阿里云
对象存储 OSS

最佳实践

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

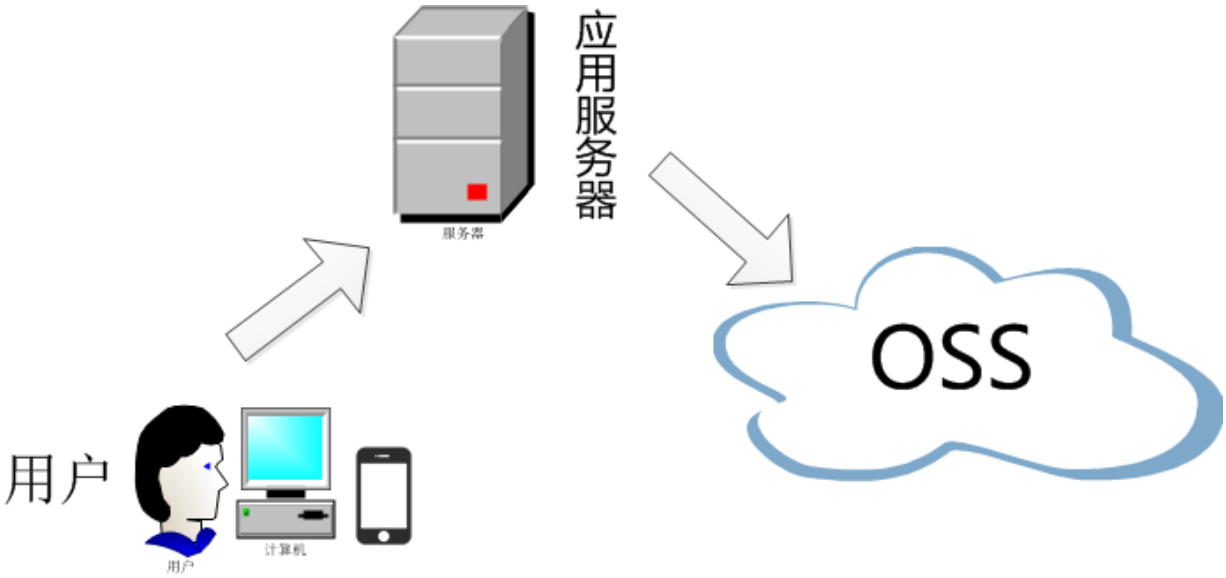
1 Web端直传实践

1.1 Web端直传实践简介

本教程介绍如何在Web端通过表单上传方式直接上传数据到OSS。

背景

每个OSS的用户都会用到上传服务。Web端常见的上传方法是用户在浏览器或app端上传文件到应用服务器，然后应用服务器再把文件上传到OSS。



和数据直传到OSS相比，以上方法有三个缺点：

- 上传慢。先上传到应用服务器，再上传到OSS，网络传送比直传到OSS多了一倍。如果直传到OSS，不通过应用服务器，速度将大大提升，而且OSS采用BGP带宽，能保证各地各运营商的速度。
- 扩展性差。如果后续用户多了，应用服务器会成为瓶颈。
- 费用高。需要准备多台应用服务器。由于OSS上传流量是免费的，如果数据直传到OSS，不通过应用服务器，那么将能省下几台应用服务器。

目的

本教程的目的是通过以下三个例子介绍如何通过表单直传数据到OSS：

- *JavaScript客户端签名直传*讲解在客户端通过JavaScript代码完成签名，然后通过表单直传数据到OSS。
- 服务端签名后直传讲解在服务端通过PHP代码完成签名，然后通过表单直传数据到OSS。

- 服务端签名直传并设置上传回调讲解在服务端通过PHP代码完成签名，并且服务端设置了上传后回调，然后通过表单直传数据到OSS。OSS回调完成后，应用服务器再返回结果给客户端。

1.2 JavaScript客户端签名直传

本示例讲解如何在客户端通过JavaScript代码完成签名，然后通过表单直传数据到OSS。

Demo

PC浏览器测试样例

本示例采用Plupload 直接提交表单数据（即*PostObject*）到OSS，可以运行在PC浏览器、手机浏览器、微信等。您可以同时选择多个文件上传，并设置上传到指定目录和设置上传文件名字是随机文件名还是本地文件名。您还可以通过进度条查看上传进度。

说明：

文件上传到一个测试的公共Bucket，会定时清理，所以不要传一些敏感及重要数据。

步骤 **1**：下载并安装**Plugload**

Plupload是一款简单易用且功能强大的文件上传工具， 支持多种上传方式，包括html5、flash、silverlight,、html4。它会智能检测当前环境，选择最适合的上传方式，并且会优先采用Html5方式。请参见*Plupload*官网进行下载和安装。

步骤 **2**：下载应用服务器代码

下载地址

步骤 **3**：修改配置文件

将下载包解压后，修改upload.js文件：

```
accessid= '<yourAccessKeyId>';
accesskey= <yourAccessKeySecrety>';
//如果是STS方式====
accessid = 'STS.ACCESSKEYID';
accesskey = 'STS.ACCESSSECRET';
token = 'STS.token';
//===========
host = 'http://post-test.oss-cn-hangzhou.aliyuncs.com';
```

- $id：您的AccessKeyId
- $key：您的AessKeySecret
- $host：格式为`BucketName.Endpoint`，例如`post-test.oss-cn-hangzhou.aliyuncs.com`

> **说明：**
> 关于Endpoint的介绍，请参见*Endpoint#访问域名#*。

步骤 4：设置**CORS**

HTML表单直接上传到OSS会产生跨域请求。为了浏览安全，需要为Bucket设置跨域规则（CORS），支持Post方法。

具体操作步骤请参见设置跨域访问。设置如下图所示：


跨域规则
* 来源
*
来源可以设置多个，每行一个，每行最多能有一个通配符「*」
* 允许 Methods
GET
POST
PUT
DELETE
HEAD
允许 Headers
*
允许 Headers 可以设置多个，每行一个，每行最多能有一个通配符「
暴露 Headers
etag
x-oss-request-id
暴露 Headers 可以设置多个，每行一个，不允许出现通配符「*」
缓存时间（秒）
0

说明：

在低版本IE浏览器，Plupload会以flash方式执行。您需要设置crossdomain.xml ，设置方法请参见*OSS Web*直传—使用*Flash*上传。

步骤 5：体验**JavaScript**客户端签名直传

1. 将应用服务器代码zip包解压到Web根目录下。
2. 在Web浏览器中输入<Web应用服务器地址>/oss-h5-upload-js-direct/index.html，例如http://abc.com:8080/oss-h5-upload-js-direct/index.html。
3. 选择一个或多个文件进行上传。
4. 上传成功后，通过控制台查看上传结果。

核心代码解析

因为OSS支持POST协议，所以只要在Plupload发送POST请求时带上OSS签名即可。核心代码如下：

```
var uploader = new plupload.Uploader({
    runtimes : 'html5,flash,silverlight,html4',
    browse_button : 'selectfiles',
    //runtimes : 'flash',
    container: document.getElementById('container'),
    flash_swf_url : 'lib/plupload-2.1.2/js/Moxie.swf',
    silverlight_xap_url : 'lib/plupload-2.1.2/js/Moxie.xap',
    url : host,
    multipart_params: {
        'Filename': '${filename}',
        'key' : '${filename}',
        'policy': policyBase64,
        'OSSAccessKeyId': accessid,
        'success_action_status' : '200', //让服务端返回200，不设置则默认返回204
        'signature': signature,
        'x-oss-security-token':token
    },
  ....
}
```

上述代码中，'Filename': '${filename}'表示上传后保持原来的文件名。如果您想上传到特定目录如abc下，且文件名不变，请修改代码如下：

```
multipart_params: {
        'Filename': 'abc/' + '${filename}',
        'key' : '${filename}',
        'policy': policyBase64,
        'OSSAccessKeyId': accessid,
        'success_action_status' : '200', //让服务端返回200，不设置则默认返回204
        'signature': signature,
```

```
    },
```

- 设置成随机文件名

  如果想在上传时固定设置成随机文件名，后缀保持跟客户端文件一致，可以将函数改为：

```
function check_object_radio() {
    g_object_name_type = 'random_name';
}
```

- 设置成用户的文件名

  如果想在上传时固定设置成用户的文件名，可以将函数改为：

```
function check_object_radio() {
    g_object_name_type = 'local_name';
}
```

- 设置上传目录

  您可以将文件上传到指定目录下。下面的代码是将上传目录改成abc/，注意目录必须以正斜线（/）结尾。

```
function get_dirname()
{
    g_dirname = "abc/";
}
```

- 上传签名

  上传签名主要是对policyText进行签名，最简单的例子如下：

```
var policyText = {
    "expiration": "2020-01-01T12:00:00.000Z", // 设置Policy的失效时
间，如果超过失效时间，就无法通过此Policy上传文件
    "conditions": [
    ["content-length-range", 0, 1048576000] // 设置上传文件的大小限
制，如果超过限制，文件上传到OSS会报错
    ]
}
```

总结

在客户端通过JavaScript代码完成签名，无需过多配置，即可实现直传，非常方便。但是客户端通过JavaScript把AccesssKeyID 和AccessKeySecret写在代码里面有泄露的风险，建议采用服务端签名后直传。

1.3 服务端签名后直传

本示例讲解如何在服务端通过PHP代码完成签名，然后通过表单直传数据到OSS。

> 说明：
> 本示例无法实现分片上传与断点续传。

背景

采用JavaScript客户端直接签名（参见*JavaScript客户端签名直传*）有一个严重的安全隐患：OSS AccessKey暴露在前端页面，这是非常不安全的做法。因此，OSS提供了服务端签名后直传的方案。

Demo

您可以通过样例体验服务端签名后直传效果：*PC浏览器测试样例*

原理介绍

服务端签名后直传的逻辑图如下：

流程如下：

1. 用户发送上传Policy请求到应用服务器。
2. 应用服务器返回上传Policy和签名给用户。
3. 用户使用Plupload直接上传数据到OSS。

步骤 1：下载并安装Plugload

Plupload是一款简单易用且功能强大的文件上传工具， 支持多种上传方式，包括html5、flash、silverlight,、html4。它会智能检测当前环境，选择最适合的上传方式，并且会优先采用Html5方式。请参见*Plupload*官网进行下载和安装。

步骤 2：下载应用服务器代码

- PHP：下载地址
- Java：下载地址
- Python：下载地址
- Go：下载地址

步骤 3：修改配置文件

本示例采用PHP编写。将下载包解压后，修改以下文件：

- php/get.php文件：

```
$id= '<yourAccessKeyId>';
 $key= '<yourAccessKeySecret>';
 $host = 'http://post-test.oss-cn-hangzhou.aliyuncs.com
```

  - $id：您的AccessKeyId
  - $key：您的AessKeySecret
  - $host：格式为`BucketName.Endpoint`，例如`post-test.oss-cn-hangzhou.aliyuncs.com`

    说明：

    关于Endpoint的介绍，请参见*Endpoint#访问域名#*。

- upload.js文件

  将变量severUrl改成服务器部署的地址，例如`http://abc.com:8080/oss-h5-upload-js-php/get.php`。

步骤 4：设置**CORS**

HTML表单直接上传到OSS会产生跨域请求。为了浏览安全，需要为Bucket设置跨域规则（CORS），支持Post方法。

具体操作步骤请参见设置跨域访问。设置如下图所示：


跨域规则
* 来源
*
来源可以设置多个，每行一个，每行最多能有一个通配符
* 允许 Methods
GET
POST
PUT
DELETE
允许 Headers
*
允许 Headers 可以设置多个，每行一个，每行最多能有一
暴露 Headers
暴露 Headers 可以设置多个，每行一个，不允许出现通配
缓存时间（秒）
600

说明：

在低版本IE浏览器，Plupload会以Flash方式执行。您需要设置crossdomain.xml ，设置方法请参见*OSS Web直传—使用Flash上传*。

步骤 5：体验服务端签名后直传

1. 将应用服务器代码zip包解压到Web根目录下。
2. 在Web浏览器中输入`<Web应用服务器地址>/oss-h5-upload-js-php/index.html`，例如`http://abc.com:8080/oss-h5-upload-js-php/index.html`。
3. 选择一个或多个文件进行上传。
4. 上传成功后，通过控制台查看上传结果。

核心代码解析

- 设置成随机文件名

  如果想在上传时固定设置成随机文件名，后缀保持跟客户端文件一致，可以将函数改为：

```
function check_object_radio() {
    g_object_name_type = 'random_name';
}
```

- 设置成用户的文件名

  如果想在上传时固定设置成用户的文件名，可以将函数改为：

```
function check_object_radio() {
    g_object_name_type = 'local_name';
}
```

- 设置上传目录

  上传的目录由服务端（即PHP）指定， 每个客户端只能上传到指定的目录，实现安全隔离。下面的代码是将上传目录改成abc/，注意目录必须以正斜线（/）结尾。

```
$dir = 'abc/';
```

- 设置上传过滤条件

  您可以利用Plupload的属性filters设置上传的过滤条件，如设置只能上传图片、上传文件的大小、不能有重复上传等。

```
var uploader = new plupload.Uploader({
    ......
    filters: {
        mime_types : [ //只允许上传图片和zip文件
```

```
        { title : "Image files", extensions : "jpg,gif,png,bmp" },
        { title : "Zip files", extensions : "zip" }
        ],
        max_file_size : '400kb', //最大只能上传400KB的文件
        prevent_duplicates : true //不允许选取重复文件
    },
```

- mime_types：限制上传的文件后缀
- max_file_size：限制上传的文件大小
- prevent_duplicates：限制不能重复上传

> 说明：
> filters过滤条件不是必须的。如果不想设置过滤条件，只要把该项注释即可。

- 获取上传后的文件名

  如果要知道文件上传成功后的文件名，可以用Plupload调用FileUploaded事件获取，如下所示：

```
FileUploaded: function(up, file, info) {
            if (info.status == 200)
            {
                document.getElementById(file.id).getElement
sByTagName('b')[0].innerHTML = 'upload to oss success, object name:'
 + get_uploaded_object_name(file.name);
            }
            else
            {
                document.getElementById(file.id).getElement
sByTagName('b')[0].innerHTML = info.response;
            }
    }
```

  可以利用如下函数，得到上传到OSS的文件名，其中file.name记录了上传本地文件的名称。

```
get_uploaded_object_name(file.name)
```

- 上传签名

  JavaScript可以从服务端获取policyBase64、accessid、signature这三个变量，获取这三个变量的核心代码如下：

```
phpUrl = './php/get.php'
        xmlhttp.open( "GET", phpUrl, false );
        xmlhttp.send( null );
        var obj = eval ("(" + xmlhttp.responseText+ ")");
        host = obj['host']
        policyBase64 = obj['policy']
        accessid = obj['accessid']
        signature = obj['signature']
        expire = parseInt(obj['expire'])
```

```
        key = obj['dir']
```

xmlhttp.responseText解析如下：

说明：

以下仅为示例，并不要求必须是相同的格式，但是必须有accessid、policy、signature这三个值。

```
{"accessid":"6MKOqxGiGU4AUk44",
"host":"http://post-test.oss-cn-hangzhou.aliyuncs.com",
"policy":"eyJleHBpcmF0aW9uIjoiMjAxNS0xMS0wNVQyMDoyMzoyM1oiLC
Jjxb25kaXRpb25zIjpbWyJjcb250ZW50LWxlbmd0aC1yYW5nZSIsMCwxMDQ4
NTc2MDAwXSxbInN0YXJ0cy13aXRoIiwiJGtleSIsInVzZXItZGlyXC8iXV19",
"signature":"I2u57FWjTKqX/AE6doIdyff151E=",
"expire":1446726203,"dir":"user-dir/"}
```

- accessid：用户请求的accessid。
- host：用户要往哪个域名发送上传请求。
- policy：用户表单上传的策略（Policy），是经过base64编码过的字符串。
- signature：对变量policy签名后的字符串。
- expire：上传策略失效时间，在PolicyText里指定。在失效时间之前，都可以利用此Policy上传文件，所以没有必要每次上传都去服务端获取签名。

说明：

为了减少服务端的压力，设计思路是：初始化上传时，每上传一个文件后，获取一次签名。然后再上传时，比较当前时间与签名时间，看签名时间是否失效。如果失效了，就重新获取一次签名，如果没有失效，就使用之前的签名。这里就用到了变量expire，核心代码如下：

```
now = timestamp = Date.parse(new Date()) / 1000;
[color=#000000]//可以判断当前expire是否超过了当前时间，如果超过了当前时
间，就重新取一次签名，缓冲时间为3[/color]
    if (expire < now + 3)
{
      .....
      phpUrl = './php/get.php'
      xmlhttp.open( "GET", phpUrl, false );
      xmlhttp.send( null );
      ......
}
return .
```

解析Policy的内容如下：

```
{"expiration":"2015-11-05T20:23:23Z",
"conditions":[["content-length-range",0,1048576000],
```

```
["starts-with","$key","user-dir/"]]
```

说明：

Policy的详细信息请参见*Policy*语法结构。

上面Policy中增加了starts-with，用来指定此次上传的文件名必须以user-dir开头，用户可自行指定此字符串。增加starts-with的原因是：在很多场景下，一个应用对应一个Bucket，为了防止数字覆盖，每个用户上传到OSS的文件都可以有特定的前缀。这样就存在一个问题，用户获取到这个Policy后，在失效期内都能修改上传前缀，从而上传到别人的目录下。为了解决这个问题，可以设置应用服务器在上传时就指定用户上传的文件必须是某个前缀。这样如果用户获取到了Policy也没有办法上传到别人的前缀上，从而保证了数据的安全性。

总结

本示例中，web端向服务端请求签名，然后直接上传，不会对服务端产生压力，而且安全可靠。但是这个示例有个问题，就是用户上传了多少文件，上传了什么文件，服务端并不能马上知道，如果想实时了解用户上传了什么文件，可以采用服务端签名直传并设置上传回调。

1.4 服务端签名直传并设置上传回调

1.4.1 原理介绍

本示例讲解在服务端通过各种语言代码完成签名，并且设置上传回调，然后通过表单直传数据到OSS。

背景

采用服务端签名后直传方案有个问题：用户上传数据后，很多场景下，应用服务器需要知道用户上传了哪些文件以及文件名称，如果是图片的话，还需要知道图片的大小等。为此OSS提供了上传回调方案。OSS回调完成后，应用服务器再返回结果给客户端。这样服务端就可以实时了解用户上传了什么文件。

