OSS云存储

(讲师：程道)

主要课程内容

阿里云对象存储服务(Object Storage Service 简称OSS) 是阿里云提供的海量 安全 低成本 高可靠的云存储服务。

- **第一部分：阿里云OSS云存储简介**

  什么是阿里云存储服务 OSS与自建存储对比的优势 应用场景 计量计费

- **第二部分：OSS云存储基本概念**

  存储空间 对象 Region Endpoint AccessKey Service

- **第三部分：OSS功能详解**

  基本功能(创建存储空间 上传文件 下载文件 删除文件 删除存储空间)

  外链地址规则 OSS防盗链 自定义域名CNAME 访问日志

- **第四部分：OSS云存储的权限控制**

  ACL RAM BucketPolicy

- **第五部分 : OSS存储开放接口规范 和 错误响应**

  Service Bucket Object 分块上传 跨域资源共享 Live Channel 错误响应

- **第六部分 : OSS云储存实战**

  java访问OSS SpringBoot 访问OSS 数据处理(图片 音视频 IMM) CDN加速

一.阿里云OSS云存储简介

1.什么是阿里云oss云存储

阿里云对象存储服务（Object Storage Service，简称 OSS），是阿里云提供的海量、安全、低成本、高可靠的云存储服务。其数据设计持久性不低于 99.9999999999%（12 个 9），服务设计可用性（或业务连续性）不低于 99.995%。

可以使用阿里云提供的 API、SDK 接口或者 OSS 迁移工具轻松地将海量数据移入或移出阿里云 OSS。数据存储到阿里云 OSS 以后，可以选择标准存储（Standard）作为移动应用、大型网站、图片分享或热点音视频的主要存储方式，也可以选择成本更低、存储期限更长的低频访问存储（Infrequent Access）和归档存储（Archive）作为不经常访问数据的存储方式。

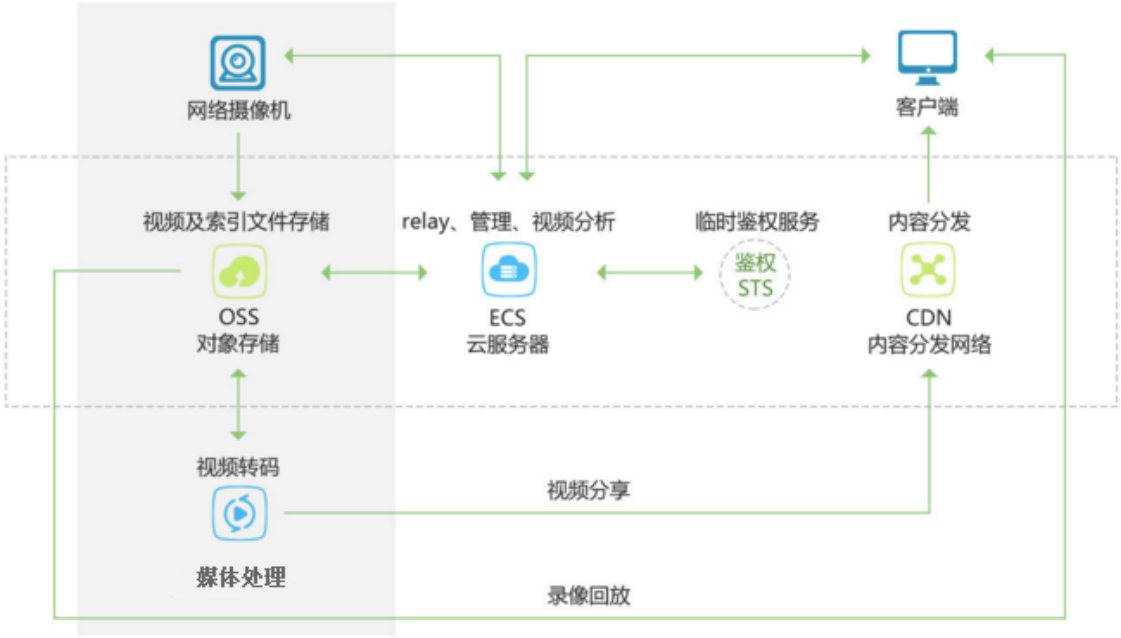
OSS 具有与平台无关的 RESTful API 接口，可以在任何应用、任何时间、任何地点存储和访问任意类型的数据。

2. OSS与自建存储对比的优势

对比项	对象存储OSS	自建服务器存储
可靠性	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的多重冗余架构设计，为数据持久存储提供可靠保障。同时，OSS基于高可用架构设计，消除单节故障，确保数据业务的持续性。服务设计可用性不低于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。	受限于硬件可靠性，易出问题，一旦出现磁盘坏道，容易出现不可逆转的数据丢失。人工数据恢复困难、耗时、耗力。
安全	提供企业级多层次安全防护，包括服务端加密、客户端加密、防盗链、IP黑白名单、细粒度权限管控、日志审计、WORM特性等。多用户资源隔离机制，支持异地容灾机制。获得多项合规认证，包括SEC和FINRA 等，满足企业数据安全与合规要求。	需要另外购买清洗和黑洞设备。需要单独实现安全机制。
成本	多线BGP骨干网络，无带宽限制，上行流量免费。无需运维人员与托管费用，0成本运维。	存储受硬盘容量限制，需人工扩容。单线或双线接入速度慢，有带宽限制，峰值时期需人工扩容。需专人运维，成本高。
智能存储	提供多种数据处理能力，如图片处理、视频截帧、文档预览、图片场景识别、人脸识别、SQL就地查询等，并无缝对接Hadoop生态、以及阿里云函数计算、EMR、DataLakeAnalytics、BatchCompute、MaxCompute、DBS等产品，满足企业数据分析与管理的需求。	需要额外采购，单独部署。

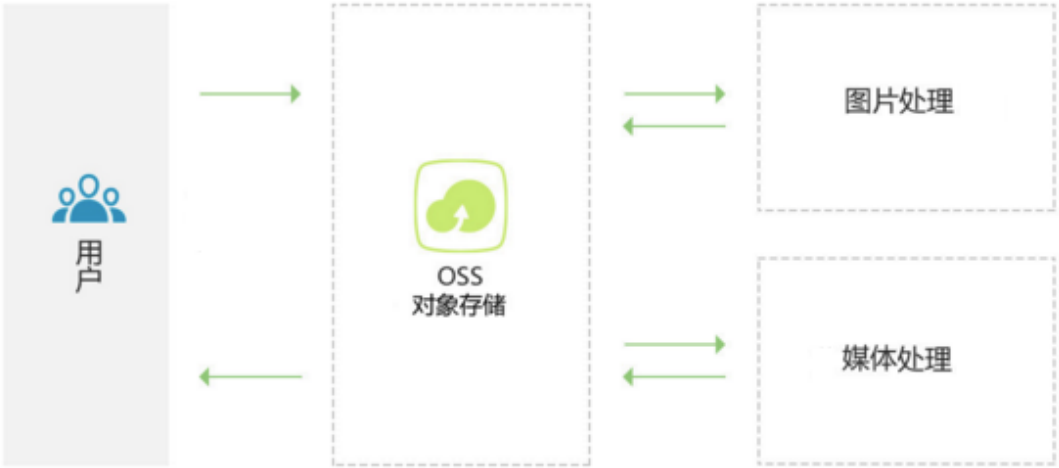
3.应用场景

- **图片和音视频等应用的海量存储**



- **网页或者移动应用的静态和动态资源分离**

- **云端数据处理**



4.计量计费

阿里云对象存储 OSS 服务费用的各项组成部分及计费方式分为按量计费和包年包月两种。

按量付费：按实际使用量*单价的方式计费，每小时统计前一小时的实际用量并从账户余额中扣除实际消费金额。例如当前时间是 9:30，结算的是 8:00-9:00 产生的费用 。

包年包月：预先购买指定资源包，之后使用资源时，扣除相应的额度。一般情况下，包年包月比按量付费更加优惠。资源包目前仅提供标准（LRS）存储包、低频（LRS）存储包、归档（LRS）存储包、标准（ZRS）存储包、低频（ZRS）存储包、下行流量包、 回源流量包、传输加速包，可购买地域请参见购买对象存储 OSS 资源包。

