控制台。
2. 从 bucket 名称列表中，选择 `originbucket`。
3. 单击**域名管理**页签。
4. 单击**绑定用户域名**。
5. 在**用户域名**文本框中，输入根域名`examplewebsite.com`。
6. 将 CNAME 记录定义为 originbucket。
   - 如果已在阿里云帐户下解析域名，则可以打开**自动添加 CNAME 记录**开关。然后单击**提交**。
   - 如果未在阿里云主帐户下解析域名，**自动添加 CNAME 记录**开关处于禁用状态。执行以下步骤，手动添加 CNAME 记录，然后单击**提交**。
     1. 在云解析中添加域名。
        - 如果域名是在阿里云中注册的，它会自动添加到云解析列表中。您可以跳过该步骤。
     2. 在云解析控制台中，找到域名。
     3. 单击域名。
     4. 单击**添加解析**。
     5. 在**添加解析**对话框中，从**记录类型**下拉框中选择CNAME，然后在**记录值**文本框中输入 bucket 的 OSS 域名。在本示例中，输入**originbucket.oss-cn-beijing.aliyuncs.com**。
     6. 单击**确认**。
7. 参考上述步骤，将子域名`www.examplewebsite.com`绑定到 OSS bucket**redirectbucket**。

步骤 4：配置网站重定向功能

您已配置 bucket 的网站托管功能并将自定义域名绑定到 OSS bucket，接下来需要配置**redirectbucket**，将`http://www.examplewebsite.com`的所有请求重定向到 `http://examplewebsite.com`。

参考以下步骤配置网站的重定向功能：

1. 登录*OSS* 控制台。
2. 从 bucket 列表中，选择`redirectbucket`。

3. 单击**基础设置**页签，找到**镜像回源**区域。
4. 单击**设置**，然后单击**创建规则**。
5. 创建 404 重定向规则，具体步骤如下：
   a. 在**回源类型**区域，选择**重定向**。
   b. 在**回源条件**区域，设置HTTP 状态码为**404**。
   c. 在**回源地址**区域，选择**添加前后缀**，输入**originbucket**的域名。在本示例中，输入**examplewebsite.com**。
   d. 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 让您的网站文件（如 HTML、图像和视频）可供全球各地的数据中心（即，边缘节点）使用。当访问者从您的网站请求文件时，CDN 自动将请求重定向到最近边缘节点上的文件副本。因此下载速度要快于访问者从较远的数据中心请求内容。

CDN 在边缘节点缓存内容的时间长度由您指定。如果访问者请求的内容的缓存时间超过了到期日期，CDN 会检查源服务器，看看是否有新版本的内容可用。如果有新版本，CDN 将新版本复制到边缘节点。您对原始内容所做的更改会在访问者请求内容时复制到边缘节点。但在到期日期前，内容仍为之前的版本。我们建议打开**CDN 缓存自动刷新**开关，以便您对原始内容所做的更改可以在 CDN 缓存中自动实时刷新。

参考以下步骤使用 CDN 加快**originbucket**速度：

1. 在 CDN 控制台中添加 CDN 加速域名，并复制该域名的 CNAME 地址。详情请参考*CDN 快速入门*中的步骤 2：添加加速域名。
2. 在云解析控制台中定义 CNAME 记录。详情请参考配置*CNAME*流程。
3. 打开**CDN 缓存自动刷新**开关。
   a. 登录 *OSS 控制台*。
   b. 从 bucket 列表中，选择 `originbucket`。
   c. 单击**域名管理**页签。
   d. 打开**CDN 缓存自动刷新**开关。
4. 参考上述步骤，使用 CDN 为`redirectbucket`加速。

步骤 6：测试网站

要验证网站是否正常运行，请在浏览器中尝试以下 URL：

URL	结果
`http://examplewebsite.com`	显示**originbucket**中的索引文档。
originbucket中不存在的文件的 URL，例如 `http://examplewebsite.com/abc`	显示**originbucket**中的错误页面。
`http://www.examplewebsite.com`	将您的请求重定向到`http://examplewebsite.com`。
`http://www.examplewebsite.com/abc`	将您的请求重定向到`http://examplewebsite.com/abc`。

 说明：

某些情况下，可能需要清除 Web 浏览器的缓存才能看到预期行为。

清理

如果您创建静态网站只是为了练习，别忘了删除分配的阿里云资源以免产生不必要的费用。删除阿里云资源后，网站不再可用。

参考以下步骤进行清理：

1. 禁用阿里云 CDN 控制台中的域名，然后将其删除。
2. 删除云解析控制台中的 CNAME 记录。
3. 删除阿里云 OSS 控制台中的 OSS 文件和 bucket。

12 云端数据处理

图片处理

图片处理（Image Processing，简称IMG）是阿里云OSS对外提供的海量、安全、低成本、高可靠的图片处理服务。您可以将原始图片上传保存在OSS上，通过简单的 RESTful 接口，在任何时间、任何地点、任何互联网设备上对图片进行处理。IMG提供了图片处理功能，还提供了图片水印、管道、图片样式等操作。图片处理服务提供图片处理接口，图片上传请使用OSS上传接口。基于IMG，您可以搭建出跟图片相关的服务。

图片服务提供以下功能：

- 图片缩放、裁剪、旋转
- 图片添加图片、文字、图文混合水印
- 图片格式转换
- 自定义图片处理样式
- 通过管道顺序调用多种图片处理功能
- 获取图片信息

更多功能及详细介绍请参见图片处理文档。

媒体处理

媒体处理是为多媒体数据提供的转码计算服务。它以经济、易用、弹性和高可扩展的音视频转换方法，帮助您将存储于OSS的音视频转码成适合在PC、TV以及移动终端上播放的格式。

媒体处理基于阿里云云计算服务构建，它改变了以往进行转码时需要购买、搭建、管理转码软硬件的高昂投入以及配置优化、转码参数适配等复杂性问题。同时，借助云计算服务的弹性伸缩特性，可以按需提供转码能力，从而最大限度的满足业务转码需求、避免资源浪费。

媒体处理功能包括Web管理控制台、服务API和软件开发工具包。您可以通过它们使用和管理媒体处理服务，也可以将媒体处理功能集成到您自己的应用和服务中。

媒体处理功能包括：

- 转码
- 管道
- 截图
- 媒体信息

- 水印
- 预置模版
- 自定义模版
- 剪辑输出
- 分辨率按比例缩放
- M3U8输出自定义切片时长
- 音视频抽取
- 视频画面旋转
- 视频转GIF