Demo

您可以通过样例体验服务端签名直传并设置上传回调的效果：*PC*浏览器测试样例

流程介绍

流程如下：

1. 用户向应用服务器请求上传Policy和回调。

2. 应用服务器返回上传Policy和回调设置。
3. 用户直接向OSS发送文件上传请求。
4. OSS根据用户的回调设置，发送回调请求给应用服务器。
5. 应用服务器返回响应给OSS。
6. OSS将应用服务器返回的内容返回给用户。

简单讲，就是用户要上传一个文件到OSS，而且希望将上传的结果返回给应用服务器，这时就需要设置一个回调函数，将请求告知应用服务器。用户上传完文件后，不会直接得到返回结果，而是先通知应用服务器，再把结果转达给用户。

流程解析

服务端签名直传并设置上传回调提供了PHP、Java、Python、Go、Ruby语言版本的具体示例。以下根据流程讲解核心代码和消息内容。

1. 用户向应用服务器请求上传Policy和回调。

   在客户端源码中的`upload.js`文件中，如下代码片段的变量`serverUrl`的值可以用来设置签名服务器的URL。设置好之后，客户端会向该`serverUrl`发送GET请求来获取需要的信息。

   ```
   // serverUrl是 用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
   serverUrl = 'http://88.88.88.88:8888'
   ```

2. 应用服务器返回上传Policy和回调设置代码。

   应用服务器侧的签名直传服务会处理客户端发过来的GET请求消息，您可以设置对应的代码让应用服务器能够给客户端返回正确的消息。各个语言版本的配置文档中都有明确的说明供您参考。

   下面是签名直传服务返回给客户端消息body内容的示例，这个body的内容将作为客户端上传文件的重要参数。

   ```
   {
   "accessid":"6MKOqxGiGU4AUk44",
   "host":"http://post-test.oss-cn-hangzhou.aliyuncs.com",
   "policy":"eyJleHBpcmF0aW9uIjoiMjAxNS0xMS0wNVQyMDo1MjoyOVoiLC
   Jjdb25kaXRpb25zIjpbWyJjdb250ZW50LWxlbmd0aC1yYW5nZSIsMCwxMDQ4
   NTc2MDAwXSxbInN0YXJ0cy13aXRoIiwiJGtleSIsInVzZXItZGlyXC8iXV19",
   "signature":"VsxOcOudxDbtNSvz93CLaXPz+4s=",
   "expire":1446727949,
   "callback":"eyJjYWxsYmFja1VybCI6Imh0dHA6Ly9vc3MtZGVtby5hbGl5dW
   5jcy5jdb206MjM0NTAiLCJjYWxsYmFja0hvc3QiOiJvc3MtZGVtby5hbGl5d
   W5jcy5jdb20iLCJjYWxsYmFja0JvZHkiOiJmaWxlbmFtZT0ke29iamVjdH0m
   c2l6ZT0ke3NpemV9Jm1pbWVUeXBlPSR7bWltZVR5cGV9JmhlaWdodD0ke2lt
   YWdlSW5mby5oZWlnaHR9JndpZHRoPSR7aW1hZ2VJdbmZvLndpZHRofSIsImN
   hbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVybGV
   uY29kZWQifQ==",
   ```

```
"dir":"user-dirs/"
}
```

上述示例的callback内容采用的是base64编码。经过base64解码后的内容如下：

```
{"callbackUrl":"http://oss-demo.aliyuncs.com:23450",
"callbackHost":"oss-demo.aliyuncs.com",
"callbackBody":"filename=${object}&size=${size}&mimeType=${mimeType
}&height=${imageInfo.height}&width=${imageInfo.width}",
"callbackBodyType":"application/x-www-form-urlencoded"}
```

内容解析如下：

- CallbackUrl：OSS往这个服务器发送的URL请求。
- callbackHost：OSS发送这个请求时，请求头部所带的Host头。
- callbackBody：OSS请求时，发送给应用服务器的内容，可以包括文件的名称、大小、类型。如果是图片，可以是图片的高度、宽度。
- callbackBodyType：请求发送的Content-Type。

3. 用户直接向OSS发送文件上传请求。

   在客户端源码`upload.js`文件中，`callbackbody`的值是步骤2中应用服务器返回给客户端消息body中callback的内容。

```
new_multipart_params = {
     'key' : key + '${filename}',
     'policy': policyBase64,
     'OSSAccessKeyId': accessid,
     'success_action_status' : '200', //让服务端返回200，不设置则默认返回
204
     'callback':   callbackbody,
     'signature': signature,
 };
```

4. OSS根据用户的回调设置，发送回调请求给应用服务器。

   客户端上传文件到OSS结束后，OSS解析客户端的上传回调设置，发送POST回调请求给应用服务器。消息内容大致如下：

```
Hypertext Transfer Protocol
    POST / HTTP/1.1\r\n
    Host: 47.97.168.53\r\n
    Connection: close\r\n
    Content-Length: 76\r\n
    Authorization: fsNxFF0wNOpC0YMNAoFb//a8x6v2lI1ih7kXIh3nFU
DALgku9bhC+cWQsnxuCo88vahKtBUmnDI6k1PofggA4g==\r\n
    Content-MD5: eiEMyp7lbL8KStPBzMdr9w==\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    Date: Sat, 15 Sep 2018 10:24:12 GMT\r\n
    User-Agent: aliyun-oss-callback\r\n
    x-oss-additional-headers: \r\n
    x-oss-bucket: signedcallback\r\n
```

```
    x-oss-owner: 1544709436620439\r\n
    x-oss-pub-key-url: aHR0cHM6Ly9nb3NzcHVibGljLmFsaWNkbi5jb20v
Y2FsbGJhY2tfcHViX2tleV92MS5wZW0=\r\n
    x-oss-request-id: 5B9CDDCC9D2B0CA88AE2BEA1\r\n
    x-oss-requester: 1544709436620439\r\n
    x-oss-signature-version: 1.0\r\n
    x-oss-tag: CALLBACK\r\n
    eagleeye-rpcid: 0.1\r\n
    \r\n
    [Full request URI: http://47.97.168.53/]
    [HTTP request 1/1]
    [Response in frame: 39]
    File Data: 76 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
    Form item: "filename" = ".snappython.png"
    Form item: "size" = "6014"
    Form item: "mimeType" = "image/png"
    Form item: "height" = "221"
```

5. 应用服务器返回响应给OSS。

   应用服务器根据OSS发送消息中的`authorization`来进行验证，如果验证通过，则向OSS返回如下json格式的成功消息。

```
String value: OK
Key: Status
```

6. OSS将应用服务器返回的消息返回给用户。

客户端源码解析

客户端JavaScript代码使用的是Plupload组件。

> 说明：
> Plupload是一款简单易用且功能强大的文件上传工具， 支持多种上传方式，包括html5、flash、silverlight,、html4。它会智能检测当前环境，选择最适合的上传方式，并且会优先采用Html5方式。详情请参见*Plupload*官网。

下面举例介绍几个关键功能的代码实现：

- 设置成随机文件名

  若上传时采用固定格式的随机文件名，且后缀跟客户端文件名保持一致，可以将函数改为：

```
function check_object_radio() {
    g_object_name_type = 'random_name';
}
```

- 设置成用户的文件名

如果想在上传时设置成用户的文件名，可以将函数改为：

```
function check_object_radio() {
    g_object_name_type = 'local_name';
}
```

- 设置上传目录

  上传的目录由服务端指定， 每个客户端只能上传到指定的目录，实现安全隔离。下面的代码会把上传目录改成abc/，注意目录必须以正斜线 (/) 结尾。

  ```
  $dir ='abc/';
  ```

- 设置上传过滤条件

  您可以利用Plupload的属性filters设置上传的过滤条件，如设置只能上传图片、上传文件的大小、不能有重复上传等。

  ```
  var uploader = new plupload.Uploader({
      ......
      filters: {
          mime_types : [ //只允许上传图片和zip文件
          { title : "Image files", extensions : "jpg,gif,png,bmp" },
          { title : "Zip files", extensions : "zip" }
          ],
          max_file_size : '400kb', //最大只能上传400KB的文件
          prevent_duplicates : true //不允许选取重复文件
      },
  ```

  - mime_types：限制上传的文件后缀。
  - max_file_size：限制上传的文件大小。
  - prevent_duplicates：限制不能重复上传。

- 获取上传后的文件名

  如果要知道文件上传成功后的文件名，可以用Plupload调用FileUploaded事件获取，如下所示：

  ```
  FileUploaded: function(up, file, info) {
              if (info.status == 200)
              {
                  document.getElementById(file.id).getElement
  sByTagName('b')[0].innerHTML = 'upload to oss success, object name:'
   + get_uploaded_object_name(file.name);
              }
              else
              {
                  document.getElementById(file.id).getElement
  sByTagName('b')[0].innerHTML = info.response;
              }
      }
  ```

  可以利用如下函数，得到上传到OSS的文件名，其中file.name记录了本地文件上传的名称。

`get_uploaded_object_name(file.name)`

- 上传签名

JavaScript可以从服务端获取policyBase64、accessid、signature这三个变量，核心代码如下：

```
function get_signature()
{
    // 判断expire的值是否超过了当前时间，如果超过了当前时间，就重新获取签名，缓冲时间为3秒。
    now = timestamp = Date.parse(new Date()) / 1000;
    if (expire < now + 3)
    {
        body = send_request()
        var obj = eval ("(" + body + ")");
        host = obj['host']
        policyBase64 = obj['policy']
        accessid = obj['accessid']
        signature = obj['signature']
        expire = parseInt(obj['expire'])
        callbackbody = obj['callback']
        key = obj['dir']
        return true;
    }
    return false;
};
```

从服务端返回的消息解析如下：

说明：

以下仅为示例，并不要求相同的格式，但必须有accessid、policy、signature三个值。

```
{"accessid":"6MKOqxGiGU4AUk44",
"host":"http://post-test.oss-cn-hangzhou.aliyuncs.com",
"policy":"eyJleHBpcmF0aW9uIjoiMjAxNS0xMS0wNVQyMDoyMzoyM1oiLC
Jjxb25kaXRpb25zIjpbWyJjcb250ZW50LWxlbmd0aC1yYW5nZSIsMCwxMDQ4
NTc2MDAwXSxbInN0YXJ0cy13aXRoIiwiJGtleSIsInVzZXItZGlyXC8iXV19",
"signature":"I2u57FWjTKqX/AE6doIdyff151E=",
"expire":1446726203,"dir":"user-dir/"}
```

- accessid：用户请求的accessid。
- host：用户要往哪个域名发送上传请求。
- policy：用户表单上传的策略（Policy），是经过base64编码过的字符串。
- signature：对变量policy签名后的字符串。
- expire：上传策略Policy失效时间，在服务端指定。失效时间之前都可以利用此Policy上传文件，无需每次上传都去服务端获取签名。

说明：

为了减少服务端的压力，设计思路是：初始化上传时，每上传一个文件，获取一次签名。再上传时，比较当前时间与签名时间，看签名时间是否失效。如果失效，就重新获取一次签名，如果没有失效，就使用之前的签名。

解析Policy的内容如下：

```
{"expiration":"2015-11-05T20:23:23Z",
"conditions":[["content-length-range",0,1048576000],
["starts-with","$key","user-dir/"]]
```

上面Policy中增加了starts-with，用来指定此次上传的文件名必须以user-dir开头，用户也可自行指定。增加starts-with的原因是：在很多场景下，一个应用对应一个Bucket，为了防止数据覆盖，每个用户上传到OSS的文件都可以有特定的前缀。但这样存在一个问题，用户获取到这个Policy后，在失效期内都能修改上传前缀，从而上传到别人的目录下。解决方法为，在应用服务器端就指定用户上传文件的前缀。如果用户获取了Policy也没有办法上传到别人的目录，从而保证了数据的安全性。

- 设置应用服务器的地址

  在客户端源码`upload.js`文件中，如下代码片段的变量`serverUrl`的值可以用来设置签名服务器的URL，设置好之后，客户端会向该`serverUrl`发送GET请求来获取需要的信息。

```
// serverUrl是 用户获取签名和Policy等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
serverUrl = 'http://88.88.88.88:8888'
```

1.4.2 Go

本文以Golang语言为例，讲解在服务端通过Golang代码完成签名，并且设置上传回调，然后通过表单直传数据到OSS。

前提条件

- 应用服务器对应的域名可通过公网访问。
- 应用服务器已经安装Golang 1.6以上版本（执行命令`go version`进行验证）。
- PC端浏览器支持JavaScript。

步骤一：配置应用服务器

1. 下载应用服务器源码（Go版本）到应用服务器的目录下。下载地址： *aliyun-oss-appserver-go-master.zip*
2. 以Ubuntu 16.04为例，将源码放置到`/home/aliyun/aliyun-oss-appserver-go`目录下。

3. 进入该目录，打开源码文件`appserver.go`，修改以下代码片段：

```
// 请填写您的AccessKeyId。
var accessKeyId string = "<yourAccessKeyId>"

// 请填写您的AccessKeySecret。
var accessKeySecret string = "<yourAccessKeySecret>"

// host的格式为 bucketname.endpoint，请替换为您的真实信息。
var host string = "http://bucket-name.oss-cn-hangzhou.aliyuncs.com'"

// callbackUrl 为上传回调服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
var callbackUrl string = "http://88.88.88.88:8888";

// 上传文件时指定的前缀。
var upload_dir string = "user-dir-prefix/"

// 上传策略Policy的失效时间，单位为秒。
var expire_time int64 = 30
```

- accessKeyId：设置您的AccessKeyId。
- accessKeySecret：设置您的AessKeySecret。
- host：格式为`bucketname.endpoint`，例如`bucket-name.oss-cn-hangzhou.aliyuncs.com`。关于Endpoint的介绍，请参见*Endpoint访问域名*。
- callbackUrl：设置上传回调URL，即回调服务器地址，用于处理应用服务器与OSS之前的通信。OSS会在文件上传完成后，把文件上传信息通过此回调URL发送给应用服务器。本例中修改为：

  `var callbackUrl string="http://11.22.33.44:1234";`
- upload_dir：指定上传文件的前缀，以免与其他文件发生冲突，您也可以填写空值。

步骤二：配置客户端

1. 下载客户端源码到PC侧的本地目录。下载地址：*aliyun-oss-appserver-js-master.zip*
2. 以`D:\aliyun\aliyun-oss-appserver-js`目录为例。进入该目录，打开`upload.js`文件，找到下面的代码语句：

   ```
   // serverUrl是用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
   serverUrl ='http://88.88.88.88:8888'
   ```
3. 将变量`severUrl`修改为应用服务器的地址。如本例中修改为：

   ```
   // serverUrl是用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
   ```

```
serverUrl ='http://11.22.33.44:1234'
```

步骤三： 修改**CORS**

客户端进行表单直传到OSS时，会从浏览器向OSS发送带有Origin的请求消息。OSS对带有Origin头的请求消息会进行跨域规则（CORS）的验证。因此需要为Bucket设置跨域规则以支持Post方法。

具体操作步骤请参见设置跨域访问。

步骤四：体验上传回调

1. 启动应用服务器。

   在/home/aliyun/aliyun-oss-appserver-go目录下，执行Go命令：

   go run appserver.go 11.22.33.44 1234

2. 启动客户端。

   a. 在PC侧的客户端源码目录中，打开index.html 文件。

   b. 单击选择文件，选择指定类型的文件之后，单击开始上传。上传成功后，显示回调服务器返回的内容。

应用服务器核心代码解析

应用服务器源码包含了签名直传服务和上传回调服务两个功能。

- 签名直传服务

  签名直传服务响应客户端发送给应用服务器的GET消息，代码片段如下：

```
func handlerRequest(w http.ResponseWriter, r *http.Request) {
        if (r.Method == "GET") {
                response := get_policy_token()
                w.Header().Set("Access-Control-Allow-Methods", "POST
")
                w.Header().Set("Access-Control-Allow-Origin", "*")
                io.WriteString(w, response)
  }
```

- 上传回调服务

  上传回调服务响应OSS发送给应用服务器的POST消息，代码片段如下：

```
if (r.Method == "POST") {
                fmt.Println("\nHandle Post Request ... ")

                // Get PublicKey bytes
                bytePublicKey, err := getPublicKey(r)
                if (err != nil) {
                        responseFailed(w)
```

```
                        return
                }

                // Get Authorization bytes : decode from Base64Stri
ng
                byteAuthorization, err := getAuthorization(r)
                if (err != nil) {
                        responseFailed(w)
                        return
                }

                // Get MD5 bytes from Newly Constructed Authrization
 String.
                byteMD5, err := getMD5FromNewAuthString(r)
                if (err != nil) {
                        responseFailed(w)
                        return
                }

                // verifySignature and response to client
                if (verifySignature(bytePublicKey, byteMD5,
byteAuthorization)) {
                        // do something you want accoding to
callback_body ...

                        responseSuccess(w)  // response OK : 200
                } else {
                        responseFailed(w)   // response FAILED : 400

                }
        }
```

1.4.3 PHP

本文以PHP语言为例，讲解在服务端通过PHP代码完成签名，并且设置上传回调，然后通过表单直传数据到OSS。

前提条件

- Web服务器已部署。
- Web服务器对应的域名可通过公网访问。
- Web服务器能够解析PHP（执行命令`php -v`进行查看）。
- PC端浏览器支持JavaScript。

步骤一：配置Web服务器

本文档以`Ubuntu16.04`和`Apache2.4.18`为例。本示例中的环境配置如下：

- Web服务器外网IP地址为`11.22.33.44`。您可以在配置文件`/etc/apache2/apache2.conf`中增加`ServerName 11.22.33.44`来进行修改。
- Web服务器的监听端口为`8080`。您可以在配置文件`/etc/apache2/ports.conf`中进行修改相关内容`Listen 8080`。

- 确保Apache能够解析PHP文件：`sudo apt-get install libapache2-mod-php5`（其他平台请根据实际情况进行安装配置）。

您可以根据自己的实际环境修改IP地址和监听端口。更新配置后，需要重启Apache服务器，重启命令为`/etc/init.d/apache2 restart`。

步骤二：配置应用服务器

- 下载部署应用服务器源码

  下载应用服务器源码(PHP版本)：*aliyun-oss-appserver-php-master.zip*

  下载应用服务器源码后，将其部署到应用服务器的相应目录。本文示例中需要部署到`Ubuntu16.04`的`/var/www/html/aliyun-oss-appserver-php`目录下。

  在PC侧浏览器中访问应用服务器URL `http://11.22.33.44:8080/aliyun-oss-appserver-php/index.html`，显示的页面内容与测试样例主页相同则验证通过。

- 开启Apache捕获HTTP头部Authorization字段的功能。

  您的应用服务器收到的回调请求有可能没有Authotization头，这是因为有些Web应用服务器会将Authorization头自行解析掉。比如Apache2，需要设置成不解析头部。