这个看一个实际的计费案例即可

https://help.aliyun.com/document_detail/109686.html?spm=a2c4g.11186623.6.562.4bbd7c28AAnl9D

二.OSS云存储基本概念

1.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，没有文件系统目录的概念,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。

2.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，只有通过追加上传的 Object 可以继续通过追加上传写入数据，其他上传方式上传的 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 字节之间。

- 不能以正斜线（/）或者反斜线（\）开头。

3.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。

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。

4.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。

5.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。

```
注意：
可以登录阿里云官网-“用户中心” -“我的帐户” -“安全认证” 获取 Access Key ID 和 Access Key Secret，一个阿里云帐号可以生成 5 对 Access Key ID 和 Access Key Secret。并支持启用/禁用设置。
```

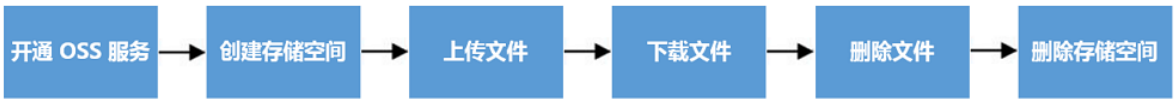
6.Service

OSS 提供给用户的虚拟存储空间，在这个虚拟空间中，每个用户可拥有一个 到多个 Bucket。

三.OSS功能详解

1.基本功能

使用阿里云管理控制台来完成 OSS 基本操作的流程如下：



1)开通OSS服务器

前提条件

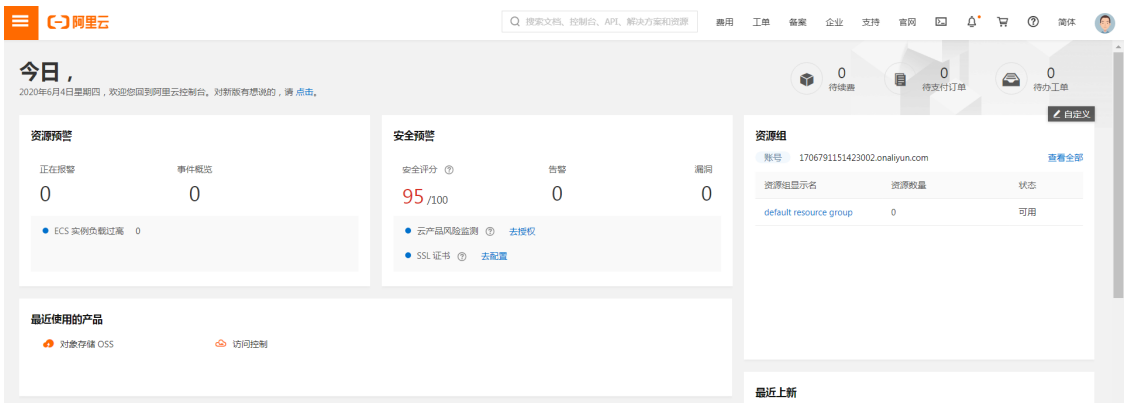
在使用阿里云 OSS 服务之前，请确保您已经注册了阿里云账号并完成实名认证。如果您还没有创建阿里云账号，系统会在您开通 OSS 时提示您注册账号。

操作步骤

1. 登录阿里云官网。
2. 将鼠标移至产品，单击对象存储 OSS，打开 OSS 产品详情页面。
3. 在 OSS 产品详情页，单击立即开通。

1. 开通服务后，在 OSS 产品详情页单击管理控制台直接进入 OSS 管理控制台界面。

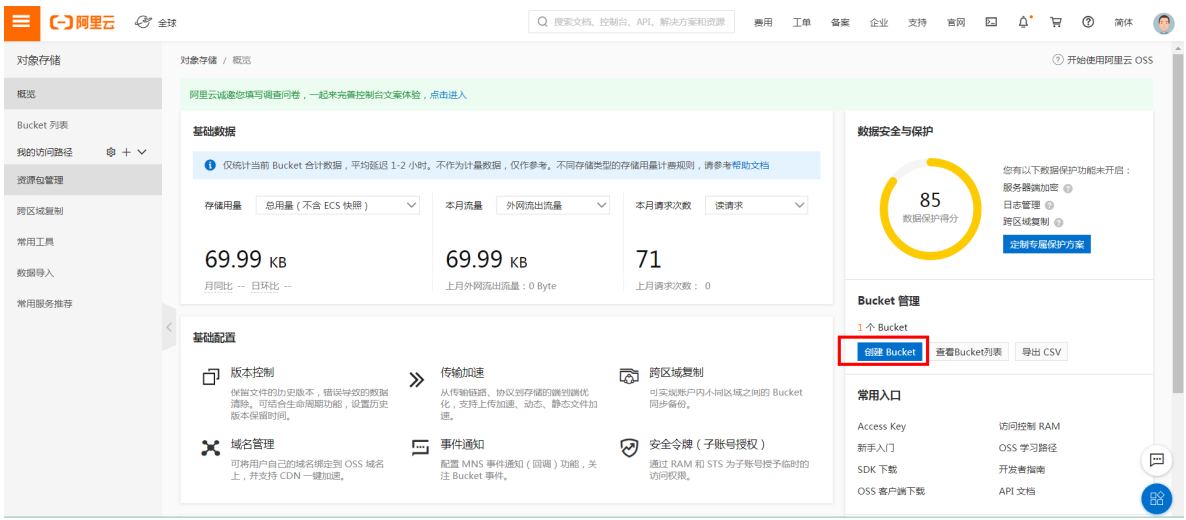
   您也可以单击位于官网首页右上方菜单栏的控制台，进入阿里云管理控制台首页，然后单击左侧的对象存储 OSS 菜单进入 OSS 管理控制台界面。



2)创建存储空间

1. 登录OSS管理控制台。
2. 单击Bucket列表，之后单击创建Bucket。

   您也可以单击概览，之后单击右侧的创建Bucket。
3. 在创建Bucket页面配置Bucket参数。

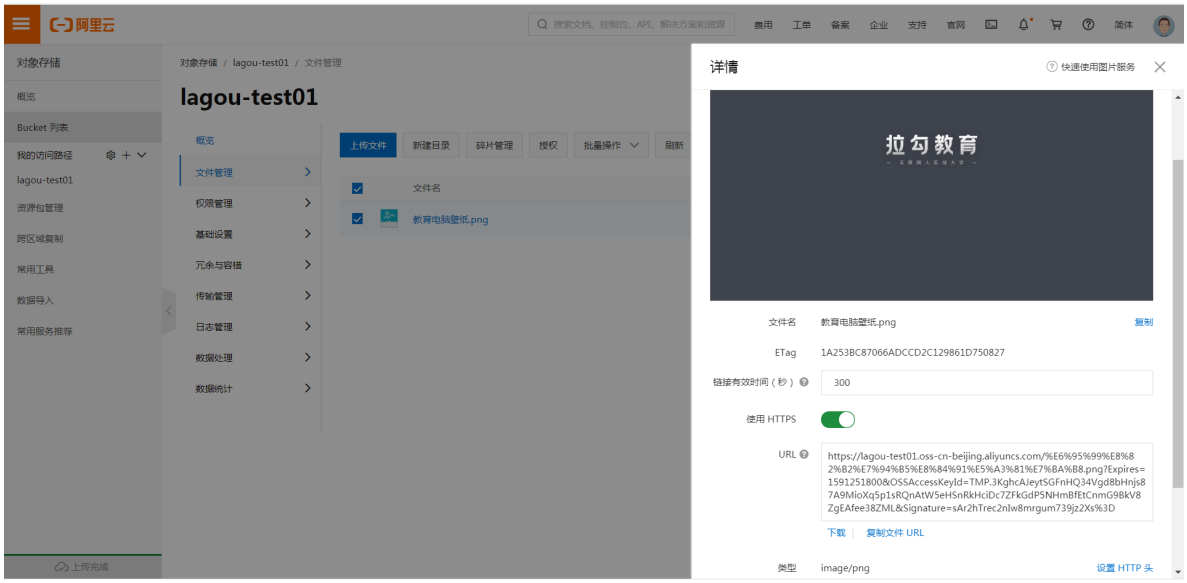


3)上传文件

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。
5. 在上传任务页面等待任务完成，之后关闭对话框。

4)下载文件

1. 登录OSS管理控制台。
2. 进入目标Bucket。
    - 单击Bucket列表，之后单击目标Bucket名称。
3. 单击文件管理页签，您可以进行以下操作：