  - 打开Apache2配置文件`/etc/apache2/apache2.conf`，找到如下片段，进行相应修改：

    ```
    <Directory /var/www/>
            Options Indexes FollowSymLinks
            AllowOverride All
            Require all granted
    </Directory>
    ```

  - 在`/var/www/html/aliyun-oss-appserver-php`目录下，新建一个`.htaccess`文件，填写如下内容：

    ```
    <IfModule mod_rewrite.c>
    RewriteEngine on
    RewriteCond %{HTTP:Authorization} .
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
    </IfModule>
    ```

  其他Web服务器或其：他Apache版本，请根据实际情况进行配置。

- 修改配置应用服务器

  在`/var/www/html/aliyun-oss-appserver-php/php`目录下打开文件`get.php`，修改如下代码片段：

  ```
  $ id= '<yourAccessKeyId>';          // 请填写您的AccessKeyId。
  ```

```
    $ key= '<yourAccessKeySecret>';      // 请填写您的AccessKeySecret。

    // $host的格式为 bucketname.endpointx， 请替换为您的真实信息。
    $host = 'http://bucket-name.oss-cn-hangzhou.aliyuncs.com';

    // $callbackUrl为上传回调服务器的URL， 请将下面的IP和Port配置为您自己的
真实URL信息。
    $callbackUrl = 'http://88.88.88.88:8888/aliyun-oss-appserver-php
/php/callback.php';

    $dir = 'user-dir-prefix/';           // 上传文件时指定的前缀。
```

- $id ： 设置您的AccessKeyId。
- $key ： 设置您的AessKeySecret。
- $host： 格式为`BucketName.Endpoint`，例如`bucket-name.oss-cn-hangzhou.aliyuncs.com`。关于Endpoint的介绍，请参见*Endpoint访问域名*。
- $callbackUrl：设置上传回调URL，即回调服务器地址，用于处理应用服务器与OSS之前的通信。OSS会在文件上传完成后，把文件上传信息通过此回调URL发送给应用服务器。本例中修改为：

  `$callbackUrl='http://11.22.33.44:8080/aliyun-oss-appserver-php/php/callback.php';`
- $dir： 设置上传到OSS文件的前缀，以便区别于其他文件从而避免冲突，您也可以填写空值。

步骤三：配置客户端

在应用服务器的`/var/www/html/aliyun-oss-appserver-php`目录下修改文件`upload.js`：

对于PHP版本的应用服务器源码，一般不需要修改文件`upload.js`内容的，因为相对路径也是可以正常工作的。

如果确实需要修改，请找到如下的代码片段：

`serverUrl ='./php/get.php'`

将变量`severUrl`改成服务器部署的地址，用于处理浏览器和应用服务器之间的通信。

比如本示例中可以如下修改：

`serverUrl ='http://11.22.33.44:8080/aliyun-oss-appserver-php/php/get.php'`

步骤四：修改CORS

从浏览器向OSS发出的请求消息带有`Origin`的消息头，OSS对带有`Origin`头的请求消息会首先进行跨域规则的验证。

即客户端进行表单直接上传到OSS会产生跨域请求，需要为Bucket设置跨域规则（CORS），支持Post方法。

具体操作步骤请参见设置跨域访问。

步骤五：体验上传回调

在PC侧的Web浏览器中输入`http://11.22.33.44:8080/aliyun-oss-appserver-php/index.html`。

单击**选择文件**，选择指定类型的文件后，单击**开始上传**。上传成功后，显示回调服务器返回的内容。

应用服务器核心代码解析

应用服务器源码包含了签名直传服务和上传回调服务两个功能。

- 签名直传服务

  签名直传服务响应客户端发送给应用服务器的GET消息，代码文件是`aliyun-oss-appserver-php/php/get.php`。代码片段如下：

```
$response = array();
$response['accessid'] = $id;
$response['host'] = $host;
$response['policy'] = $base64_policy;
$response['signature'] = $signature;
$response['expire'] = $end;
$response['callback'] = $base64_callback_body;
$response['dir'] = $dir;
```

- 上传回调服务

  `上传回调服务`响应OSS发送给应用服务器的POST消息，代码文件是`aliyun-oss-appserver-php/php/callback.php`。

  代码片段如下：

```
// 6.验证签名
$ok = openssl_verify($authStr, $authorization, $pubKey, OPENSSL_ALGO_MD5);
if ($ok == 1)
{
    header("Content-Type: application/json");
    $data = array("Status"=>"Ok");
    echo json_encode($data);
```

```
}
```

1.4.4 Java

本文以Java语言为例，讲解在服务端通过Java代码完成签名，并且设置上传回调，然后通过表单直传数据到OSS。

前提条件

- 应用服务器对应的域名可通过公网访问。
- 确保应用服务器已经安装Java 1.6以上版本（执行命令java -version进行查看）。
- 确保PC端浏览器支持JavaScript。

步骤一：配置应用服务器

下载应用服务器源码(Java版本)：*aliyun-oss-appserver-java-master.zip*

将源码下载到应用服务器的硬盘，本示例中以Ubuntu 16.04为例，放置到/home/aliyun/aliyun-oss-appserver-java目录下。进入该目录，找到并打开源码文件CallbackServer.java，修改如下的代码片段：

```
String accessId = "<yourAccessKeyId>";      // 请填写您的AccessKeyId。
String accessKey = "<yourAccessKeySecret>"; // 请填写您的AccessKeySecret
。
String endpoint = "oss-cn-hangzhou.aliyuncs.com"; // 请填写您的 endpoint
。
String bucket = "bucket-name";                    // 请填写您的
bucketname 。
String host = "http://" + bucket + "." + endpoint; // host的格式为
bucketname.endpoint

// callbackUrl为 上传回调服务器的URL，请将下面的IP和Port配置为您自己的真实信
息。
String callbackUrl = "http://88.88.88.88:8888";
String dir = "user-dir-prefix/"; // 用户上传文件时指定的前缀。
```

- accessId ： 设置您的AccessKeyId。
- accessKey ： 设置您的AessKeySecret。
- host： 格式为bucketname.endpoint，例如bucket-name.oss-cn-hangzhou.aliyuncs.com。关于Endpoint的介绍，请参见*Endpoint访问域名*。
- callbackUrl： 设置上传回调URL，即回调服务器地址，用于处理应用服务器与OSS之前的通信。OSS会在文件上传完成后，把文件上传信息通过此回调URL发送给应用服务器。本例中修改为：

  ```
  String callbackUrl ="http://11.22.33.44:1234";
  ```

- dir： 设置上传到OSS文件的前缀，以便区别于其他文件从而避免冲突，您也可以填写空值。

步骤二：配置客户端

下载客户端源码：*aliyun-oss-appserver-js-master.zip*

下载客户端源码到PC侧的本地硬盘。本例中以`D:\aliyun\aliyun-oss-appserver-js`目录为例。

进入该目录，打开`upload.js`文件，找到下面的代码语句：

```
// serverUrl是 用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
serverUrl = 'http://88.88.88.88:8888'
```

将变量`severUrl`改成应用服务器的地址，客户端可以通过它可以获取签名直传Policy等信息。如本例中可修改为：

```
// serverUrl是 用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
serverUrl = 'http://11.22.33.44:1234'
```

步骤三： 修改**CORS**

从浏览器向OSS发出的请求消息带有`Origin`的消息头，OSS对带有`Origin`头的请求消息会首先进行跨域规则的验证。

即客户端进行表单直接上传到OSS会产生跨域请求，需要为Bucket设置跨域规则（CORS），支持Post方法。

具体操作步骤请参见设置跨域访问。

步骤四：体验上传回调

- 启动应用服务器

  在`/home/aliyun/aliyun-oss-appserver-java`目录下，执行`mvn package`命令编译打包，然后执行命令启动应用服务器：

  ```
  java -jar target/appservermaven-1.0.0.jar 1234
  ```

  也可以在PC端使用`Eclipse/Intellij IDEA`等IDE工具导出`jar包`，然后将`jar`包拷贝到应用服务器，再执行`jar`包启动应用服务器。

- 启动客户端

  在PC侧的客户端源码目录中，打开`index.html` 文件。

单击**选择文件**，选择指定类型的文件，单击**开始上传**。上传成功后，显示回调服务器返回的内容。

应用服务器核心代码解析

应用服务器源码包含了签名直传服务和上传回调服务两个功能。

- 签名直传服务

  签名直传服务响应客户端发送给应用服务器的`GET`消息，代码片段如下：

```
protected void doGet(HttpServletRequest request, HttpServletResponse
 response)
   throws ServletException, IOException {

  String accessId = "<yourAccessKeyId>"; // 请填写您的AccessKeyId。
  String accessKey = "<yourAccessKeySecret>"; // 请填写您的AccessKeyS
ecret。
  String endpoint = "oss-cn-hangzhou.aliyuncs.com"; // 请填写您的
endpoint。
  String bucket = "bucket-name"; // 请填写您的 bucketname 。
  String host = "http://" + bucket + "." + endpoint; // host的格式为
bucketname.endpoint
  // callbackUrl为 上传回调服务器的URL，请将下面的IP和Port配置为您自己的真实
信息。
  String callbackUrl = "http://88.88.88.88:8888";
  String dir = "user-dir-prefix/"; // 用户上传文件时指定的前缀。

  OSSClient client = new OSSClient(endpoint, accessId, accessKey);
  try {
   long expireTime = 30;
   long expireEndTime = System.currentTimeMillis() + expireTime *
1000;
   Date expiration = new Date(expireEndTime);
   PolicyConditions policyConds = new PolicyConditions();
   policyConds.addConditionItem(PolicyConditions.COND_CONTE
NT_LENGTH_RANGE, 0, 1048576000);
   policyConds.addConditionItem(MatchMode.StartWith, PolicyCond
itions.COND_KEY, dir);

   String postPolicy = client.generatePostPolicy(expiration,
policyConds);
   byte[] binaryData = postPolicy.getBytes("utf-8");
   String encodedPolicy = BinaryUtil.toBase64String(binaryData);
   String postSignature = client.calculatePostSignature(postPolicy);

   Map<String, String> respMap = new LinkedHashMap<String, String
>();
   respMap.put("accessid", accessId);
   respMap.put("policy", encodedPolicy);
   respMap.put("signature", postSignature);
   respMap.put("dir", dir);
   respMap.put("host", host);
   respMap.put("expire", String.valueOf(expireEndTime / 1000));
   // respMap.put("expire", formatISO8601Date(expiration));
```

```
    JSONObject jasonCallback = new JSONObject();
    jasonCallback.put("callbackUrl", callbackUrl);
    jasonCallback.put("callbackBody",
      "filename=${object}&size=${size}&mimeType=${mimeType}&height=${
imageInfo.height}&width=${imageInfo.width}");
    jasonCallback.put("callbackBodyType", "application/x-www-form-
urlencoded");
    String base64CallbackBody = BinaryUtil.toBase64String(jasonCallb
ack.toString().getBytes());
    respMap.put("callback", base64CallbackBody);

    JSONObject ja1 = JSONObject.fromObject(respMap);
    // System.out.println(ja1.toString());
    response.setHeader("Access-Control-Allow-Origin", "*");
    response.setHeader("Access-Control-Allow-Methods", "GET, POST");
    response(request, response, ja1.toString());

  } catch (Exception e) {
   // Assert.fail(e.getMessage());
   System.out.println(e.getMessage());
  }
 }
```

- 上传回调服务

  上传回调服务响应OSS发送给应用服务器的POST消息，代码片段如下：

```
protected boolean VerifyOSSCallbackRequest(HttpServletRequest
request, String ossCallbackBody)
    throws NumberFormatException, IOException {
  boolean ret = false;
  String autorizationInput = new String(request.getHeader("
Authorization"));
  String pubKeyInput = request.getHeader("x-oss-pub-key-url");
  byte[] authorization = BinaryUtil.fromBase64String(autorizati
onInput);
  byte[] pubKey = BinaryUtil.fromBase64String(pubKeyInput);
  String pubKeyAddr = new String(pubKey);
  if (!pubKeyAddr.startsWith("http://gosspublic.alicdn.com/")
    && !pubKeyAddr.startsWith("https://gosspublic.alicdn.com/")) {
   System.out.println("pub key addr must be oss addrss");
   return false;
  }
  String retString = executeGet(pubKeyAddr);
  retString = retString.replace("-----BEGIN PUBLIC KEY-----", "");
  retString = retString.replace("-----END PUBLIC KEY-----", "");
  String queryString = request.getQueryString();
  String uri = request.getRequestURI();
  String decodeUri = java.net.URLDecoder.decode(uri, "UTF-8");
  String authStr = decodeUri;
  if (queryString != null && !queryString.equals("")) {
   authStr += "?" + queryString;
  }
  authStr += "\n" + ossCallbackBody;
  ret = doCheck(authStr, authorization, retString);
  return ret;
 }

 protected void doPost(HttpServletRequest request, HttpServle
tResponse response)
    throws ServletException, IOException {
```

```
  String ossCallbackBody = GetPostBody(request.getInputStream(),
    Integer.parseInt(request.getHeader("content-length")));
  boolean ret = VerifyOSSCallbackRequest(request, ossCallbackBody);
  System.out.println("verify result : " + ret);
  // System.out.println("OSS Callback Body:" + ossCallbackBody);
  if (ret) {
   response(request, response, "{\"Status\":\"OK\"}", HttpServle
tResponse.SC_OK);
  } else {
   response(request, response, "{\"Status\":\"verdify not ok\"}",
HttpServletResponse.SC_BAD_REQUEST);
  }
 }
```

1.4.5 Python

本文以Python语言为例，讲解在服务端通过Python代码完成签名，并且设置上传回调，然后通过表单直传数据到OSS。

前提条件

- 应用服务器对应的域名可通过公网访问。
- 确保应用服务器已经安装Python 2.6以上版本（执行命令python --version进行查看）。
- 确保PC端浏览器支持JavaScript。

步骤一：配置应用服务器

下载应用服务器源码(Python版本)：*aliyun-oss-appserver-python-master.zip* 。

将源码下载到应用服务器的硬盘，本示例中以Ubuntu 16.04为例，放置到/home/aliyun/aliyun-oss-appserver-python目录下。进入该目录，打开源码文件appserver.py，修改如下的代码片段：

```
# 请填写您的AccessKeyId。
access_key_id = '<yourAccessKeyId>'
# 请填写您的AccessKeySecret。
access_key_secret = '<yourAccessKeySecret>'
# host的格式为 bucketname.endpoint ，请替换为您的真实信息。
host = 'http://bucket-name.oss-cn-hangzhou.aliyuncs.com';
# callback_url为 上传回调服务器的URL，请将下面的IP和Port配置为您自己的真实URL信息。
callback_url = "http://88.88.88.88:8888";
# 用户上传文件时指定的前缀。
upload_dir = 'user-dir-prefix/'
```

- access_key_id ： 设置您的AccessKeyId。
- access_key_secret ： 设置您的AessKeySecret。

- host ： 格式为`bucketname.endpoint`，例如`bucket-name.oss-cn-hangzhou.aliyuncs.com`。关于Endpoint的介绍，请参见*Endpoint访问域名*。
- callback_url ： 设置上传回调URL，即回调服务器地址，用于处理应用服务器与OSS之前的通信。OSS会在文件上传完成后，把文件上传信息通过此回调URL发送给应用服务器。本例中修改为：

  `var callbackUrl string ="http://11.22.33.44:1234";`
- upload_dir ： 指定上传文件的前缀，以便区别于其他文件以免发生冲突，您也可以填写空值。

步骤二：配置客户端

下载客户端源码：*aliyun-oss-appserver-js-master.zip* 。

下载客户端源码到PC侧的本地硬盘。本例中以`D:\aliyun\aliyun-oss-appserver-js`目录为例。

进入该目录，打开`upload.js`文件，找到下面的代码语句：

```
// serverUrl是 用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
serverUrl = 'http://88.88.88.88:8888'
```

将变量`severUrl`改成应用服务器的地址，客户端可以通过它可以获取签名直传Policy等信息。如本例中可修改为：

```
// serverUrl是 用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
serverUrl = 'http://11.22.33.44:1234'
```

步骤三： 修改**CORS**

从浏览器向OSS发出的请求消息带有`Origin`的消息头，OSS对带有`Origin`头的请求消息会首先进行跨域规则的验证。

即客户端进行表单直接上传到OSS会产生跨域请求，需要为Bucket设置跨域规则（CORS），支持Post方法。

具体操作步骤请参见设置跨域访问。

步骤四：体验上传回调

- 启动应用服务器

在/home/aliyun-oss-appserver-python目录下，执行命令启动应用服务器：

```
python appserver.py 11.22.33.44 1234
```

- 启动客户端

  在PC侧的客户端源码目录中，打开index.html 文件。

  单击**选择文件**，选择指定类型的文件后，单击**开始上传**。上传成功后，显示回调服务器返回的内容。

应用服务器核心代码解析

应用服务器源码包含了签名直传服务和上传回调服务两个功能。

- 签名直传服务

  签名直传服务响应客户端发送给应用服务器的GET消息，代码片段如下：

```
def do_GET(self):
        print "********************* do_GET "
        token = get_token()
        self.send_response(200)
        self.send_header('Access-Control-Allow-Methods', 'POST')
        self.send_header('Access-Control-Allow-Origin', '*')
        self.send_header('Content-Type', 'text/html; charset=UTF-8')
        self.end_headers()
        self.wfile.write(token)
```

- 上传回调服务

  上传回调服务响应OSS发送给应用服务器的POST消息，代码片段如下：

```
def do_POST(self):
        print "********************* do_POST "
        # get public key
        pub_key_url = ''
        try:
            pub_key_url_base64 = self.headers['x-oss-pub-key-url']
            pub_key_url = pub_key_url_base64.decode('base64')
            url_reader = urllib2.urlopen(pub_key_url)
            pub_key = url_reader.read()
        except:
            print 'pub_key_url : ' + pub_key_url
            print 'Get pub key failed!'
            self.send_response(400)
            self.end_headers()
            return

        # get authorization
        authorization_base64 = self.headers['authorization']
        authorization = authorization_base64.decode('base64')

        # get callback body
        content_length = self.headers['content-length']
```

```
        callback_body = self.rfile.read(int(content_length))

        # compose authorization string
        auth_str = ''
        pos = self.path.find('?')
        if -1 == pos:
            auth_str = self.path + '\n' + callback_body
        else:
            auth_str = urllib2.unquote(self.path[0:pos]) + self.path
[pos:] + '\n' + callback_body

        # verify authorization
        auth_md5 = md5.new(auth_str).digest()
        bio = BIO.MemoryBuffer(pub_key)
        rsa_pub = RSA.load_pub_key_bio(bio)
        try:
            result = rsa_pub.verify(auth_md5, authorization, 'md5')
        except e:
            result = False

        if not result:
            print 'Authorization verify failed!'
            print 'Public key : %s' % (pub_key)
            print 'Auth string : %s' % (auth_str)
            self.send_response(400)
            self.end_headers()
            return

        # do something accoding to callback_body

        # response to OSS
        resp_body = '{"Status":"OK"}'
        self.send_response(200)
        self.send_header('Content-Type', 'application/json')
        self.send_header('Content-Length', str(len(resp_body)))
        self.end_headers()
        self.wfile.write(resp_body)
```