5)删除文件

1. 登录OSS管理控制台。
2. 进入目标Bucket。
    - 单击Bucket列表，之后单击目标Bucket名称。
3. 选择一个或多个文件，选择批量操作 > 删除。

   您也可以选择目标文件右侧的更多 > 删除来删除单个文件。
4. 在删除文件对话框中，单击确定。

6)删除存储空间

1. 进入OSS管理控制台界面。
2. 进入目标Bucket，找到删除Bucket按钮。
    - 单击基础设置页签，找到Bucket管理区域。

3. 单击删除Bucket。

4. 在弹出的对话框中，单击确定。

2. Object外链地址的构成规则

如果一个bucket设置成公开读权限，意味着允许其他用户来访问属于你的object。你的object的外链地址构成规则如下：

```
http:// <你的bucket名字>.<数据库中心服务域名>/<你的object名字>
```

构成规则的示意图如下：

图 1 OSS 的 URL 构成规则示意图

假设 oss-example 这个 bucket 在青岛数据中心，这个 object 的外链 URL 为：

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

用户可以直接该URL链接放入HTML中使用：

```
<img src="http://oss-example.oss-cn-qingdao.aliyuncs.com/aliyun-logo.png"/>
```

OSS 的短域名（oss.aliyuncs.com）作为 OSS 杭州数据中心的另一个域名仍然可以使用，但只能处理 bucket 属于杭州数据中心的请求。 为了统一起见，本 文接下来章节的例子中， 都将以 oss-cn-hangzhou 域名为例。 请用户根据自己 bucket 所属的数据中心自行替换请求所访问的 Host。

```
注意:在使用 OSS 时，请一直使用 OSS 服务域名，而不要使用固定的 IP 地址。
```

3.OSS防盗链

OSS 是按使用收费的服务， 为了防止用户在 OSS 上的数据被其他人盗链， OSS 支持基于 HTTP header 中表头字段 referer 的防盗链方法。通过 OSS 的控制台--权限管理--防盗链,可以对一个 bucket 设置 referer 字段的白名单和是否允许 referer 字段为空的请求访问。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：

1) 用户只有通过 URL 签名或者匿名访问 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。

2) 一个 bucket 可以支持多个 referer 参数，这些参数之间由“，”号分隔。oss控制台配置时 使用换行

3) Referer 参数支持通配符“*”和“？”。

4) 用户可以设置是否允许 referer 字段为空的请求访问。

5) 白名单为空时，不会检查 referer 字段是否为空（不然所有的请求都会被拒绝）。

6) 白名单不为空，且设置了不允许 referer 字段为空的规则；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。

7) 如果白名单不为空，但设置了允许 referer 字段为空的规则；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。

8) Bucket 的三种权限（private， public-read， public-read-write）都会检查 referer 字段。

4.自定义域名绑定(CNAME)

OSS 支持用户将自定义的域名绑定在属于自己的 bucket 上面，这个操作必 须通过 OSS 控制台（http://oss.aliyun.com）-“Bucket 属性 传输管理 ” -“绑定域名”页面 配置来实现。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，必须提供阿里云备案号，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。

CNAME 应用场景例子：

- 用户 A 拥有一个域名为 abc.com 的网站；这个网站的所有图片存储在 img.abc.com 这个子域名下；
- 为了应对日益增长的图片流量压力， 用户 A 在 OSS 上创建了一个名为 abc-img 的 bucket， 并将所有图片存在 OSS 上；
- 通过 OSS 控制台， 提交将 img.abc.com CNAME 成 abc-img.oss-cn-hangzhou.aliyuncs.com 的申请，并提供相应的材料
- 通过阿里云审核后，在自己的域名服务器上，添加一条 CNAME 规则（https://dns.console.aliyun.com/?spm=a2c4g.11186623.2.12.70c759cbsvcLcR#/dns/domainList）， 将 img.abc.com 映射成 abc-img.oss-cn-hangzhou.aliyuncs.com，这样所有对 img.abc.com 的访问都将变成访问 abc-img 这个 bucket。例如：一个对 http://img.abc.com/logo.png 的访问，实际上访问的是 http://abc-img.oss-cn-hangzhou.aliyuncs.com/logo.png

5.访问日志记录(Server_Access_Logging)

OSS为用户提供自动保存访问日志记录功能。Bucket的拥有者可以通过OSS控制台（http://oss.aliyun.com）日志管理，为其所拥有的bucket开启访问日志记录功能。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写入用户指定的bucket（目标Bucket，Target Bucket）。

存储访问日志记录的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；*YYYY*, *mm*, *DD*, *HH*, *MM*和*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。一个实际的用于存储OSS访问日志的Object名称例子如下：

```
MyLog-oss-example-2012-09-10-04-00-00-0000
```

上例中，“MyLog-”是用户指定的Object前缀；“oss-example”是源bucket的名称；“2012-09-10-04-00-00”是该Object被创建时的北京时间；“0000” 是OSS系统生成的字符串。

LOG文件格式（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
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

名 称	例 子	含 义
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）
Referrer	http://oss.aliyun.com	请求的HTTP Referrer
User-Agent	curl/7.15.5	HTTP的User-Agent头
HostName	oss-example.oss.aliyuncs.com	请求访问域名
Request ID	505B01695037C2AF032593A4	用于唯一标示该请求的UUID
LoggingFlag	true	是否开启了访问日志功能
Reserved	-	保留字段
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
UserID	1657136103983691	Bucket拥有者ID
Delta DataSize	280	Bucket大小的变化量；若没有变化为“-”

细节分析：

1) 源Bucket和目标Bucket必须属于同一个用户。

2) “*TargetPrefix*”表示存储访问日志记录的object名字前缀，可以为空。

3) 源bucket和目标bucket可以是同一个Bucket，也可以是不同的Bucket；用户也可以将多个的源bucket的LOG都保存在同一个目标bucket内（建议指定不同的TargetPrefix）。

4) 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。

5) OSS生成的Log文件命名规则中的“UniqueString”仅仅是OSS为其生成的UUID，用于唯一标识该文件。

6) 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，用户可以按照普通的Object来操作这些LOG文件。

7) 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。如果你想从海量的访问日志中，标示一个特殊的请求，可以在URL中添加一个“*x-*”开头的query-string参数。

如下:

```
http://oss-example.oss.aliyuncs.com/aliyun-logo.png
http://oss-example.oss.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，你可以通过搜索“*x-user=admin*”，很方便地定位　出经过标记的这个请求。

8) 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。

9) 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。

四.OSS云存储的权限控制

4.1 权限控制方式

针对存放在 Bucket 的 Object 的访问，OSS 提供了多种权限控制方式，包括 ACL、RAM Policy 和 Bucket Policy。

- ACL：OSS 为权限控制提供访问控制列表（ACL）。ACL是基于资源的授权策略，可授予 Bucket 和 Object 访问权限。 可以在创建 Bucket 或上传　Object 时设置 ACL，也可以在创建 Bucket 或上传Object 后的任意时间内修改 ACL。
- 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。
- 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。

4.2 ACL

Bucket ACL

- Bucket ACL 介绍

  Bucket ACL是 Bucket 级别的权限访问控制。目前有三种访问权限：public-read-write，public-read 和 private，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。

操作方式

操作方式	特点
控制台	Web应用程序，直观易用
图形化工具ossbrowser	图形化工具，易操作
Java SDK	丰富、完整的各类语言SDK demo

Object ACL

- Object ACL 介绍

  Object ACL是Object 级别的权限访问控制。目前有四种访问权限：private、public-read、public-read-write、default。PutObjectACL 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。

  Object ACL 的四种访问权限含义如下：

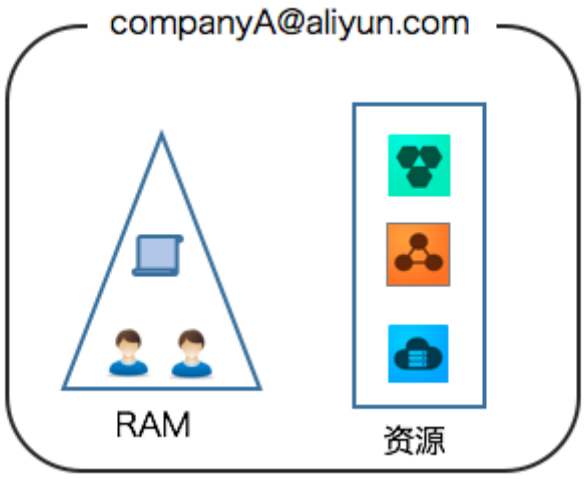
权限值	中文名称	权限对访问者的限制
public-read-write	公共读写	该 ACL 表明某个 Object 是公共读写资源，即所有用户拥有对该 Object 的读写权限。
public-read	公共读，私有写	该 ACL 表明某个 Object 是公共读资源，即非 Object Owner 只有该 Object 的读权限，而 Object Owner 拥有该 Object 的读写权限。
private	私有读写	该 ACL 表明某个 Object 是私有资源，即只有该 Object 的 Owner 拥有该 Object 的读写权限，其他的用户没有权限操作该 Object。
default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

  操作方式和上面类似 参考官方文档即可

https://help.aliyun.com/document_detail/100676.html?spm=a2c4g.11186623.2.9.695b5a5bnz0mcm#concept-blw-yqm-2gb

4.3 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。使用RAM，您可以创建、管理RAM用户，并可以控制这些RAM用户对资源的操作权限。当您的企业存在多用户协同操作资源时，使用RAM可以让您避免与其他用户共享云账号密钥，按需为用户分配最小权限，管理更加方便，权限更加明确，信息更加安全。

[

]

注意

- 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所需的RAM Policy。
- RAM Policy操作比较复杂，强烈推荐您使用简单易用的图形化配置方式Bucket Policy。

https://help.aliyun.com/document_detail/102600.html?spm=a2c4g.11186623.6.693.143258f6f33HST

4.4 Bucket Policy

Bucket Policy是基于资源的授权策略。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拥有者直接可以进行访问授权。

Bucket Policy常见的应用场景如下：

- 向其他账号的RAM用户授权访问。

  您可以授予其他账号的RAM用户访问您的OSS资源的权限。

- 向匿名用户授予带特定IP条件限制的访问权限。

  某些场景下，您需要向匿名用户授予带IP限制的访问策略。例如，企业内部的机密文档，只允许在企业内部访问，不允许在其他区域访问。由于企业内部人员较多，如果针对每个人配置RAM Policy，工作量非常大。此时，您可以基于Bucket Policy设置带IP限制的访问策略，从而高效方便地进行授权。

Bucket Policy的配置方法和教程视频请参见使用Bucket Policy授权其他用户访问OSS资源。

五.OSS存储开放接口规范 和 错误响应

5.1 开放接口规范

开发者在发送请求给 OSS 时，既可以使用 带签名认证的请求，也可以使用匿名访问。OSS提供的相关API接口如下:

关于Service操作

API	描述
getService (listBuckets)	返回请求者拥有的所有Bucket

关于Bucket的操作

API	描述
createBucket	创建Bucket
putBucketACL	设置Bucket访问权限
putBucketLogging	开启Bucket日志
putBucketWebsite	设置Bucket为静态网站托管模式
putBucketReferer	设置Bucket的防盗链规则
putBucketLifecycle	设置Bucket中Object的生命周期规则
getBucket(ListObject)	列出Bucket中所有Object的信息
getBucketAcl	获得Bucket访问权限
getBucketLocation	获得Bucket所属的数据中心位置信息
getBucketInfo	获取Bucket信息
getBucketLogging	查看Bucket的访问日志配置情况
getBucketWebsite	查看Bucket的静态网站托管状态
getBucketReferer	查看Bucket的防盗链规则
getBucketLifecycle	查看Bucket中Object的生命周期规则
deleteBucket	删除Bucket
deleteBucketLogging	关闭Bucket访问日志记录功能
deleteBucketWebsite	关闭Bucket的静态网站托管模式
deleteBucketLifecycle	删除Bucket中Object的生命周期规则
putBucketEncryption	配置Bucket的加密规则
getBucketEncryption	获取Bucket的加密规则
deleteBucketEncryption	删除Bucket的加密规则

关于Object的操作

API	描述
putObject	上传Object
copyObject	拷贝一个Object成另外一个Object
getObject	获取Object
appendObject	在Object尾追加上传数据
deleteObject	删除Object
deleteMultipleObjects	删除多个Object
headObject	只返回某个Object的meta信息，不返回文件内容
getObjectMeta	返回Object的基本meta信息，包括该Object的ETag、Size（文件大小）、LastModified，不返回文件内容
postObject	使用Post上传Object
putObjectACL	设置ObjectACL
getObjectACL	获取ObjectACL信息
callback	上传回调
putSymlink	创建软链接
getSymlink	获取软链接
restoreObject	解冻文件
selectObject	用SQL语法查询Object内容
putObjectTagging	设置或更新对象标签
getObjectTagging	获取对象标签信息
deleteObjectTagging	删除指定的对象标签

关于Multipart Upload的操作

API	描述
InitiateMultipartUpload	初始化MultipartUpload事件
uploadPart	分块上传文件
uploadPartCopy	分块复制上传文件
completeMultipartUpload	完成整个文件的MultipartUpload上传
abortMultipartUpload	取消MultipartUpload事件
listMultipartUploads	罗列出所有执行中的MultipartUpload事件
listParts	罗列出指定UploadID所属的所有已经上传成功Part

跨域资源共享（CORS）

API	描述
putBucketcors	在指定Bucket设定一个CORS的规则
getBucketcors	获取指定的Bucket目前的CORS规则
deleteBucketcors	关闭指定Bucket对应的CORS功能并清空所有规则
optionObject	跨域访问preflight请求

关于Live Channel的操作

API	描述
putLiveChannelStatus	切换LiveChannel的状态
putLiveChannel	创建LiveChannel
getVodPlaylist	获取播放列表
postVodPlaylist	生成播放列表
get LiveChannelStat	获取LiveChannel的推流状态信息
getLiveChannelInfo	获取LiveChannel的配置信息
getLiveChannelHistory	获取LiveChannel的推流记录
listLiveChannel	列举LiveChannel
deleteLiveChannel	删除LiveChannel

案例剖析

从上面的请求中 挑选任意接口分析

createBucket 创建Bucket

putObject 上传Object

getObject 下载Object

5.2 OSS云存储错误响应

当用户访问OSS出现错误时，OSS会返回给用户相应的错误码和错误信息，便于用户定位问题，并做出适当的处理。

1.OSS的错误响应格式

当用户访问OSS出错时，OSS会返回给用户一个合适的3xx，4xx或者5xx的HTTP状态码；以及一个application/xml格式的消息体。

错误响应的消息体例子：