1.4.6 Ruby

本文以Ruby语言为例，讲解在服务端通过Ruby代码完成签名，并且设置上传回调，然后通过表单直传数据到OSS。

前提条件

- 应用服务器对应的域名可通过公网访问。
- 确保应用服务器已经安装`Ruby 2.0`以上版本（执行命令`ruby -v`进行查看）。
- 确保PC端浏览器支持JavaScript。

步骤一：配置应用服务器

下载应用服务器源码(Ruby版本)：*aliyun-oss-appserver-ruby-master.zip*

将源码下载到应用服务器的硬盘，本示例中以Ubuntu 16.04为例，放置到/home/aliyun/aliyun-oss-appserver-ruby目录下。进入该目录，打开源码文件appserver.rb，修改如下代码片段：

```
# 请填写您的AccessKeyId。
$access_key_id = '<yourAccessKeyId>'

# 请填写您的AccessKeySecret。
$access_key_secret = '<yourAccessKeySecret>'

# $host的格式为 bucketname.endpoint ，请替换为您的真实信息。
$host = 'http://bucket-name.oss-cn-hangzhou.aliyuncs.com';

# $callbackUrl为 上传回调服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
$callback_url = "http://88.88.88.88:8888";

# 用户上传文件时指定的前缀。
$upload_dir = 'user-dir-prefix/'
```

- $access_key_id ： 设置您的AccessKeyId。
- $access_key_secret ： 设置您的AessKeySecret。
- $host ： 格式为bucketname.endpoint，例如bucket-name.oss-cn-hangzhou.aliyuncs.com。关于Endpoint的介绍，请参见*Endpoint访问域名*。
- $callback_url ： 设置上传回调URL，即回调服务器地址，用于处理应用服务器与OSS之前的通信。OSS会在文件上传完成后，把文件上传信息通过此回调URL发送给应用服务器。本例中修改为：

  $callback_url="http://11.22.33.44:1234";
- $upload_dir ： 设置上传到OSS文件的前缀，以便区别于其他文件从而避免冲突，您也可以填写空值。

步骤二：配置客户端

下载客户端源码：*aliyun-oss-appserver-js-master.zip*

下载客户端源码到PC侧的本地硬盘。本例中以D:\aliyun\aliyun-oss-appserver-js目录为例。

进入该目录，打开upload.js文件，找到下面的代码语句：

```
// serverUrl是 用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
```

```
serverUrl = 'http://88.88.88.88:8888'
```

将变量severUrl改成应用服务器的地址，客户端可以通过它获取签名直传Policy等信息。如本例中可修改为：

```
// serverUrl是 用户获取 '签名和Policy' 等信息的应用服务器的URL，请将下面的IP和Port配置为您自己的真实信息。
serverUrl = 'http://11.22.33.44:1234'
```

步骤三： 修改**CORS**

从浏览器向OSS发出的请求消息带有Origin的消息头，OSS对带有Origin头的请求消息会首先进行跨域规则的验证。

即客户端进行表单直接上传到OSS会产生跨域请求，需要为Bucket设置跨域规则（CORS），支持Post方法。

具体操作步骤请参见设置跨域访问。

步骤四：体验上传回调

- 启动应用服务器

  在/home/aliyun/aliyun-oss-appserver-ruby目录下，执行以下命令启动应用服务器：

  ```
  ruby appserver.rb 11.22.33.44 1234
  ```

- 启动客户端

  在PC侧的客户端源码目录中，打开index.html 文件。

  单击**选择文件**，选择指定类型的文件后，单击**开始上传**。上传成功后，显示回调服务器返回的内容。

应用服务器核心代码解析

应用服务器源码包含了签名直传服务和上传回调服务两个功能。

- 签名直传服务

  签名直传服务响应客户端发送给应用服务器的GET消息，代码片段如下：

  ```
  def get_token()
      expire_syncpoint = Time.now.to_i + $expire_time

      expire = Time.at(expire_syncpoint).utc.iso8601()
      response.headers['expire'] = expire

      policy_dict = {}
  ```

```
    condition_arrary = Array.new
    array_item = Array.new
    array_item.push('starts-with')
    array_item.push('$key')
    array_item.push($upload_dir)
    condition_arrary.push(array_item)
    policy_dict["conditions"] = condition_arrary
    policy_dict["expiration"] = expire
    policy = hash_to_jason(policy_dict)
    policy_encode = Base64.strict_encode64(policy).chomp;
    h = OpenSSL::HMAC.digest('sha1', $access_key_secret, policy_enc
ode)
    hs = Digest::MD5.hexdigest(h)
    sign_result = Base64.strict_encode64(h).strip()

    callback_dict = {}
    callback_dict['callbackBodyType'] = 'application/x-www-form-
urlencoded';
    callback_dict['callbackBody'] = 'filename=${object}&size=${size
}&mimeType=${mimeType}&height=${imageInfo.height}&width=${imageInfo.
width}';
    callback_dict['callbackUrl'] = $callback_url;
    callback_param = hash_to_jason(callback_dict)
    base64_callback_body = Base64.strict_encode64(callback_param);

    token_dict = {}
    token_dict['accessid'] = $access_key_id
    token_dict['host'] = $host
    token_dict['policy'] = policy_encode
    token_dict['signature'] = sign_result
    token_dict['expire'] = expire_syncpoint
    token_dict['dir'] = $upload_dir
    token_dict['callback'] = base64_callback_body
    response.headers["Access-Control-Allow-Methods"] = "POST"
    response.headers["Access-Control-Allow-Origin"] = "*"
    result = hash_to_jason(token_dict)

    result
end

get '/*' do
  puts "********************* GET "
  get_token()
end
```

- 上传回调服务

  上传回调服务响应OSS发送给应用服务器的POST消息，代码片段如下：

```
post '/*' do
  puts "********************* POST"
  pub_key_url = Base64.decode64(get_header('x-oss-pub-key-url'))
  pub_key = get_public_key(pub_key_url)
  rsa = OpenSSL::PKey::RSA.new(pub_key)

  authorization = Base64.decode64(get_header('authorization'))
  req_body = request.body.read
  if request.query_string.empty? then
    auth_str = CGI.unescape(request.path) + "\n" + req_body
  else
```

```
    auth_str = CGI.unescape(request.path) + '?' + request.query_stri
ng + "\n" + req_body
  end

  valid = rsa.public_key.verify(
    OpenSSL::Digest::MD5.new, authorization, auth_str)

  if valid
    #body({'Status' => 'OK'}.to_json)
    body(hash_to_jason({'Status' => 'OK'}))
  else
    halt 400, "Authorization failed!"
  end
end
```

2 数据迁移

2.1 如何将HDFS容灾备份到OSS

背景

当前业界有很多公司是以Hadoop技术构建数据中心，而越来越多的公司和企业希望将业务顺畅地迁移到云上。

在阿里云上使用最广泛的存储服务是对象存储OSS。OSS的数据迁移工具ossimport2可以将您本地或第三方云存储服务上的文件同步到OSS上，但这个工具无法读取Hadoop文件系统的数据，从而发挥Hadoop分布式的特点。并且，该工具只支持本地文件，需要将HDFS上的文件先下载到本地，再通过工具上传，整个过程耗时又耗力。

阿里云E-MapReduce团队开发的Hadoop数据迁移工具**emr-tools**，能让您从Hadoop集群直接迁移数据到OSS上。

本文介绍如何快速地将Hadoop文件系统（HDFS）上的数据迁移到OSS。

前提条件

确保当前机器可以正常访问您的Hadoop集群，即能够用Hadoop命令访问HDFS。

```
hadoop fs -ls /
```

Hadoop数据迁移到OSS

1. 下载*emr-tools*。

   说明：

   emr-tools兼容Hadoop 2.4.x、2.5.x、2.6.x、2.7.x版本，如果有其他Hadoop版本兼容性的需求，请提交工单。

2. 解压缩工具到本地目录。

   ```
   tar jxf emr-tools.tar.bz2
   ```

3. 复制HDFS数据到OSS上。

   ```
   cd emr-tools
   ./hdfs2oss4emr.sh /path/on/hdfs oss://accessKeyId:accessKeySecret@
   bucket-name.oss-cn-hangzhou.aliyuncs.com/path/on/oss
   ```

   参数说明如下。

参数	说明
accessKeyId	访问OSS API的密钥。
accessKeySecret	
bucket-name.oss-cn-hangzhou.aliyuncs.com	OSS的访问域名，包括bucket名称和endpoint地址。

系统将启动一个Hadoop MapReduce任务（DistCp）。

4. 运行完毕之后会显示本次数据迁移的信息。信息内容类似如下所示。

```
17/05/04 22:35:08 INFO mapreduce.Job: Job job_1493800598643_0009
completed successfully
17/05/04 22:35:08 INFO mapreduce.Job: Counters: 38
 File System Counters
        FILE: Number of bytes read=0
        FILE: Number of bytes written=859530
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=263114
        HDFS: Number of bytes written=0
        HDFS: Number of read operations=70
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=14
        OSS: Number of bytes read=0
        OSS: Number of bytes written=258660
        OSS: Number of read operations=0
        OSS: Number of large read operations=0
        OSS: Number of write operations=0
 Job Counters
        Launched map tasks=7
        Other local map tasks=7
        Total time spent by all maps in occupied slots (ms)=60020
        Total time spent by all reduces in occupied slots (ms)=0
        Total time spent by all map tasks (ms)=30010
        Total vcore-milliseconds taken by all map tasks=30010
        Total megabyte-milliseconds taken by all map tasks=45015000
 Map-Reduce Framework
        Map input records=10
        Map output records=0
        Input split bytes=952
        Spilled Records=0
        Failed Shuffles=0
        Merged Map outputs=0
        GC time elapsed (ms)=542
        CPU time spent (ms)=14290
        Physical memory (bytes) snapshot=1562365952
        Virtual memory (bytes) snapshot=17317421056
        Total committed heap usage (bytes)=1167589376
 File Input Format Counters
        Bytes Read=3502
 File Output Format Counters
        Bytes Written=0
 org.apache.hadoop.tools.mapred.CopyMapper$Counter
        BYTESCOPIED=258660
        BYTESEXPECTED=258660
```

```
          COPY=10
copy from /path/on/hdfs to oss://accessKeyId:accessKeySecret@bucket-
name.oss-cn-hangzhou.aliyuncs.com/path/on/oss does succeed !!!
```

5. 您可以用osscmd等工具查看OSS上数据情况。

```
osscmd ls oss://bucket-name/path/on/oss
```

OSS数据迁移到Hadoop

如果您已经在阿里云上搭建了Hadoop集群，可以使用如下命令把数据从OSS上迁移到新的Hadoop集群。

```
./hdfs2oss4emr.sh oss://accessKeyId:accessKeySecret@bucket-name.oss-cn
-hangzhou.aliyuncs.com/path/on/oss /path/on/new-hdfs
```

更多使用场景

除了线下的集群，在ECS上搭建的Hadoop集群也可以使用emr-tools，将自建集群迅速地迁移到*E-MapReduce*服务上。

如果您的集群已经在ECS上，但是在经典网络中，无法和VPC中的服务做很好的互操作，所以想把集群迁移到VPC中。可以按照如下步骤迁移：

1. 使用emr-tools迁移数据到OSS上。
2. 在VPC环境中新建一个集群（自建或使用E-MapReduce服务）。
3. 将数据从OSS上迁移到新的HDFS集群中。

2.2 使用OssImport迁移数据

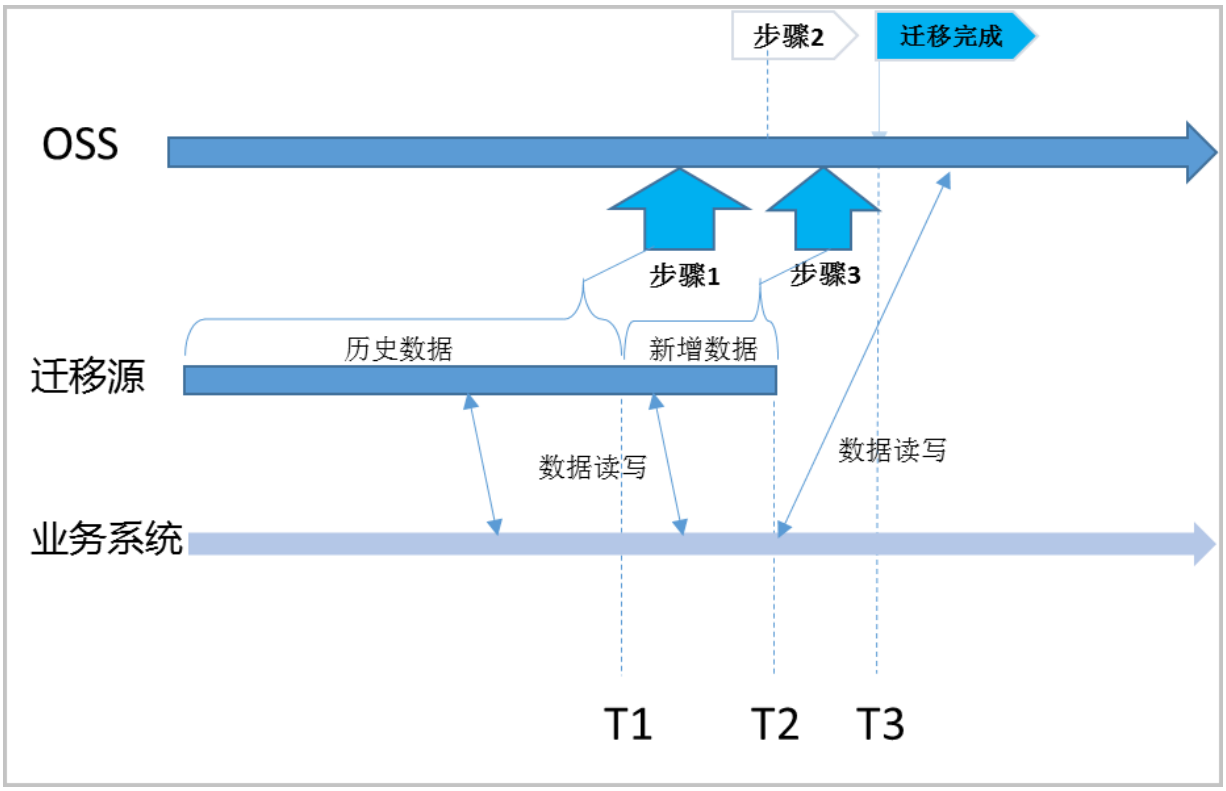
本文向您介绍如何使用OssImport将数据从第三方存储（或OSS）迁移到OSS。

工具选择：单机模式和分布式模式

OssImport有单机模式和分布式模式两种部署方式。一般建议使用分布式模式。您参考OssImport官网指导文档，即可完成迁移过程。本文介绍您在整体迁移方案中可能会关注的内容，以及可以参考的官网文档资源。

迁移方案（从第三方存储迁移到OSS）

以下步骤可以完成从其它存储到OSS的无缝切换（参考官网支持的第三方存储类型*OssImport 架构及配置*）：



具体步骤如下：

1. 全量迁移T1前的历史数据，请参考：*OssImport* 架构及配置。
   - 记录迁移开始时间T1（注意为Unix时间戳，即自1970年1月1日UTC零点以来的秒数，通过命令`date +%s`获取）。
   - 迁移指导说明参考OssImport官网文档，请参考迁移工具-分布式。
2. 打开OSS镜像回源，并将读写切换到OSS，迁移源不再新增数据。
   - 步骤1迁移完成后，在OSS控制台打开*OSS*镜像回源功能，回源地址为迁移源（第三方存储）。
   - 在业务系统读写切换到OSS，假设业务系统修改好的时间为T2。
   - 此时T1前的数据从OSS读取，T1后的数据，OSS利用镜像回源从第三方服务读取，而新数据完全写入OSS。
3. 快速迁移T1~T2到数据。
   - 在步骤2完成后，第三方存储不会再新增数据，数据读写已切到OSS。
   - 修改配置文件`job.cfg`的配置项`importSince=T1`，新发起迁移job，迁移T1~T2数据。
4. 步骤3完成后，即完成迁移全过程。

- 步骤3完成后，您业务的所有的读写都在OSS上，第三方存储只是一份历史数据，您可以根据需要决定保留或删除。
- OssImport负责数据的迁移和校验，不会删除任何数据。

迁移成本

迁移过程涉及到成本一般有ECS费用、流量费用、存储费用、时间成本。其中，多数情况下，比如数据超过TB级别，存储成本和迁移时间成正比，而ECS费用相对流量、存储费用较小。

环境准备

假设您有如下迁移需求：

需求项	需求情况
迁移源	AWS S3东京
迁移目的	OSS香港
数据量	500TB
迁移时间要求	1周内完成迁移

您需要准备的环境：

环境项	说明
开通OSS	开通OSS步骤如下： 1. 使用您的账号创建香港地域的OSS Bucket。 2. 在RAM控制台创建子帐号，并授权该子账号可访问OSS。保存AccessKeyID和AccessKeySecret。
购买ECS	购买OSS同区域(香港)的ECS，一般普通的2核4G机型即可，如果迁移后ECS需释放，建议按需购买ECS。正式迁移时的ECS数量，请参考关于迁移用的*ECS*数量。
配置OssImport	说明： *destDomain*参数，即OSS目的endpoint，建议设置为OSS内网的endpoint，避免从ECS数据上传到OSS时产生外网流量费用。具体的endpoint，请参考访问域名和数据中心。

环境项	说明
迁移过程	1. 在ECS上搭建OssImport分布式环境。 2. 使用OssImport从东京AWS S3下载数据到ECS（香港），建议使用外网。 3. 使用OssImport从ECS（香港）将数据上传到OSS（香港），建议使用内网。