```
<?xml version="1.0" ?>
<Error xmlns="http://doc.oss.aliyuncs.com">
```

```
<Code>
          AccessDenied
</Code>
<Message>
   Query-string authentication requires the Signature, Expires and
OSSAccessKeyId parameters
</Message>
<RequestId>
          1D842BC5425544BB
</RequestId>
<HostId>
          oss.aliyuncs.com
</HostId>
</Error>
```

所有错误的消息体中都包括以下几个元素：

- **Code**：OSS返回给用户的错误码。
- **Message**：OSS给出的详细错误信息。
- **RequestId**：用于唯一标识该次请求的UUID；当你无法解决问题时，可以凭这个RequestId来请求OSS开发工程师的帮助。
- **HostId**：用于标识访问的OSS集群（目前统一为oss.aliyuncs.com）

2.OSS的错误码

OSS的错误码列表如下：

错误码	描述	HTTP**状态码**
AccessDenied	拒绝访问	403
BucketAlreadyExists	Bucket已经存在	409
BucketNotEmpty	Bucket不为空	409
EntityTooLarge	实体过大	400
EntityTooSmall	实体过小	400
FileGroupTooLarge	文件组过大	400
FilePartNotExist	文件Part不存在	400
FilePartStale	文件Part过时	400
InvalidArgument	参数格式错误	400
InvalidAccessKeyId	Access Key ID不存在	403
InvalidBucketName	无效的Bucket名字	400
InvalidDigest	无效的摘要	400
InvalidObjectName	无效的Object名字	400
InvalidPart	无效的Part	400

错误码	描述	HTTP**状态码**
InvalidPartOrder	无效的part顺序	400
InvalidTargetBucketForLogging	Logging操作中有无效的目标bucket	400
InternalError	OSS内部发生错误	500
MalformedXML	XML格式非法	400
MethodNotAllowed	不支持的方法	405
MissingArgument	缺少参数	411
MissingContentLength	缺少内容长度	411
NoSuchBucket	Bucket不存在	404
NoSuchKey	文件不存在	404
NoSuchUpload	Multipart Upload ID不存在	404
NotImplemented	无法处理的方法	501
PreconditionFailed	预处理错误	412
RequestTimeTooSkewed	发起请求的时间和服务器时间超出15分钟	403
RequestTimeout	请求超时	400
SignatureDoesNotMatch	签名错误	403
TooManyBuckets	用户的Bucket数目超过限制	400

3.请求一个需要Referer字段的Object

https://lagou-imgs.oss-cn-beijing.aliyuncs.com/1.png

4.OSS参数不支持的操作

如果在OSS合法的操作中，添加了OSS不支持的请求头参数（例如在PUT的时候，加入If-Modified-Since参数），OSS会返回501 Not Implemented错误

错误请求示例：

```
PUT /my-image.jpg HTTP/1.1
Host:oss-example. oss.aliyuncs.com
Date: Wed, 28 May 2011 22:32:00 GMT
If-Modified-Since: Wed, 06 Apr 2011 10:02:46 GMT
```

返回示例：

```
501 Not Implemented
x-oss-request-id: 77E534EBF90372BE
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Thu, 28 Apr 2011 08:03:07 GMT
Connection: close
Server: AliyunOSS


<?xml version="1.0" ?>
<Error xmlns="http://doc.oss.aliyuncs.com">
<Code>
          NotImplemented
</Code>
<Message>
          A header you provided implies functionality that is not implemented.
</Message>
<Header>
          If-Modified-Since
</Header>
<RequestId>
          77E534EBF90372BE
</RequestId>
<HostId>
          oss.aliyuncs.com
</HostId>
</Error>
```

5.OSS不支持的操作

如果试图以OSS不支持的操作来访问某个资源，返回405 Method Not Allowed错误。

错误请求示例：

```
abc / HTTP/1.1
Host:oss-example. oss.aliyuncs.com
Date: date
Authorization: signatureValue
```

返回示例：

```
x-oss-request-id: 2403382433A2EDA8
Allow: GET, DELETE, HEAD, PUT
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Thu, 31 Mar 2011 10:01:52 GMT
Server: AliyunOSS

<?xml version="1.0" ?>
<Error xmlns="http://doc.oss.aliyuncs.com">
<Code>
          MethodNotAllowed
</Code>
```

```
<Message>
          The specified method is not allowed against this resource.
</Message>
<ResourceType>
          BUCKET
</ResourceType>
<Method>
          abc
</Method>
<RequestId>
          2403382433A2EDA8
</RequestId>
<HostId>
          oss.aliyuncs.com
</HostId>
</Error>
```

六.OSS云储存实战

1.快速入门

1)案例描述

图片上传

2)配置环境

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.lagou</groupId>
    <artifactId>oss-project01</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

     <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
```

```
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.5</version>
        </dependency>
        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.2</version>
        </dependency>

        <dependency>
            <groupId>com.aliyun.oss</groupId>
            <artifactId>aliyun-sdk-oss</artifactId>
            <version>3.8.0</version>
        </dependency>

    </dependencies>

</project>
```

3)图片上传代码实现

```
package com.lagou.oss;

import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

public class FileUpLoad {
    public static void main(String[] args) throws FileNotFoundException {
        // 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";
       String endpoint = "http://oss-cn-beijing.aliyuncs.com";
        // 云账号AccessKey有所有API访问权限，建议遵循阿里云安全最佳实践 创建并使用RAM子账号进行API访问或日常运维，
        String accessKeyId = "LTAI4GD1XcQgrQuvkKnZYhhA";
        String accessKeySecret = "1n2i3VNxd5gWr9YPYuJbVMophsXUQr";

        // 创建OSSClient实例。
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

        // 上传文件流。
        InputStream inputStream = new FileInputStream(new File("src\\main\\resources\\1.png"));
        ossClient.putObject("lagou-imgs", "拉勾教育.png", inputStream);

        // 关闭OSSClient。
        ossClient.shutdown();
```

```
    }
}
```