关于迁移用的ECS数量

您需根据迁移需求，计算您需要用来做迁移的ECS数量：

- 假设您需要迁移的数据量是X TB，要求迁移完成时间Y天，单台迁移速度Z Mbps（每天迁移约Z/100 TB数据）。
- 则正式迁移时，ECS大致需要X/Y/(Z/100)台。

假设单台ECS迁移速度达到200Mbps(即每天约迁移2TB数据)。则上面Case中，ECS约需36台（500/7/2）。

OssImport迁移步骤

配置参考

您可以阅读官网指导文档，了解配置定义*OssImport* 架构及配置、操作步骤分布式部署，在开始前请关注如下信息：

- OssImport下载：在Master下载OssImport分布式版，且master、workers都使用同样的ssh账号、密码。（worker上不用单独下载OssImport，运行`deploy`命令时，OssImport会分发到worker上，请参考分布式部署。）
- Java环境：确认Master和worker都已安装。

> 说明：
> 作为worker的ECS也需要安装Java。

- 设置workdir：通过`conf/sys.properties`的配置项`workingDir`指定，请参考分布式部署。

> 说明：
> workdir的设置请参考官网文档，不要设置成OssImport包所在的路径，同时尽量不要设置为已有内容的目录。

- 并发控制：conf/job.cfg的配置项taskObjectCountLimit，conf/sys.properties的配置项workerTaskThreadNum，请参考*OssImport* 架构及配置。

  如果小文件比较多，单台ECS迁移速度上不去且CPU load不高，可以参考调高workerTaskThreadNum、调低taskObjectCountLimit查看效果。
- 其他操作过程遇到问题，一般都是配置问题，请参考分布式部署，并可以查看master和worker上workdir/Logs的日志文件。

前期测试

建议您先搭建小型环境（比如2~3台ECS），迁移少量数据以验证配置正确与否。单台ECS迁移带宽能否达到预期，比如200Mbps，如您对迁移时间明确无要求，可不关注。

带宽查看：您可使用iftop或Nload（需先安装，如yum install ***），ECS控制台带宽统计有延时。

> 说明：
> 如果测试时文件数目较少，可能无法验证并发性，可以减少参数taskObjectCountLimit，比如减少到：文件数/workerTaskThreadNum/worker总数。

迁移步骤

1. 迁移历史数据。
   - 清任务、清配置。
   - 操作过程请参考分布式部署。
   - 开始迁移。
   - 您可在*OSS*控制台OSS Bucket中查看实际迁移完成的数据。
2. 设置镜像回源，客户业务系统读写切到OSS。
   - 在*OSS*控制台打开*OSS*镜像回源，回源地址为迁移源（第三方存储）。
   - 客户业务系统读写切换到OSS，假设业务系统修改好的时间为T2，则T2后不再有新数据写到迁移源。
3. 迁移剩余的数据。

   修改job.cfg配置项importSince=T1，请参考*OssImport* 架构及配置，迁移剩余数据（T1~T2）特殊情况下，也可以使用job.cfg中importSince = 0, isSkipExistFile=true进行再次迁移，请参考*OssImport* 架构及配置。

关于迁移速度

- 单台迁移速度：如单台迁移速度不理想（比如小于200Mbps、且CPU load不高时），可参考官网文档和上文，优化并发控制参数，即`conf/job.cfg`的配置项`taskObjectCountLimit`，`conf/sys.properties`的配置项`workerTaskThreadNum`。
- 迁移ECS数量：参考*ECS数量*估算（相对于流量、存储、时间成本，ECS费用，在迁移总成本中占比较少）。加大ECS数量，会减少迁移时间。

OssImport分布式相关官网文档

序号	官网文档
1	*OssImport* 分布式部署
2	*OssImport* 架构及配置
3	*OssImport* 最佳实践
4	*OssImport* 常见问题

2.3 Amazon S3数据迁移到OSS

OSS提供了S3 API的兼容性，可以让您的数据从Amazon S3无缝迁移到阿里云OSS上。从Amazon S3迁移到OSS后，您仍然可以使用S3 API访问OSS，仅需要对S3的客户端应用进行如下改动：

1. 获取OSS主账号或子账号的AccessKeyId和AccessKeySecret，并在您使用的客户端和SDK中配置您申请的AccessKeyId与AccessKeySecret。
2. 设置客户端连接的endpoint为OSS endpoint。OSS endpoint列表请参见访问域名和数据中心。

迁移步骤

从第三方存储迁移到OSS的具体操作步骤，请您参见使用*OssImport*迁移数据。

迁移后用S3 API访问OSS

从S3迁移到OSS后，您使用S3 API访问OSS时，需要注意以下几点：

- Path style和Virtual hosted style

  *Virtual hosted style*是指将Bucket放入host header的访问方式。OSS基于安全考虑，仅支持virtual hosted访问方式，所以在S3到OSS迁移后，客户端应用需要进行相应设置。部分S3工具默认使用Path style，也需要进行相应配置，否则可能导致OSS报错禁止访问。

- OSS对权限的定义与S3不完全一致，迁移后如有需要，可对权限进行相应调整。二者的主要区别可参考下表。

  说明：

  - 更详细的区别请参见*ACL*验证流程。
  - OSS仅支持S3中的私有、公共读和公共读写三种ACL模式。

对象	Amazon S3权限	Amazon S3	阿里云OSS
Bucket	READ	拥有Bucket的list权限	对于Bucket下的所有Object，如果某Object没有设置Object权限，则该Object可读
	WRITE	Bucket下的Object可写入或覆盖	• 对于Bucket下不存在的Object，可写入 • 对于Bucket下存在的Object，如果该Object没有设置Object权限，则该Object可被覆盖 • 可以initiate multipart upload
	READ_ACP	读取Bucket ACL	读取Bucket ACL，仅Bucket owner和授权子账号拥有此权限
	WRITE_ACP	设置Bucket ACL	设置Bucket ACL，仅Bucket owner和授权子账号拥有此权限
Object	READ	Object可读	Object可读
	WRITE	N/A	Object可以被覆盖
	READ_ACP	读取Object ACL	读取Object ACL，仅Bucket owner和授权子账号拥有此权限

对象	Amazon S3权限	Amazon S3	阿里云OSS
	WRITE_ACP	设置Object ACL	设置Object ACL，仅Bucket owner和授权子账号拥有此权限

- 存储类型

  OSS支持标准（Standard）、低频访问（IA）和归档存储（Archive）三种存储类型，分别对应Amazon S3中的STANDARD、STANDARD_IA和GLACIER。

  与Amazon S3不同的是，OSS不支持在上传object时直接指定存储类型，object的存储类型由bucket的存储类型指定。OSS支持标准、低频访问和归档三种Bucket类型，Object的存储类型还可以通过lifecycle进行转换。

  归档存储类型的object在读取之前，要先使用Restore请求进行解冻操作。与S3不同，OSS会忽略S3 API中的解冻天数设置，解冻状态默认持续1天，用户可以延长到最多7天，之后，Object又回到初始时的冷冻状态。

- ETag
  - 对于PUT方式上传的Object，OSS Object的ETag与Amazon S3在大小写上有区别。OSS为大写，而S3为小写。如果客户端有关于ETag的校验，请忽略大小写。
  - 对于分片上传的Object，OSS的ETag计算方式与S3不同。

兼容的S3 API

- Bucket操作：
  - Delete Bucket
  - Get Bucket （list objects）
  - Get Bucket ACL
  - Get Bucket lifecycle
  - Get Bucket location
  - Get Bucket logging
  - Head Bucket
  - Put Bucket
  - Put Bucket ACL
  - Put Bucket lifecycle

- Put Bucket logging

- Object操作：
  - Delete Object
  - Delete Objects
  - Get Object
  - Get Object ACL
  - Head Object
  - Post Object
  - Put Object
  - Put Object Copy
  - Put Object ACL
- Multipart操作：
  - Abort Multipart Upload
  - Complete Multipart Upload
  - Initiate Multipart Upload
  - List Parts
  - Upload Part
  - Upload Part Copy

3 数据备份和容灾

3.1 备份存储空间

针对存放在OSS上的数据，阿里云提供多种数据备份方式，以满足不同场景的备份需求。

OSS云上备份方式有如下2种：

- 跨区域复制（提供控制台配置操作以及基于API/SDK两种模式）
- 基于OssImport工具

通过跨区域复制进行备份

使用场景

参见管理跨区域复制开发指南。

控制台设置操作

参见设置跨区域复制操作指南。

常见问题

特别说明

- 源Bucket和目标Bucket属于同一个用户，且分属不同的区域。
- 源Bucket、目标Bucket存储类型都不是归档类型。
- 若要在同一区域的Bucket之间进行数据同步，则可以通过OSS的SDK/API编码实现。

通过使用OssImport工具进行备份

OssImport工具可以将本地、其它云存储的数据迁移到OSS，它有以下特点：

- 支持丰富的数据源，有本地、七牛、百度BOS、AWS S3、Azure Blob、又拍云、腾讯云COS、金山KS3、HTTP、OSS等，并可根据需要扩展。
- 支持断点续传。
- 支持流量控制。
- 支持迁移指定时间后的文件、特定前缀的文件。
- 支持并行数据下载、上传。
- 支持单机模式和分布式模式。单机模式部署简单使用方便，分布式模式适合大规模数据迁移

使用场景

参见数据迁移。

安装部署

参见说明及配置 、单机部署 、分布式部署。

常见问题

参见常见问题。

特别说明

- 不同账户的不同Bucket之间，数据量很大，10TB级别以上的情况，建议使用分布式的版本。
- 利用增量模式在OSS之间同步文件修改操作，OssImport只能同步文件的修改操作(put/apppend/multipart)，不能同步读取和删除操作，数据同步的及时性没有具体的 SLA 保证， 慎重选择。
- 如果开通了跨区域复制的地域，则推荐使用跨区域复制进行地域之间的数据同步。

3.2 数据库实时备份到OSS

本文介绍如何通过数据库备份DBS将本地IDC、公网、第三方云数据库、阿里云RDS和阿里云ECS自建数据库实时备份到OSS上。

背景

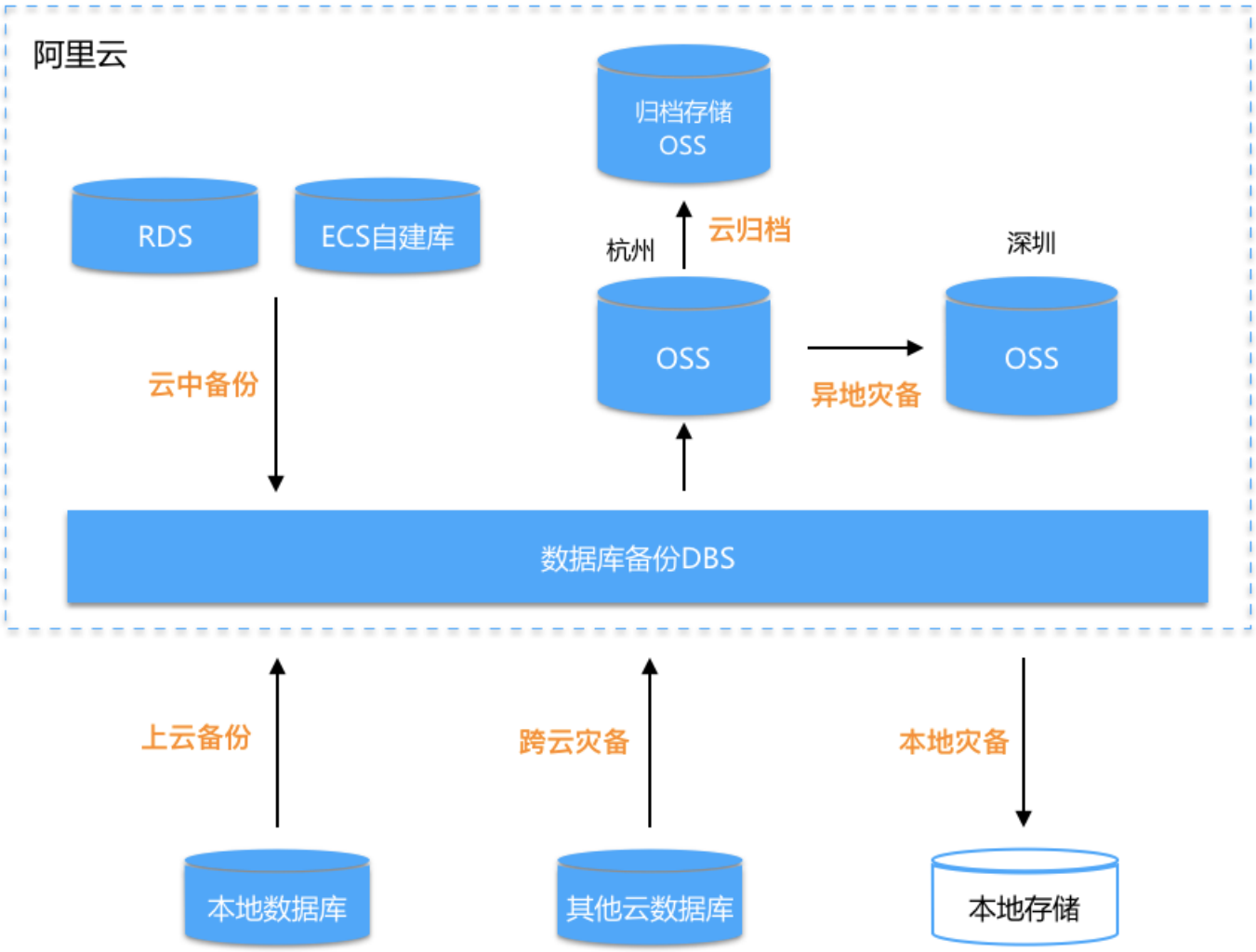
- 对象存储OSS

  对象存储*OSS*提供了标准类型存储，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，也提供了成本更低、存储期限更长的低频访问类型存储和归档类型存储，作为不经常访问数据的备份和归档。对象存储OSS非常适合作为数据库备份的存储介质。

- 数据库备份DBS

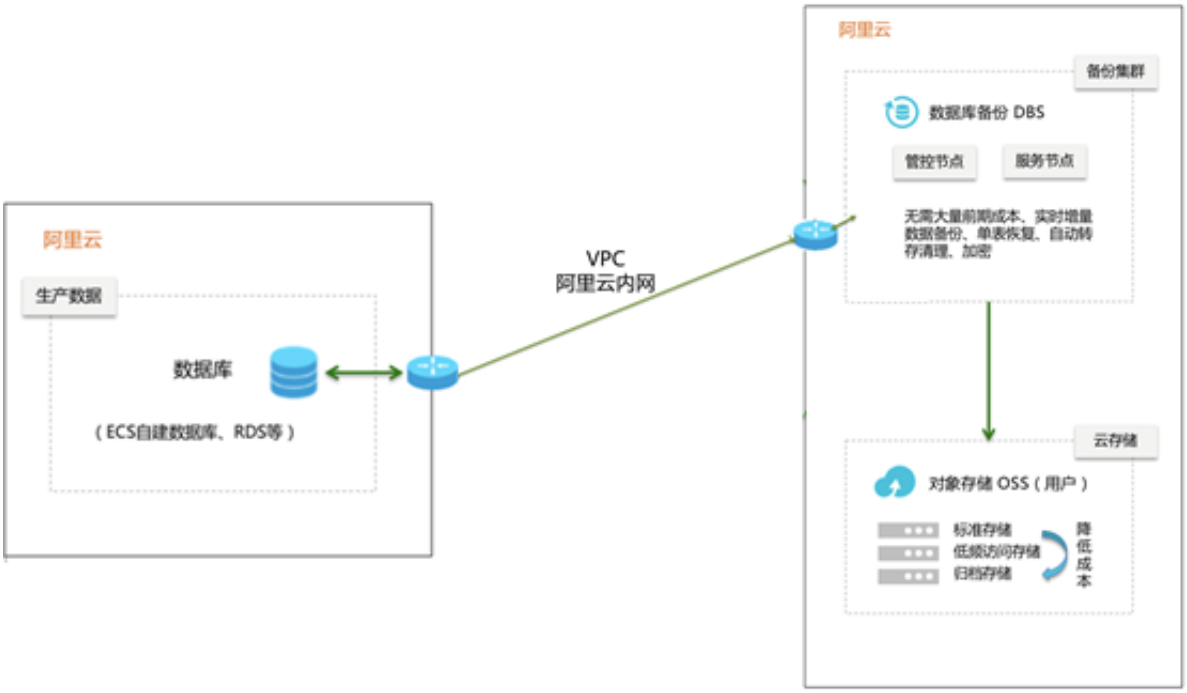
  数据库备份*DBS*是为数据库提供连续数据保护、低成本的备份服务。它可以为多种环境的数据提供强有力的保护，包括企业数据中心、其他云厂商及公共云。数据库备份提供数据备份和操作恢复的整体方案，具备实时增量备份、精确到秒级的数据恢复能力。

应用场景

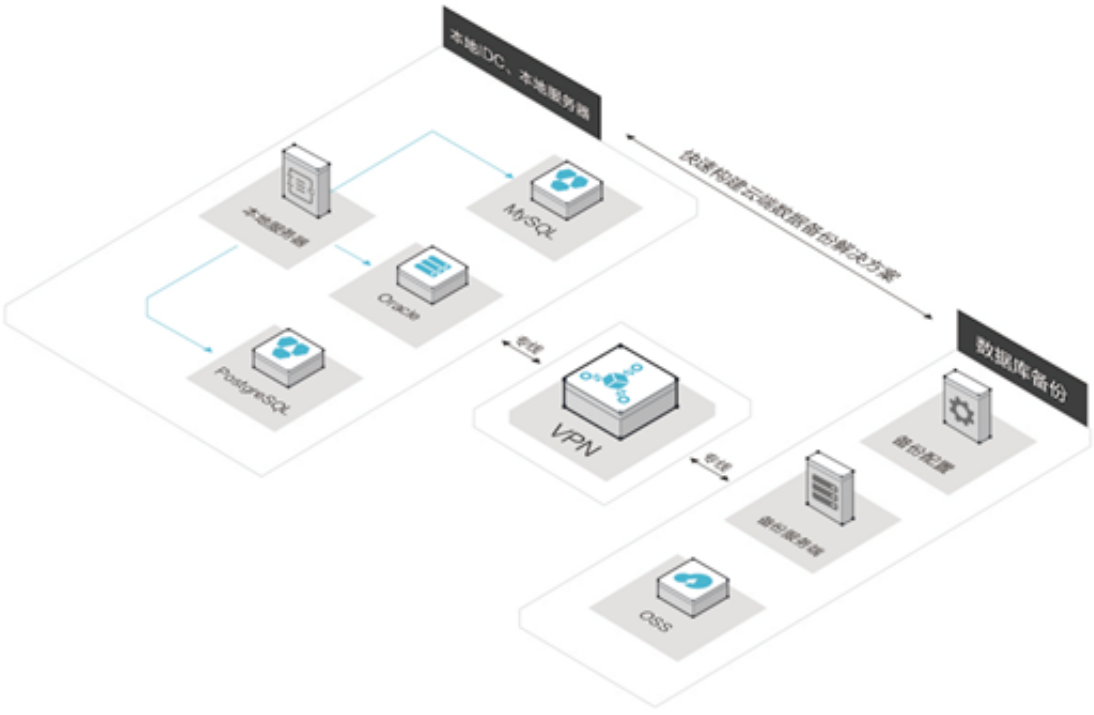


数据库备份到OSS的方案应用场景如下：

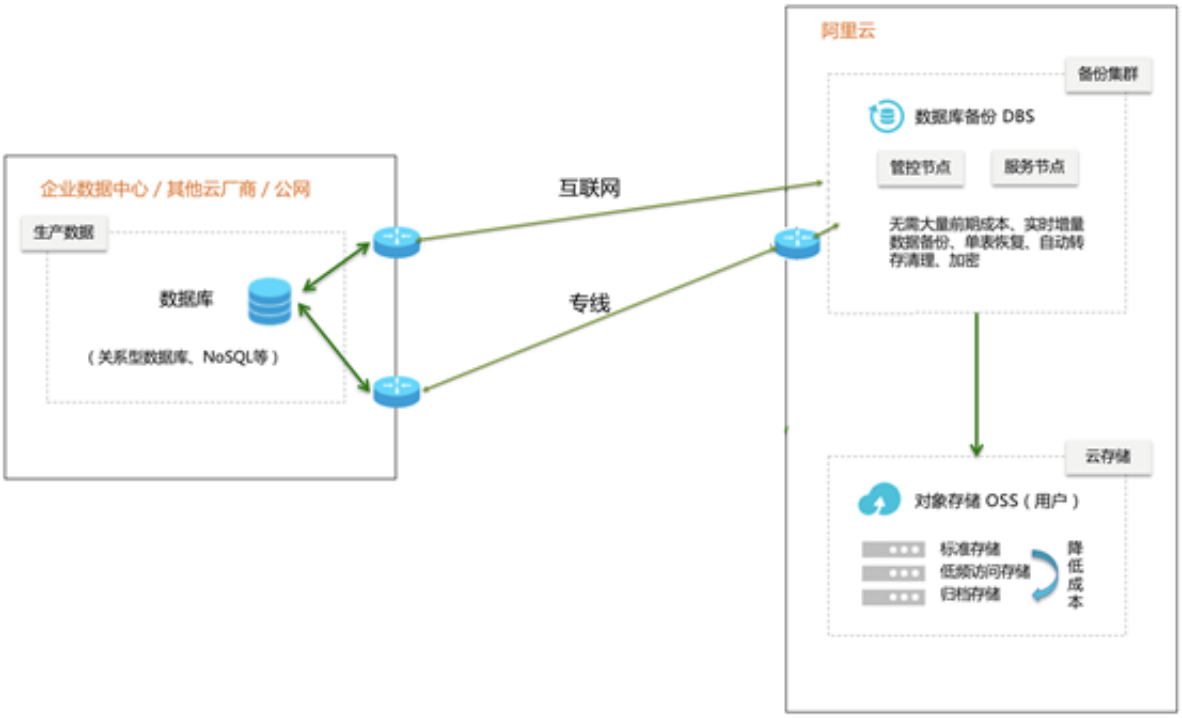
- 阿里云RDS或阿里云ECS自建数据库在OSS上备份或长期归档



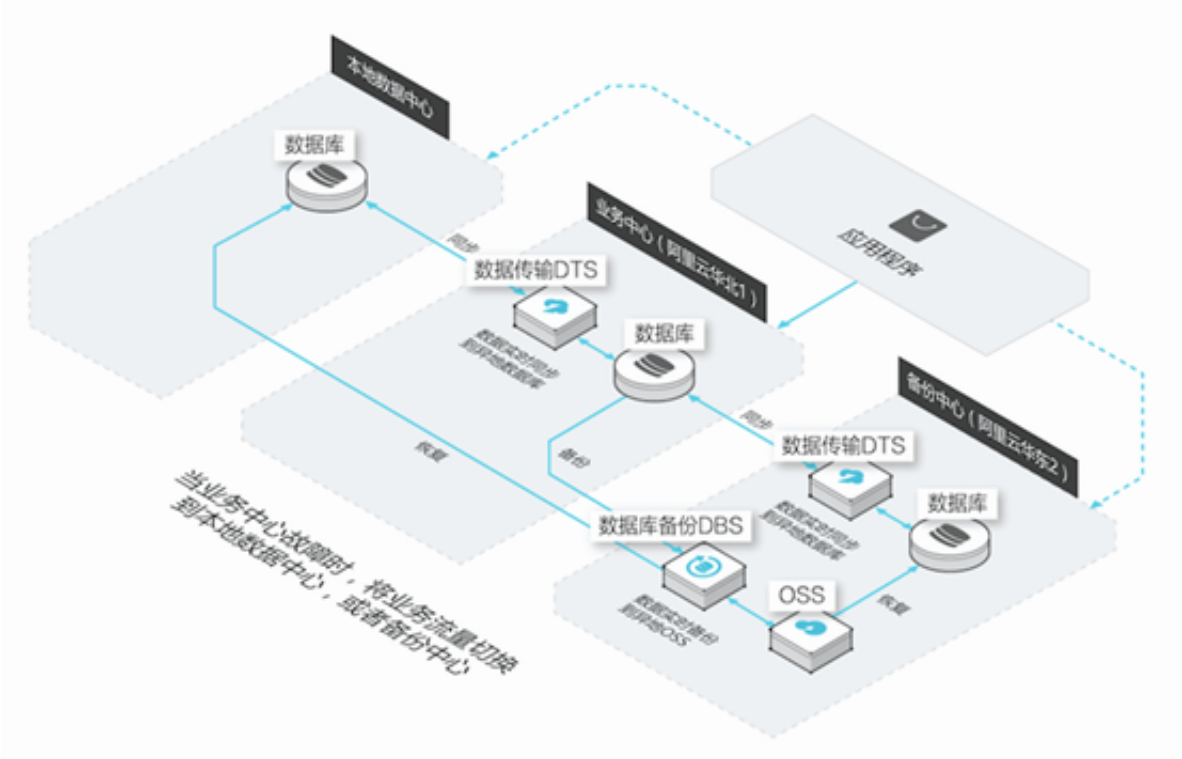
- 自建IDC数据备份上云



- 其他云数据库在阿里云OSS上做多云备份

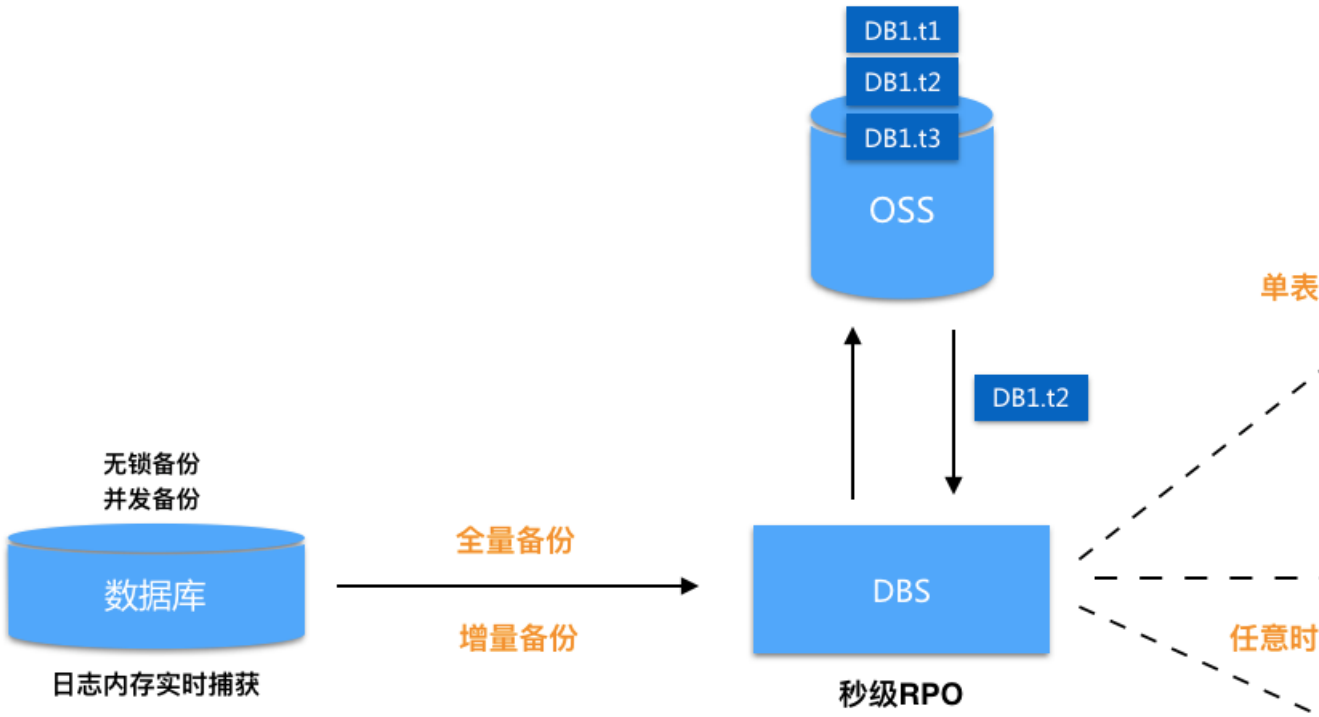


- 数据库做异地备份灾备



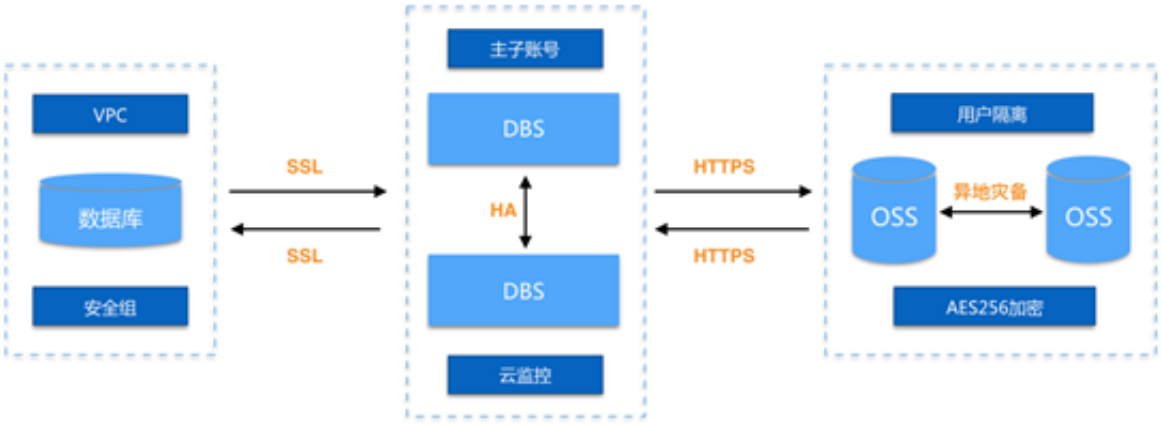
方案优势

- 支持全量或实时增量备份



- 秒级RPO：日志内存实时捕获，CDP实时备份，RPO达到秒级。
- 无锁并发：全程无锁备份、并发备份、数据拉取自适应分片。
- 精准恢复：恢复对象精准匹配，单表恢复，从而大幅降低RTO。
- 灵活恢复：提供可恢复日历及时间轴选择，实现任意时间点恢复。

- 数据强安全高可靠



- 异地灾备：利用OSS的跨区域复制功能，做备份数据的异地灾备，提升数据保护级别。
- 数据传输加密：在传输过程中进行SSL加密，保障数据安全性。
- 数据存储加密：对备份到OSS的数据进行加密存储，保障数据隐私性。
- 随时验证：随时验证数据库备份的有效性。

- 低成本

  - 按需付费：OSS存储空间按需付费，避免一次性投入大量资产。
  - 自动存储分级：OSS提供标准、低频、归档多种类型，全面优化存储成本。
  - 弹性扩展：OSS存储容量弹性扩展，无缝支撑企业在不同发展阶段的性能要求。

- 支持多环境多数据源

  - 支持MySQL、Oracle、SQL Server、MongoDB多种数据库。
  - 支持IDC、第三方云数据库、阿里云RDS、阿里云ECS自建数据库。

方案实施流程

数据库备份到OSS的方案实施流程如下：

1. 创建备份计划。详情请参见数据库备份快速入门中的创建备份计划。
2. 配置备份计划。详情请参见数据库备份快速入门中的配置备份计划。
3. 查看备份计划。详情请参见数据库备份快速入门中的查看备份计划。
4. 恢复数据库。详情请参见数据库备份快速入门中的恢复数据库。

4 通过云存储网关使用OSS服务

4.1 应用场景

对象存储OSS是海量分布式对象存储服务，提供标准、低频、归档存储类型，覆盖从热到冷的不同存储场景。OSS提供RESTful API，您可以从任何位置访问OSS，存储容量和处理能力弹性扩展。

阿里云云存储网关是一款帮助客户在现有本地应用程序、基础设施和数据存储与阿里云的存储服务之间实现无缝集成的数据服务。通过可在本地和云上部署的兼容行业标准存储协议的虚拟设备，云存储网关将现有的存储应用程序和工作负载无缝对接阿里云的存储和计算服务。有关云存储网关更多详情，请参见云存储网关产品详情页。

云存储网关与OSS结合，主要有以下几种应用场景：

云端扩容

云存储网关以阿里云OSS为后端存储。OSS是阿里云提供的海量、安全、低成本、高可靠、高可用的云存储服务。相比阿里云的EBS和NAS服务，OSS可以提供更高的容量以及更低的存储成本，这对于有存放海量数据需求的客户来说是一个很好的选择。OSS默认支持RESTful的对象接口，或者通过Hadoop来访问OSS。

您还可以使用云存储网关通过NFS、SMB、iSCSI来访问OSS，使得传统应用程序可以像使用文件夹或者块设备一样使用OSS。您无需改造现有应用，只需开通和简单配置云存储网关即可。主要有以下几种应用场景：

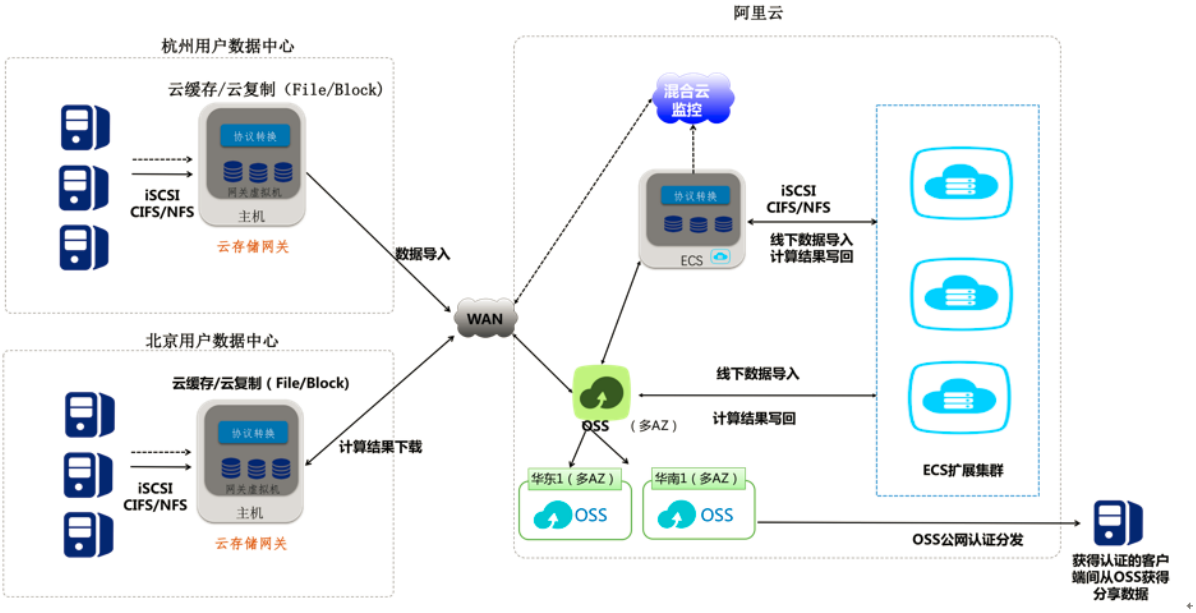
- 共享文件池：在不同计算集群之间共享文件和数据。
- 数据备份：通过类似Veaam、NBU等备份软件，将应用数据按一定的策略通过云存储网关将数据备份到阿里云的OSS存储服务。
- 冷数据归档：通过云存储网关将冷数据从线下或者ECS实例中写入OSS的低频和归档存储类型，从而释放本地空间，并减少存储成本。

跨地域共享和数据分发

云存储网关对接的OSS存储服务可以通过网络实现线下和线上多地访问。通过云存储网关，您可以轻松实现一处写入多地读取的数据共享服务。

在下图的示例场景中，杭州的用户数据中心产生大量数据，但因本地计算能力不足，您可以在阿里云上弹性扩展ECS集群。您在阿里云ECS集群里的应用既可以直接读取网关上传到OSS的对象数据，也可以部署和线下一样的应用程序，通过云存储网关读取OSS共享的数据。