2.java API操作

1) 需求描述

1.创建Bucket

2.把字符串存入OSS，Object的名称为firstKey

3.下载文件

4.文件存储入OSS

5.查看Bucket中的Object

6.删除Object

2)功能实现

pom.xml

同上一个例子

代码实现

```
package com.lagou.oss;

import java.io.*;
import java.util.List;
import java.util.Properties;

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.BucketInfo;
import com.aliyun.oss.model.OSSObject;
import com.aliyun.oss.model.OSSObjectSummary;
import com.aliyun.oss.model.ObjectListing;

import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class HelloOSS {
    static Logger logger = Logger.getLogger(HelloOSS.class);

    // endpoint是访问OSS的域名。如果您已经在OSS的控制台上 创建了Bucket，请在控制台上查看域名。
    // 如果您还没有创建Bucket，endpoint选择请参看文档中心的“开发人员指南 > 基本概念 > 访问域名”，
```

```
    // 链接地址是：https://help.aliyun.com/document_detail/oss/user_guide/oss_concept/endpoint.html?spm=5176.docoss/user_guide/endpoint_region
    // endpoint的格式形如“http://oss-cn-hangzhou.aliyuncs.com/”，注意http://后不带bucket名称，
    // 比如“http://bucket-name.oss-cn-hangzhou.aliyuncs.com”，是错误的endpoint，请去掉其中的“bucket-name”。
    private static String endpoint = "http://oss-cn-beijing.aliyuncs.com";
    // accessKeyId和accessKeySecret是OSS的访问密钥，您可以在控制台上创建和查看，
    // 创建和查看访问密钥的链接地址是：https://ak-console.aliyun.com/#/。
    // 注意：accessKeyId和accessKeySecret前后都没有空格，从控制台复制时请检查并去除多余的空格。
    private static  String accessKeyId = "LTAI4GD1XcQgrQuvkKnZYhhA";
    private static  String accessKeySecret = "1n2i3VNxd5gWr9YPYuJbVMophsXUQr";
    // Bucket用来管理所存储Object的存储空间，详细描述请参看“开发人员指南 > 基本概念 > OSS基本概念介绍”。
    // Bucket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，长度必须在3-63字节之间。
    private static String bucketName = "lagou-imgs";

    // Object是OSS存储数据的基本单元，称为OSS的对象，也被称为OSS的文件。详细描述请参看“开发人员指南 > 基本概念 > OSS基本概念介绍”。
    // Object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“/”或者“\”字符开头。
    private static String firstKey = "my-first-key";

    public static void main(String[] args) throws IOException {

        // 日志配置，OSS Java SDK使用log4j记录错误信息。示例程序会在工程目录下生成“oss-demo.log”日志文件，默认日志级别是INFO。
        // 日志的配置文件是“conf/log4j.properties”，如果您不需要日志，可以没有日志配置文件和下面的日志配置。
       // PropertyConfigurator.configure("conf/log4j.properties");
        InputStream in = HelloOSS.class.getClassLoader().getResourceAsStream("log4j.properties");
        Properties properties = new Properties();
        properties.load(in);
        PropertyConfigurator.configure(properties);
        logger.info("Started");

        // 生成OSSClient，您可以指定一些参数，详见“SDK手册 > Java-SDK > 初始化”，
        // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-sdk/init.html?spm=5176.docoss/sdk/java-sdk/get-start
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

        try {

            // 判断Bucket是否存在。详细请参看“SDK手册 > Java-SDK > 管理Bucket”。
            // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-sdk/manage_bucket.html?spm=5176.docoss/sdk/java-sdk/init
            if (ossClient.doesBucketExist(bucketName)) {
                System.out.println("您已经创建Bucket：" + bucketName + "。");
            } else {
                System.out.println("您的Bucket不存在，创建Bucket：" + bucketName + "。");
                // 创建Bucket。详细请参看“SDK手册 > Java-SDK > 管理Bucket”。
```

```
                // 链接地址是：
https://help.aliyun.com/document_detail/oss/sdk/java-sdk/manage_bucket.html?
spm=5176.docoss/sdk/java-sdk/init
                ossClient.createBucket(bucketName);
            }

            // 查看Bucket信息。详细请参看“SDK手册 > Java-SDK > 管理Bucket”。
            // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-
sdk/manage_bucket.html?spm=5176.docoss/sdk/java-sdk/init
            BucketInfo info = ossClient.getBucketInfo(bucketName);
            System.out.println("Bucket " + bucketName + "的信息如下：");
            System.out.println("\t数据中心：" + info.getBucket().getLocation());
            System.out.println("\t创建时间：" +
info.getBucket().getCreationDate());
            System.out.println("\t用户标志：" + info.getBucket().getOwner());

            // 把字符串存入OSS，Object的名称为firstKey。详细请参看“SDK手册 > Java-SDK
> 上传文件”。
            // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-
sdk/upload_object.html?spm=5176.docoss/user_guide/upload_object
            InputStream is = new ByteArrayInputStream("Hello OSS".getBytes());
            ossClient.putObject(bucketName, firstKey, is);
            System.out.println("Object：" + firstKey + "存入OSS成功。");

            // 下载文件。详细请参看“SDK手册 > Java-SDK > 下载文件”。
            // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-
sdk/download_object.html?spm=5176.docoss/sdk/java-sdk/manage_object
            OSSObject ossObject = ossClient.getObject(bucketName, firstKey);
            InputStream inputStream = ossObject.getObjectContent();
            StringBuilder objectContent = new StringBuilder();
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
            while (true) {
                String line = reader.readLine();
                if (line == null)
                    break;
                objectContent.append(line);
            }
            inputStream.close();
            System.out.println("Object：" + firstKey + "的内容是：" +
objectContent);

            // 文件存储入OSS，Object的名称为fileKey。详细请参看“SDK手册 > Java-SDK >
上传文件”。
            // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-
sdk/upload_object.html?spm=5176.docoss/user_guide/upload_object
            String fileKey = "README.md";
            ossClient.putObject(bucketName, fileKey, new
File("src\\main\\resources\\README.md"));
            System.out.println("Object：" + fileKey + "存入OSS成功。");

            // 查看Bucket中的Object。详细请参看“SDK手册 > Java-SDK > 管理文件”。
            // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-
sdk/manage_object.html?spm=5176.docoss/sdk/java-sdk/manage_bucket
            ObjectListing objectListing = ossClient.listObjects(bucketName);
            List<OSSObjectSummary> objectSummary =
objectListing.getObjectSummaries();
            System.out.println("您有以下Object：");
```

```
            for (OSSObjectSummary object : objectSummary) {
                System.out.println("\t" + object.getKey());
            }

            // 删除Object。详细请参看“SDK手册 > Java-SDK > 管理文件”。
            // 链接地址是：https://help.aliyun.com/document_detail/oss/sdk/java-sdk/manage_object.html?spm=5176.docoss/sdk/java-sdk/manage_bucket
            ossClient.deleteObject(bucketName, firstKey);
            System.out.println("删除Object：" + firstKey + "成功。");
            ossClient.deleteObject(bucketName, fileKey);
            System.out.println("删除Object：" + fileKey + "成功。");

        } catch (OSSException oe) {
            oe.printStackTrace();
        } catch (ClientException ce) {
            ce.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            ossClient.shutdown();
        }

        logger.info("Completed");
    }
}
```

log4j.properties

```
log4j.rootLogger=INFO, RFA

ossdemo.log.dir=.
ossdemo.log.file=oss-demo.log

# Logging Threshold
log4j.threshold=ALL

# Null Appender
log4j.appender.NullAppender=org.apache.log4j.varia.NullAppender

#
# Rolling File Appender - cap space usage at 5gb.
#
ossdemo.log.maxfilesize=256MB
ossdemo.log.maxbackupindex=20
log4j.appender.RFA=org.apache.log4j.RollingFileAppender
log4j.appender.RFA.File=${ossdemo.log.dir}/${ossdemo.log.file}

log4j.appender.RFA.MaxFileSize=${ossdemo.log.maxfilesize}
log4j.appender.RFA.MaxBackupIndex=${ossdemo.log.maxbackupindex}

log4j.appender.RFA.layout=org.apache.log4j.PatternLayout

# Pattern format: Date LogLevel LoggerName LogMessage
log4j.appender.RFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
# Debugging Pattern format
#log4j.appender.RFA.layout.ConversionPattern=%d{ISO8601} %-5p %c{2} (%F:%M(%L)) - %m%n
```

```
#
# Daily Rolling File Appender
#

log4j.appender.DRFA=org.apache.log4j.DailyRollingFileAppender
log4j.appender.DRFA.File=${ossdemo.log.dir}/${ossdemo.log.file}

# Rollver at midnight
log4j.appender.DRFA.DatePattern=.yyyy-MM-dd

# 30-day backup
#log4j.appender.DRFA.MaxBackupIndex=30
log4j.appender.DRFA.layout=org.apache.log4j.PatternLayout

# Pattern format: Date LogLevel LoggerName LogMessage
log4j.appender.DRFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
# Debugging Pattern format
#log4j.appender.DRFA.layout.ConversionPattern=%d{ISO8601} %-5p %c{2} (%F:%M(%L))
- %m%n


#
# console
# Add "console" to rootlogger above if you want to use this
#

log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{2}:
%m%n
```

```
HelloOSS
"E:\Program Files\JetBrains\IntelliJ IDEA 2019.3.3\jbr\bin\java.exe" ...
您已经创建Bucket：lagou-test01。
Bucket lagou-test01的信息如下：
    数据中心：oss-cn-beijing
    创建时间：Thu Jun 04 11:41:30 CST 2020
    用户标志：Owner [name=1706791151423002,id=1706791151423002]
Object：my-first-key存入OSS成功。
Object：my-first-key的内容是：Hello OSS
Object：README.md存入OSS成功。
您有以下Object：
    <yourObjectName>
    README.md
    my-first-key
    刘备图片
    刘备图片.jpg
    教育电脑壁纸.png
删除Object：my-first-key成功。
删除Object：README.md成功。
```

3.SpringBoot整合OSS上传

1)需求描述

使用SpringBoot完成上传图片

2)配置文件

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.lagou</groupId>
    <artifactId>oss-project02</artifactId>
    <version>1.0-SNAPSHOT</version>


    <!--spring boot的支持-->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.0.RELEASE</version>
    </parent>


    <dependencies>
        <!--springboot 测试支持-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.aliyun.oss</groupId>
            <artifactId>aliyun-sdk-oss</artifactId>
            <version>2.8.3</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.7</version>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.4</version>
        </dependency>
        <dependency>
            <groupId>joda-time</groupId>
            <artifactId>joda-time</artifactId>
            <version>2.9.9</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

    </dependencies>
</project>
```

application.properties

```
# Spring boot application
#server.servlet.context-path=/oss-server
server.port = 8999
#logging.level.root=DEBUG
```

aliyun.properties

```
aliyun.endpoint=http://oss-cn-beijing.aliyuncs.com
aliyun.accessKeyId=LTAI4GD1XcQgrQuvkKnZYhhA
aliyun.accessKeySecret=1n2i3VNxd5gWr9YPYuJbVMophsXUQr
aliyun.bucketName=lagou-imgs
aliyun.urlPrefix=https://lagou-imgs.oss-cn-beijing.aliyuncs.com/
```

3)编码实现

AliyunConfig

```
package  com.lagou.config
import com.aliyun.oss.OSSClient;
import lombok.Data;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;