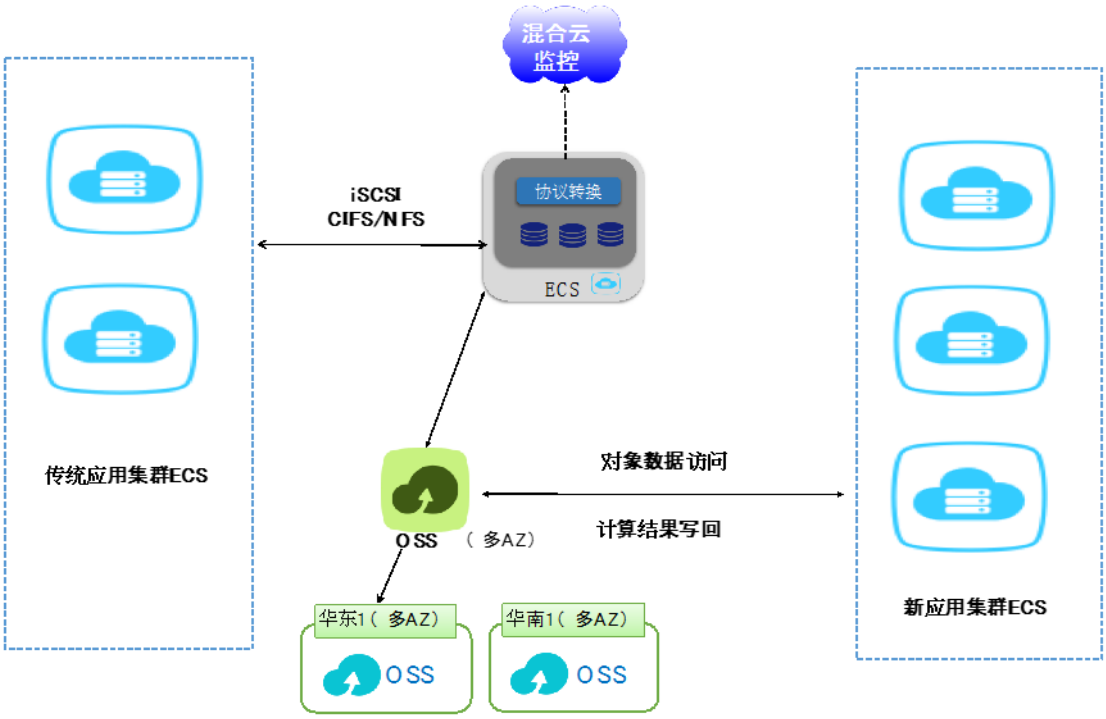
计算任务完成后，最终的结果写回到OSS。如果需要将数据发布到北京，您可以通过存储网关把数据反向同步到北京的数据中心，也可以直接通过OSS的外链功能把数据开放给认证的客户端下载。



适配传统应用

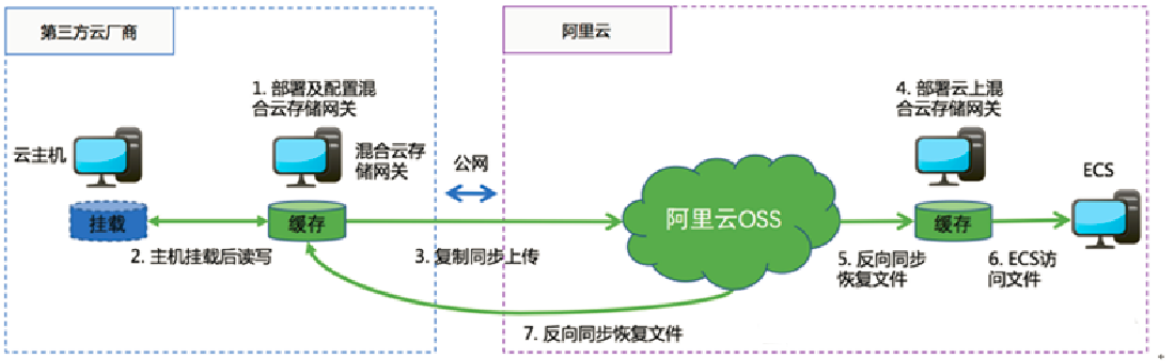
很多云上业务都是新旧业务的组合。其中，从数据中心迁移过来的旧业务使用的是标准的存储协议，如NFS、CIFS和iSCSI；新业务一般采用比较新的技术，支持对象访问的协议。要实现新旧业务之间的数据通讯需要大量的改造工作。而云存储网关恰好可以起到桥梁作用，便捷地进行新旧业务的数据交换。

在下图的示例场景中，传统应用集群可能是文件处理、邮件服务器、图片处理等传统的应用，它们只支持文件或者块访问协议。此时云存储网关的在线版本可以兼容不同的数据访问协议，让这些传统应用集群可以直接访问OSS中的数据。传统应用集群产生的中间数据存储在OSS后，新应用集群可以直接从OSS中读取。反之，新应用集群收集了数据并存放在OSS中，传统应用集群通过云存储网关也可以无缝访问和处理OSS中的数据。



云容灾

随着云计算的普及，越来越多的用户选择将业务部署到云上。为确保数据安全性和业务连续性，跨云容灾提供了高效可靠的本地备份恢复和云上故障转移。借助云存储网关对虚拟化的全面支持，可以轻松应对各种第三方云厂商对接阿里云的数据容灾。整体架构如下图所示。



通过在第三方云厂商导入阿里云的云储存网关（OVA、VHD格式），并配置公网IP，即可搭建阿里云OSS的数据通路。完成数据复制后，部署在阿里云上的ECS即可通过云上部署的阿里云存储网关获取容灾数据，并恢复服务。

注意事项

云存储网关目前还在公测和开发中。使用云储存网关前，您需要了解以下注意事项：

- 云存储网关的HA能力还在开发中。如果有HA需求的业务，可以使用阿里云的其它存储服务。
- 云存储网关目前最大的IOPS依赖于底层的缓存设备能力。公测设备的最大IOPS为5000，带宽为140MByte/s（文件）和900MByte/s（块透写模式）。如果有更高的性能要求，推荐使用阿里云的高性能NAS服务。

4.2 使用指南

4.2.1 简介

OSS阿里云提供的海量、安全、低成本、高可靠的云存储服务。它具有海量存储空间，全局命名和跨地域访问的特性，可以适配多种应用场景。OSS现推出全新的访问形式，即通过集成云存储网关的在线服务，让您可以像使用本地文件夹和磁盘一样使用OSS存储服务。

阿里云的云主机主要运行的操作系统分两大类，Linux和Windows。针对不同的操作系统，云存储网关提供了两种文件访问协议NFS和SMB（CIFS），从而实现本地共享文件夹访问。

此外，云存储网关还提供了iSCSI协议。通过将海量的OSS存储空间映射为本地磁盘，并提供高性价比的存储扩容方案以及超过EBS单盘32TB的容量，适用于对接冷数据和归档数据。

通过云存储网关提供的上述三种协议，OSS存储资源会以Bucket为基础映射成本地文件夹或者磁盘。您可以通过文件读写操作访问OSS资源，无缝衔接基于POSIX和块访问协议的应用，降低应用改造和学习成本。关于云存储网关的具体应用场景，请参见的云存储网关应用场景。

4.2.2 本地共享文件夹访问

针对Linux和Windows两种不同的操作系统，云存储网关提供了两大类文件访问协议NFS和SMB（CIFS）。

- Linux下的NFS协议共享文件夹访问

  NFS协议是Linux/Unix系统下的主流共享文件访问协议。通过简单的挂载、NFS协议共享的OSS存储可以像一个本地文件夹一样进行读写和访问。

  访问存储网关的客户端必须安装NFS，不同的linux操作系统安装方法也不一样。这里介绍ubuntu操作系统和centos操作系统的安装命令，其它操作系统上安装NFS，请查阅官方文档。

- 在ubuntu操作系统上执行以下安装命令：

```
apt-get install nfs-utils
```

- 在centos操作系统上执行以下安装命令：

```
yum install -y nfs-utils
```

命令安装完后后，请执行以下步骤：

1. 创建NFS：

   a. 在云存储网关/**NFS**页面，单击右上角的创建按钮，打开创建**NFS**对话框，如下图所示：

> **说明：**
> 仅需填写带*的必选项即可创建NFS。

   b. 在共享名称框中，填写NFS的虚拟路径（NFS v4）。

c. 在用户映射框中，下拉选择None、root_squash、all_squash、all_anomnymous四种方式中的其中一种。

d. 在模式框中，选择相应的模式。

- 同步模式：所有数据都会保存两份拷贝，一份保存在本地缓存，另一份保存在OSS。
- 缓存模式：本地缓存全量元数据和经常访问的用户数据。OSS保存全量数据。

e. 在云资源框中，单击选择，选择相应的云资源后单击确认。

f. 在缓存硬盘框中，单击选择，选择缓存硬盘后单击确认。

其他更多参数详情，请参见配置*NFS*服务文档。

2. 客户端挂载

在客户端打开命令行终端输入以下挂载命令：

```
mount.nfs x.x.x.x:/shares local-directory
```

其中，

- x.x.x.x:/shares：指的是您的存储网关上的IP地址和共享目录。
- local-directory：指的是客户端的本地目录，可以是任意有读写权限的目录，不能指定不存在的文件目录。

挂载成功后使用命令**df -h**查看系统的磁盘信息。

```
[root@centos7cb ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
/dev/vda1             99G  1.6G   92G   2% /
devtmpfs              24G     0   24G   0% /dev
tmpfs                 24G     0   24G   0% /dev/shm
tmpfs                 24G  424K   24G   1% /run
tmpfs                 24G     0   24G   0% /sys/fs/cgroup
tmpfs                4.8G     0  4.8G   0% /run/user/0
192.168.0.68:/nfs2   256T     0  256T   0% /mnt/nfs172cent7.4
[root@centos7cb ~]#
```

说明：

NFS客户端挂载后容量是云端OSS的容量大小。256TB是文件系统的最大容量，目前OSS的存储空间无大小限制。

挂载后的文件夹就是一个云盘，所以读写的数据都会存放在OSS。NFS共享可以被多个客户端主机挂载以实现NAS服务。更多详情，请参考注意事项文档。

- Windows下的SMB（CIFS）协议共享文件夹访问

SMB（CIFS）协议是Windows系统下的主流共享文件访问协议。通过网络文件夹映射，把OSS存储映射成Windows下的一个网络驱动器，提供类似硬盘的访问。具体操作步骤如下：

1. 创建SMB（CIFS）共享

   a. 在云存储网关**/CIFS**页面，单击右上角的**创建**按钮，打开**创建CIFS**对话框，如下图所示：

   b. 在**共享名称**框中，填写CIFS的网络共享名称。

   c. 在**模式**框中，选择相应的模式。

      - **同步模式**：所有数据都会保存两份拷贝，一份保存在本地缓存，另一份保存在OSS。
      - **缓存模式**：本地缓存全量元数据和经常访问的用户数据。OSS保存全量数据。

   d. **反向同步**：将OSS上的元数据同步到本地，适用于网关容灾和数据恢复、共享场景。

   e. 在**云资源**框中，单击**选择**，选择相应的云资源后单击**确认**。

   f. 在**缓存硬盘**框中，单击**选择**，选择缓存硬盘后单击**确认**。

2. 添加SMB (CIFS) 用户

a. 在云存储网关/CIFS页面，选择CIFS用户一栏。

b. 单击右上角的创建按钮，打开添加CIFS用户对话框，如下图所示：

c. 填写名称、密码后，单击确认。

> 说明：
> 添加CIFS用户时，不支持使用单个英文字母作为用户名称。

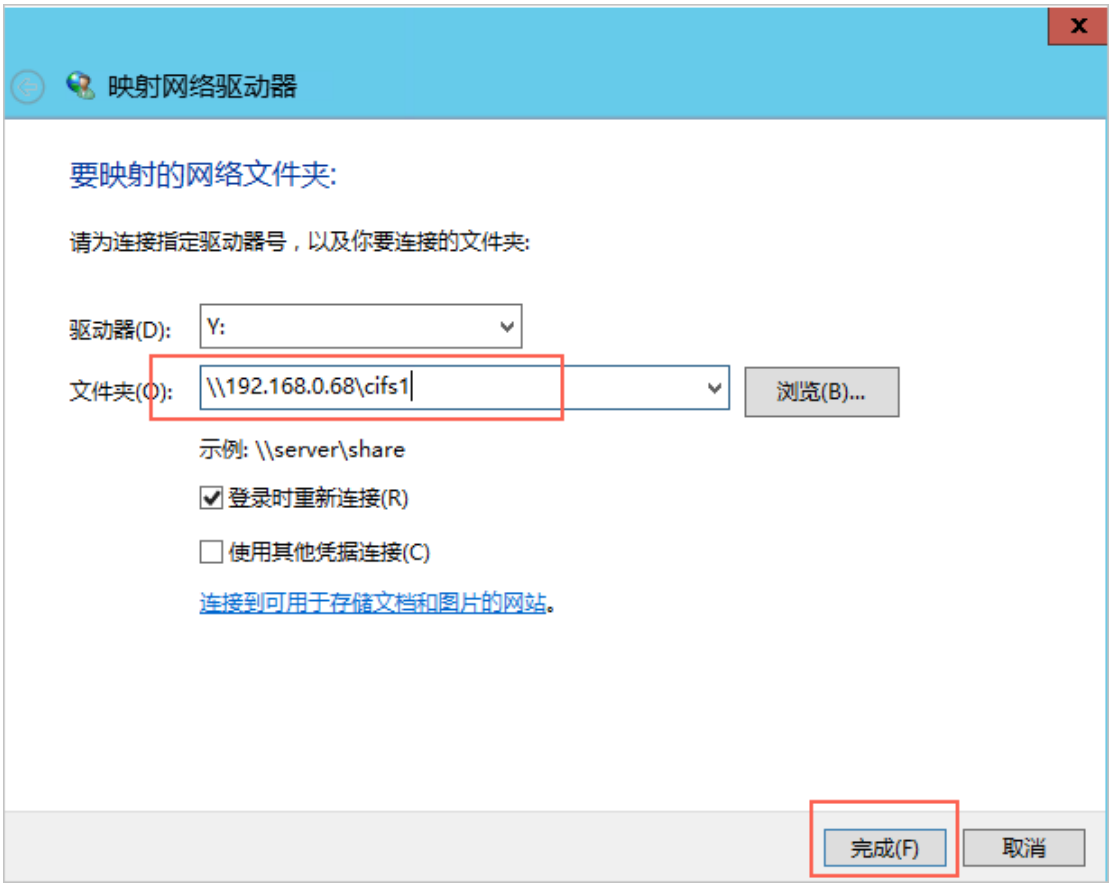
3. 访问共享目录（客户端侧）

客户端使用。indows操作系统访问存储网关，首先需要将存储网关的共享目录添加到本地的映射网络驱动器，添加成功后会在本地的目录和存储网关上的共享目录间建立映射。您通过操作本地的磁盘目录实现对远端OSS存储的操作。

> 说明：
> - CIFS共享最大允许个数为16个。
> - CIFS客户端挂载后容量是云端OSS的容量大小。256TB是文件系统的最大容量，目前OSS的存储空间无大小限制。

具体操作步骤如下：

a. 打开计算机，单击映射网络驱动器，填写文件夹一栏，如下图所示：

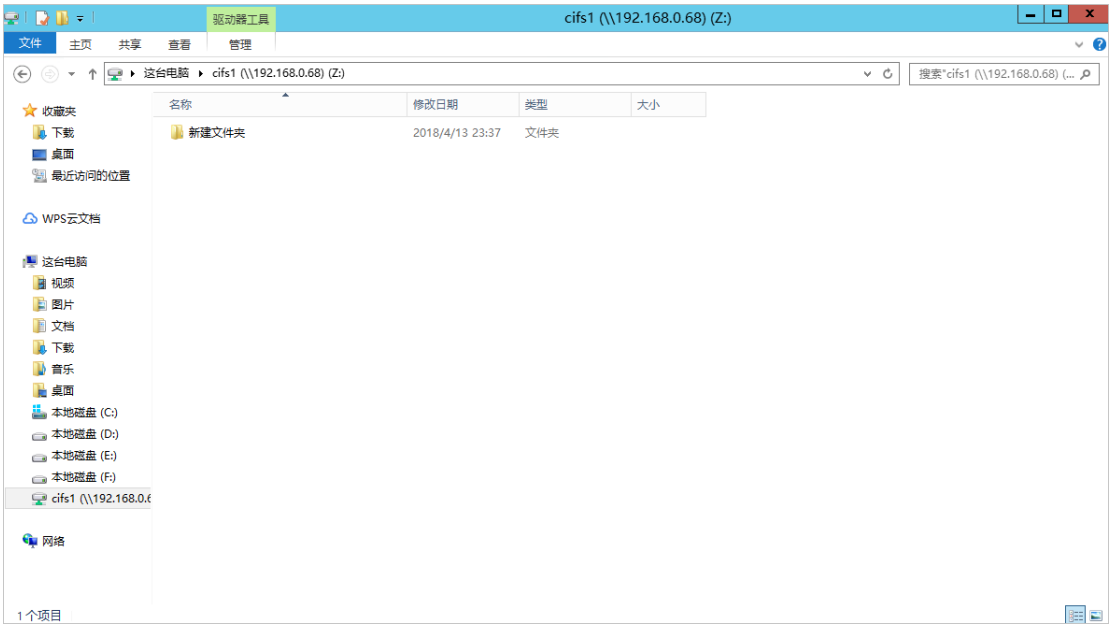


说明：

文件夹一栏填写格式为：\\文件网关与客户端互通的ip\共享目录名。

b. 单击完成，然后输入CIFS用户名密码。

c. Windows客户端添加CIFS共享完成后，在客户端访问该共享网络驱动器情况如下：



4.2.3 本地磁盘访问

云存储网关还提供了iSCSI协议，将海量的OSS存储空间映射为本地磁盘。通过集成云存储网关的在线服务，可以像使用本地磁盘一样使用OSS存储服务。

配置iSCSI Target

通过云存储网关导航列iSCSI卷选项进入块云网关/iSCSI卷界面，您可以在这个界面上创建卷、启用/禁用iSCSI功能、设置CHAP授权、删除逻辑卷等。

创建卷的具体步骤如下：

1. 单击创建，进行卷的创建。

说明：

- 卷最大创建个数：128个。

- iSCSI目标最大允许个数：128个。

2. 填写卷名称、是否启用恢复、容量、选择云资源、是否启用iSCSI、选择模式、缓存磁盘等信息。

说明：

- 在恢复选项框中：
  - 选择否（默认）：系统会直接使用云资源对应的OSS Bucket创建新的卷。
  - 选择是：如果云资源对应的OSS Bucket已经被用作卷的云存储，系统会尝试使用其中的元数据（例如卷的容量等）进行卷的恢复。
- 在模式选择框中：
  - 缓存模式：在缓存模式下，热数据会缓存在本地，读写优先访问本地的缓存盘。通常iSCSI网关的读写性能在缓存模式下更优。
  - 写透模式：在写透模式下，客户文件会透传到云上的OSS Bucket，读取时从云端直接读取，适用于冷数据备份归档场景。