@Configuration
@PropertySource("classpath:aliyun.properties")
@ConfigurationProperties(prefix = "aliyun")
@Data
public class AliyunConfig {

    private String endpoint;
    private String accessKeyId;
    private String accessKeySecret;
    private String bucketName;
    private String urlPrefix;

    @Bean
    public OSSClient ossClient() {
        return new OSSClient(endpoint, accessKeyId, accessKeySecret);
    }
}
```

实体对象

```java
package com.lagou.bean;
import lombok.Data;

@Data
public class UpLoadResult  {

    // 文件唯一标识
    private String uid;
    // 文件名
    private String name;
    // 状态有：uploading done error removed
    private String status;
    // 服务端响应内容，如：'{"status": "success"}'
    private String response;

}
```

FileUpLoadService

```java
package com.lagou.service;

import com.aliyun.oss.OSSClient;
import com.lagou.bean.UpLoadResult;
import com.lagou.config.AliyunConfig;
import org.apache.commons.lang3.RandomUtils;
import org.apache.commons.lang3.StringUtils;
import org.joda.time.DateTime;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.ByteArrayInputStream;
import java.io.IOException;

@Service
public class FileUpLoadService   {

    @Autowired
    private AliyunConfig aliyunConfig;

    @Autowired
    private OSSClient ossClient;

    // 允许上传的格式
    private static final String[] IMAGE_TYPE = new String[]{".bmp", ".jpg",
            ".jpeg", ".gif", ".png"};

    public UpLoadResult upload(MultipartFile uploadFile) {
        // 校验图片格式
        boolean isLegal = false;
        for (String type : IMAGE_TYPE) {
            if (StringUtils.endsWithIgnoreCase(uploadFile.getOriginalFilename(),
                    type)) {
                isLegal = true;
```

```
                break;
            }
        }

        UpLoadResult uploadResult = new UpLoadResult();
        if (!isLegal) {
            uploadResult.setStatus("error");
            return uploadResult;
        }

        String fileName = uploadFile.getOriginalFilename();
        String filePath = getFilePath(fileName);

        try {
            ossClient.putObject(aliyunConfig.getBucketName(),filePath, new
ByteArrayInputStream(uploadFile.getBytes()));
        } catch (IOException e) {
            e.printStackTrace();
        //上传失败
            uploadResult.setStatus("error");
            return uploadResult;
        }
        uploadResult.setStatus("done");
        uploadResult.setName(this.aliyunConfig.getUrlPrefix() + filePath);
        uploadResult.setUid(String.valueOf(System.currentTimeMillis()));
        return uploadResult;
    }

    private String getFilePath(String sourceFileName) {
        DateTime dateTime = new DateTime();
        return "images/" + dateTime.toString("yyyy")
                + "/" + dateTime.toString("MM") + "/"
                + dateTime.toString("dd") + "/" + UUID.randomUUID().toString() +
"." +
                StringUtils.substringAfterLast(sourceFileName, ".");
    }
}
```

UpLoadController 控制器

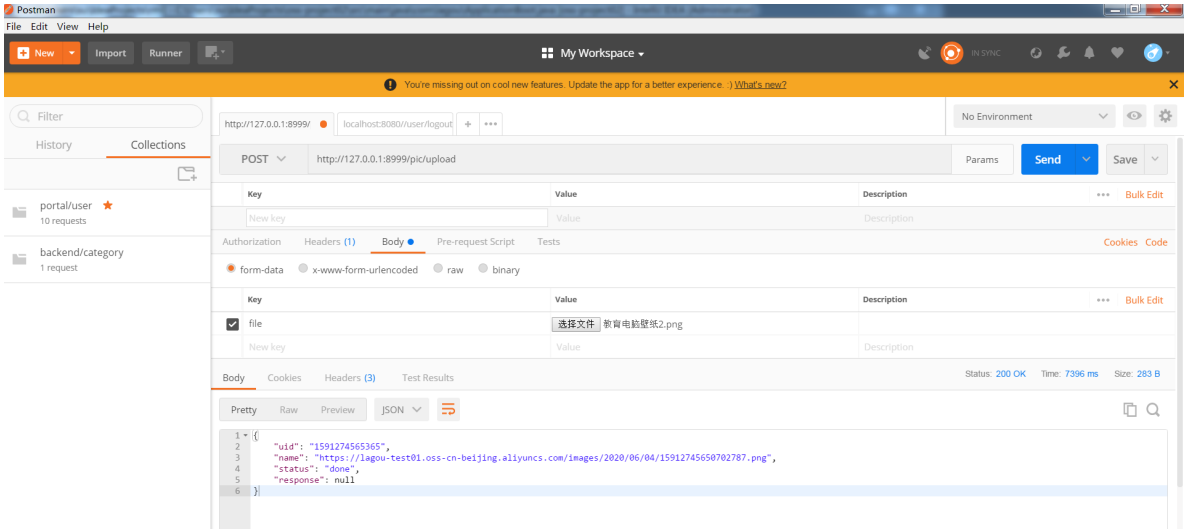
```
package com.lagou.controller;

import com.lagou.bean.UpLoadResult;
import com.lagou.service.FileUpLoadService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;

@RequestMapping("/pic")
@Controller
public class UpLoadController  {
```

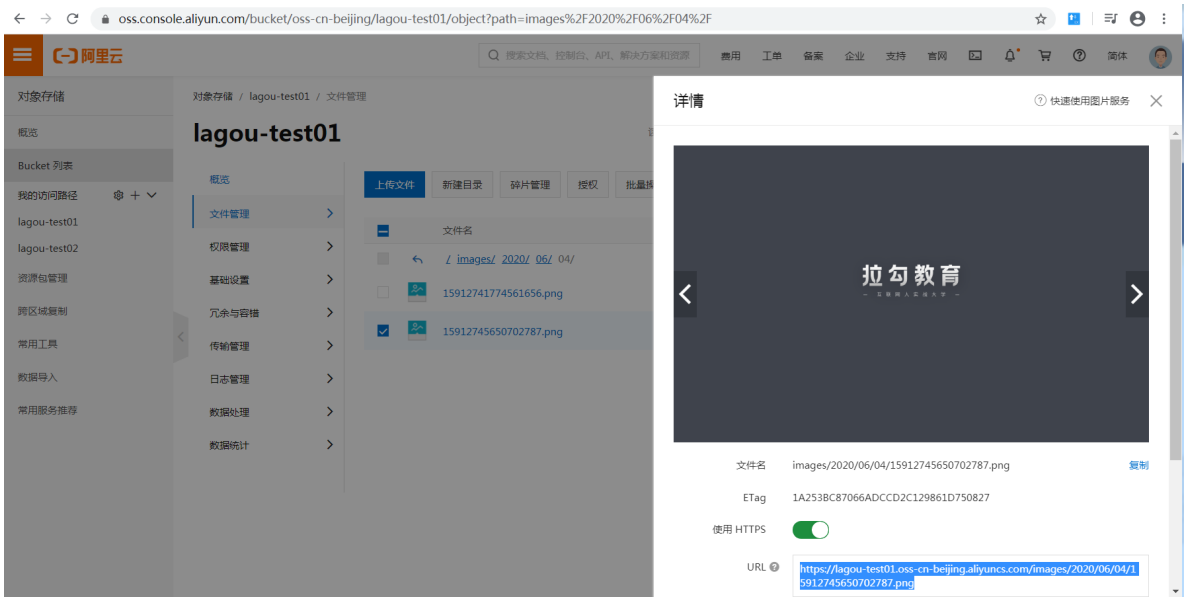
```
    @Autowired
    private FileUpLoadService fileUpLoadService;


    @PostMapping("/upload")
    @ResponseBody
    public UpLoadResult upload(@RequestParam("file") MultipartFile
multipartFile) {
        return this.fileUpLoadService.upload(multipartFile);
    }
}
```

ApplicationBoot 启动引导

```
package com.lagou;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ApplicationBoot {

    public static void main(String[] args) {
        SpringApplication.run(ApplicationBoot.class, args);
    }
}
```

4)测试





4.数据处理

4.1 介绍

阿里云OSS为在云上的数据打通了一个处理与使用的快速通道。通过简单的 RESTful 接口，可以在任何时间、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。

数据处理包含以下两种：

- 阿里云OSS原生处理服务

  阿里云OSS原生处理服务包括图片处理和视频截帧，其中图片处理包括图片的缩略、剪裁、参数调节等。 OSS原生处理服务无需开通，默认集成在OSS中，创建完Bucket后即可使用。产生的数据处理费用直接在OSS上结算。

- 智能媒体管理服务

  阿里云OSS与智能媒体管理（IMM）深度结合，支持文档预览、文档格式转换、人脸识别、图片分析、二维码识别等丰富的数据分析处理操作。

4.2 图片处理

本文档介绍如何快速使用OSS图片处理服务。您可以通过三种方式处理图片：为图片URL添加参数进行单次处理、使用图片样式对不同图片进行相同处理，或使用OSS SDK对图片进行处理。

处理操作	说明	文档
图片缩放	对图片进行等比缩放或固定宽高缩放。	图片缩放
图片裁剪	使用内切圆裁剪图片。	内切圆
通过指定范围的方式裁剪图片。	自定义裁剪	
将图片在x或y轴上等分为多个区域，然后取出指定区域。	索引切割	
使用圆角矩形裁剪图片。	圆角矩形	
图片旋转	设置是否对图片进行自动旋转	自适应方向
对图片进行指定角度的顺时针旋转。	旋转	
图片效果	为图片添加模糊效果。	模糊效果
调整图片的亮度。	亮度	
对图片进行锐化。	锐化	
调整图片的对比度。	对比度	
格式转换	将图片转换为指定的格式。	格式转换
可以对保存为JPG或WebP格式的图片进行图片质量转换。	质量变换	
指定图片的呈现方式。	渐进显示	
获取图片信息	获取图片的平均色调。	获取图片主色调
获取图片的宽度、长度、文件大小、格式、EXIF信息等。	获取信息	
图片水印	为图片添加图片、文字或混合水印。	图片水印

4.3 视频截祯

注意事项

- 当前仅支持对视频编码格式为H264的视频文件进行视频截帧。
- OSS当前没有默认保存视频截帧的操作，视频截帧的图片需手动下载到本地。

参数说明

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、png
ar	指定是否根据视频信息自动旋转图片。如果指定为auto，则会在截图生成之后根据视频旋转信息进行自动旋转。	枚举值：auto

https://help.aliyun.com/document_detail/64555.html

4.4 音视频处理

存储在OSS上的多媒体音视频数据，可以通过经济、弹性、高扩展的阿里云媒体转码服务，转换成适合在移动端、PC、TV上播放的格式。

媒体转码核心能力包括：

- 转换媒体格式，支持多平台播放。
- 保证相同画质质量的前提下，调整视频码率、提高视频压缩效率、减小文件体积，从而减少播放卡顿并节省存储空间和流量费用。
- 添加水印logo，突出品牌，增加产品识别度。
- 对视频进行剪辑/拼接等二次创作。
- 针对画质较差的视频，去除画面中的毛刺、马赛克等，修复为高清晰版本。

  ![]imges\音视频处理.jpg)

https://help.aliyun.com/document_detail/65583.html

4.5 智能媒体管理（IMM）

阿里云 OSS 能够与智能媒体管理（IMM）深度结合，支持文档预览、文档格式转换、人脸识别、图片分析、二维码识别等丰富的数据分析处理操作。下面介绍如何在 OSS 控制台中使用 IMM 的功能。

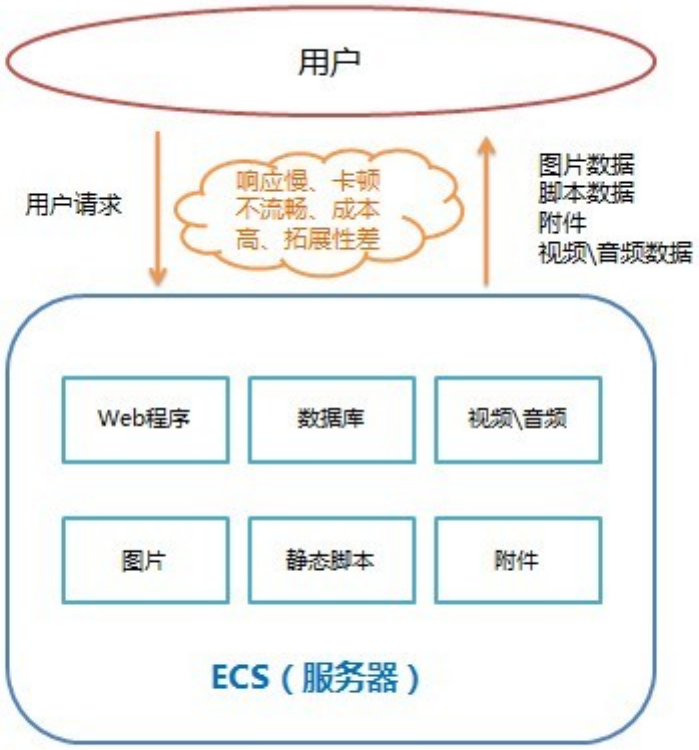
前提条件

- 要在 OSS 中使用 IMM 功能，您需要开通 IMM 服务并进行授权。有关开通服务及授权的详细步骤，请参见开通产品及创建项目中的前提条件部分。
- 如果您使用 RAM 子账号进行本文中的操作，需要同时开通对应存储空间的访问权限和 AliyunIMMFullAccess 权限。
- 创建 IMM Project 及使用 IMM 功能会产生一定的费用，如果您不需要使用 IMM 的功能，请及时解绑 IMM。详细费用请参见计费说明。

```
https://lagou-imgs.oss-cn-beijing.aliyuncs.com/1.png?x-oss-
process=imm/detecface
https://lagou-imgs.oss-cn-beijing.aliyuncs.com/1.png?x-oss-
process=imm/tagimage
```

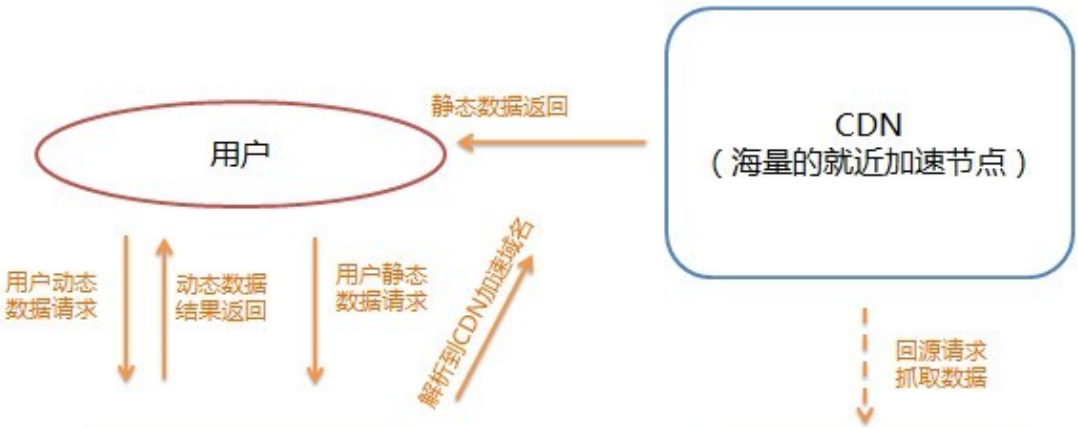
5.CDN 加速 (阿里 网宿 微软 亚马逊 akmai)

对象存储OSS与阿里云CDN服务结合，可优化静态热点文件下载加速的场景（即同一地区大量用户同时下载同一个静态文件的场景）。您可以将OSS的存储空间（Bucket）作为源站，利用阿里云CDN将源内容发布到边缘节点。当大量终端用户重复访问同一文件时，可以直接从边缘节点获取已缓存的数据，提高访问的响应速度。



传统网站架构示意

利用CDN和OSS实现动静分离，灵活的架构可以支持海量用户访问。产品架构如下图所示。



网站动静分离架构示意图

适用场景

- 静态文件访问量大，服务器负载高，I/O问题导致用户访问卡顿。
- 静态文件数量大，服务器存储空间不够。
- 静态文件用户访问分布在各地，同一个地区客户，会成千上万次重复下载同一文件。

https://www.aliyun.com/product/cdn