3. 信息填写完成后，单击确认。

卷创建成功后会在导航页显示创建的列表：

在块云网关/iSCSI卷的卷列表中，单击左侧>按钮查看卷属性。

Linux客户端上使用卷

1. 安装软件

请执行以下步骤通过Linux客户端连接到卷：

a. 使用iscsi-initiator-utils RPM 包连接到网关iSCSI目标。

> 说明：
> 使用**sudo yum install iscsi-initiator-utils** 命令安装该包，如果您已完成安装，请跳过此步骤。

b. 使用如下命令验证iSCSI守护进程是否正在运行。

- **sudo /etc/init.d/iscsi status //**：适用于RHEL 5或 RHEL 6。
- **sudo service iscsid status //**：适用于RHEL7。

如果使用上述命令检查未返回running状态，请使用如下命令运行程序：

sudo service iscsid status //

2. 挂载卷

a. 发现卷，访问端口是3260。

格式：

sudo iscsiadm -m discovery -t st －p < GATEWAY_IP >:3260

示例：

iscsiadm -m discovery -t st -p 10.0.100.51:3260

> 说明：
> 其中，10.0.100.51是网关IP ，可通过控制台的关于 > 网络信息 > **IP**信息获取。

b. 挂载卷

> 说明：
> 由于 iSCSI 协议限制，请勿将一个卷挂载到多个客户端上。

请使用如下命令挂载发现的卷。

格式：

iscsiadm --mode node --targetname <TargetName> --portal <GATEWAY_IP> -l

> 说明：

其中 TargetName 替换为需要挂载的卷的 TargetName，GATEWAY_IP 替换为您的网关 IP。

示例：

iscsiadm -m node -T iqn.2009-09.com.aliyun.iscsi-sgw:test97-block95 -p 10.0.100.51:3260 -l

```
[root@localhost ~]# iscsiadm -m discovery -t st -p 10.0.100.51:3260
10.0.100.51:3260,1 iqn.2009-09.com.aliyun.iscsi-sgw:testVolume95-block95
10.0.100.51:3260,1 iqn.2009-09.com.aliyun.iscsi-sgw:test96-block95
10.0.100.51:3260,1 iqn.2009-09.com.aliyun.iscsi-sgw:test961-block95
10.0.100.51:3260,1 iqn.2009-09.com.aliyun.iscsi-sgw:test97-block95
[root@localhost ~]# iscsiadm -m node  -T iqn.2009-09.com.aliyun.iscsi-sgw:test97-block95 -p 10.0.100.51:3260 -l
Logging in to [iface: default, target: iqn.2009-09.com.aliyun.iscsi-sgw:test97-block95, portal: 10.0.100.51,3260] (multiple)
Login to [iface: default, target: iqn.2009-09.com.aliyun.iscsi-sgw:test97-block95, portal: 10.0.100.51,3260] successful.
[root@localhost ~]#
```

c. 查看卷

您可以使用**fdisk -l**、**lsblk**等命令查看已经挂载的卷。当前状态下，卷已经是一个可用的裸磁盘。

使用**fdisk -l**查看如图所示：

```
[root@client2 ~]# fdisk -l

Disk /dev/sda: 214.7 GB, 214748364800 bytes, 419430400 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk label type: dos
Disk identifier: 0x000e1179

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200   419430399   208665600   8e  Linux LVM

Disk /dev/mapper/centos-root: 53.7 GB, 53687091200 bytes, 104857600 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/mapper/centos-swap: 8455 MB, 8455716864 bytes, 16515072 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/mapper/centos-home: 151.5 GB, 151523426304 bytes, 295944192 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/sdd: 1099.5 GB, 1099511627776 bytes, 2147483648 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 4096 bytes / 524288 bytes
```

更多具体配置信息，请参见云存储网关配置*OSS*资源文档。

Windows客户端上使用卷

1. 启动iSCSI发起程序

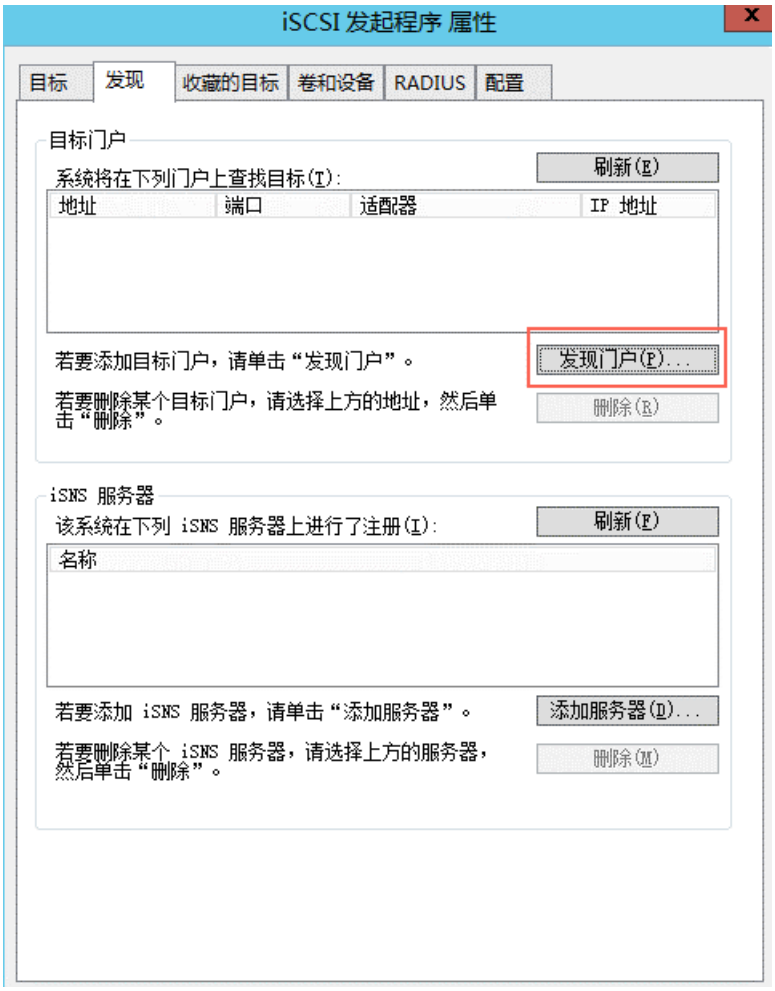
使用 Microsoft Windows 客户端连接到卷，您需要使用 Microsoft Windows iSCSI 启动程序来连接到卷，将卷作为 Windows 客户端上的本地设备。

> **说明：**
> 由于 iSCSI 协议限制，不支持将多个主机连接到同一个 iSCSI 目标。

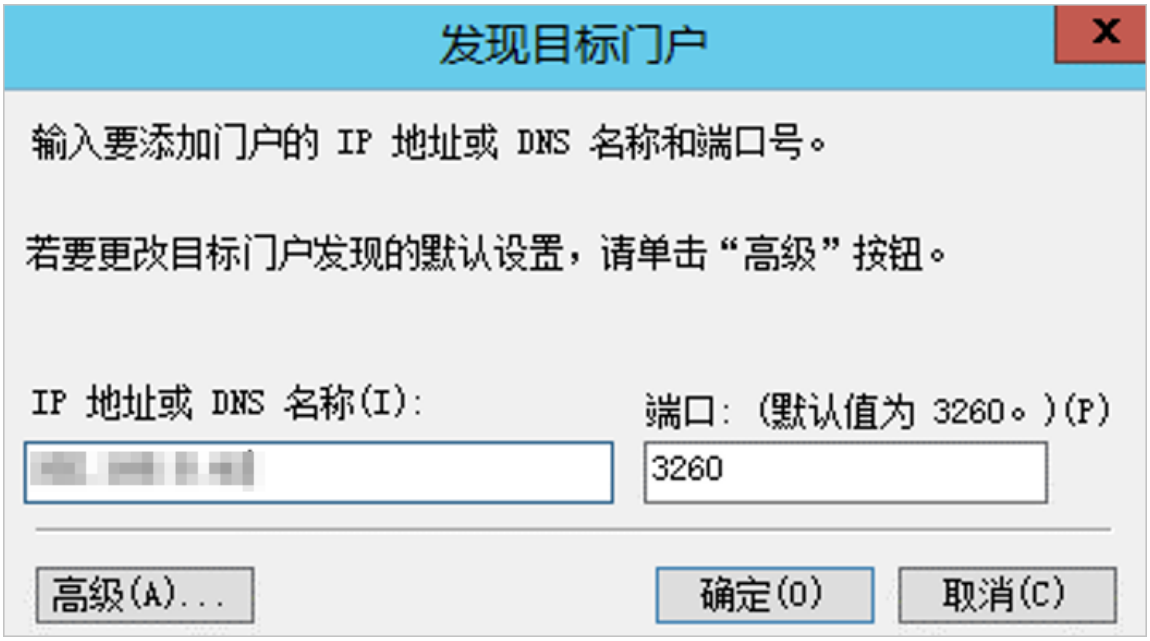
以下用Windows 2012R2为例说明如何启动iSCSI发起程序：

2. 设置iSCSI门户

   1. 在弹出的 **iSCSI** 发起程序 属性对话框中，单击发现。
   2. 在发现选项卡中单击发现门户，如下图所示：



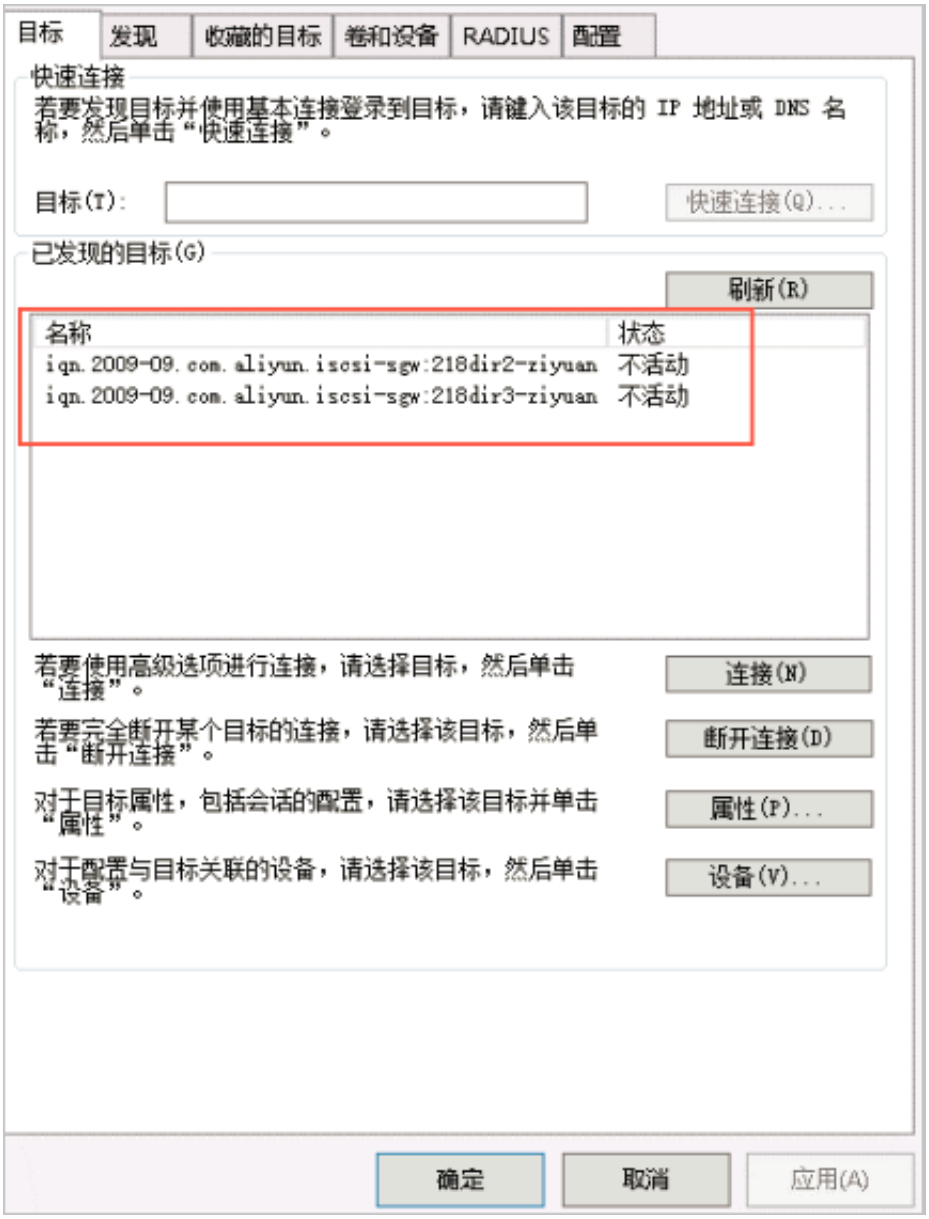
3. 在弹出的窗口中输入目标的 IP 地址，然后单击确认添加该目标门户。



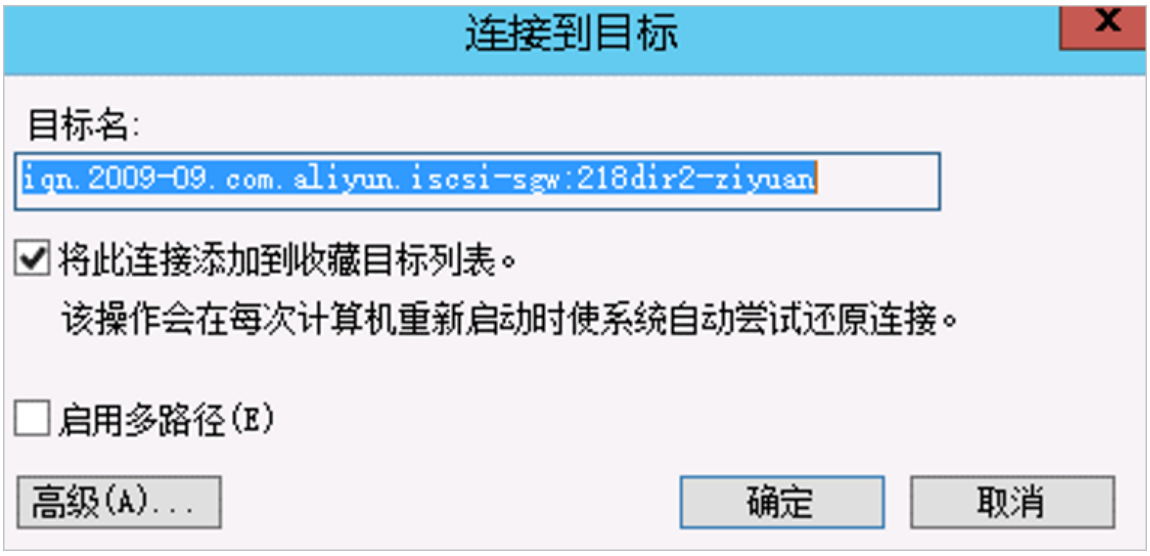
3. iSCSI Target 连接

a. 单击目标。

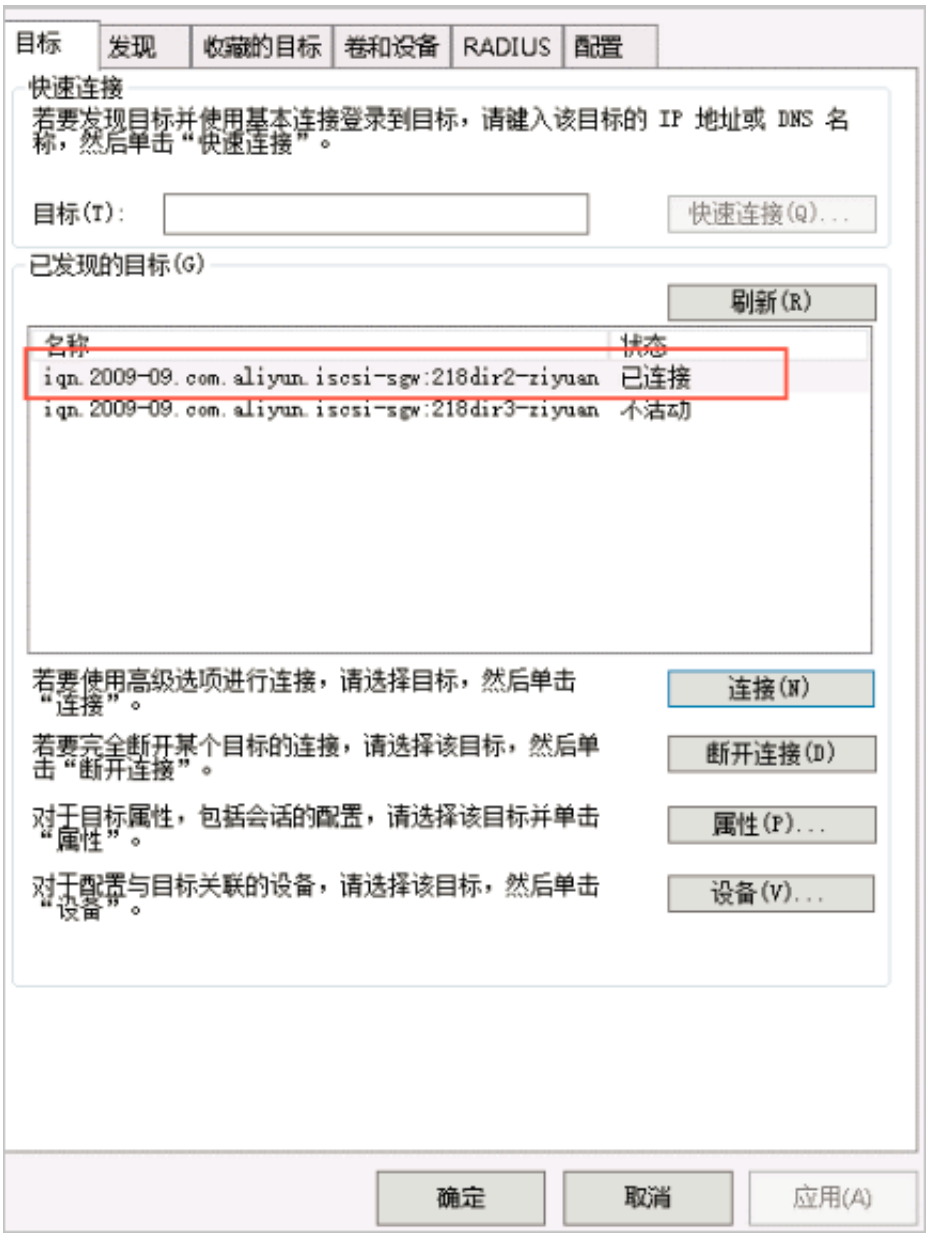
b. 在目标选项卡中，选中上一步骤中处于未激活状态的目标门户，然后单击连接按钮。



c. 在弹出的对话窗中确认目标名，并勾选将此连接添加到收藏目标列表，然后单击确定。

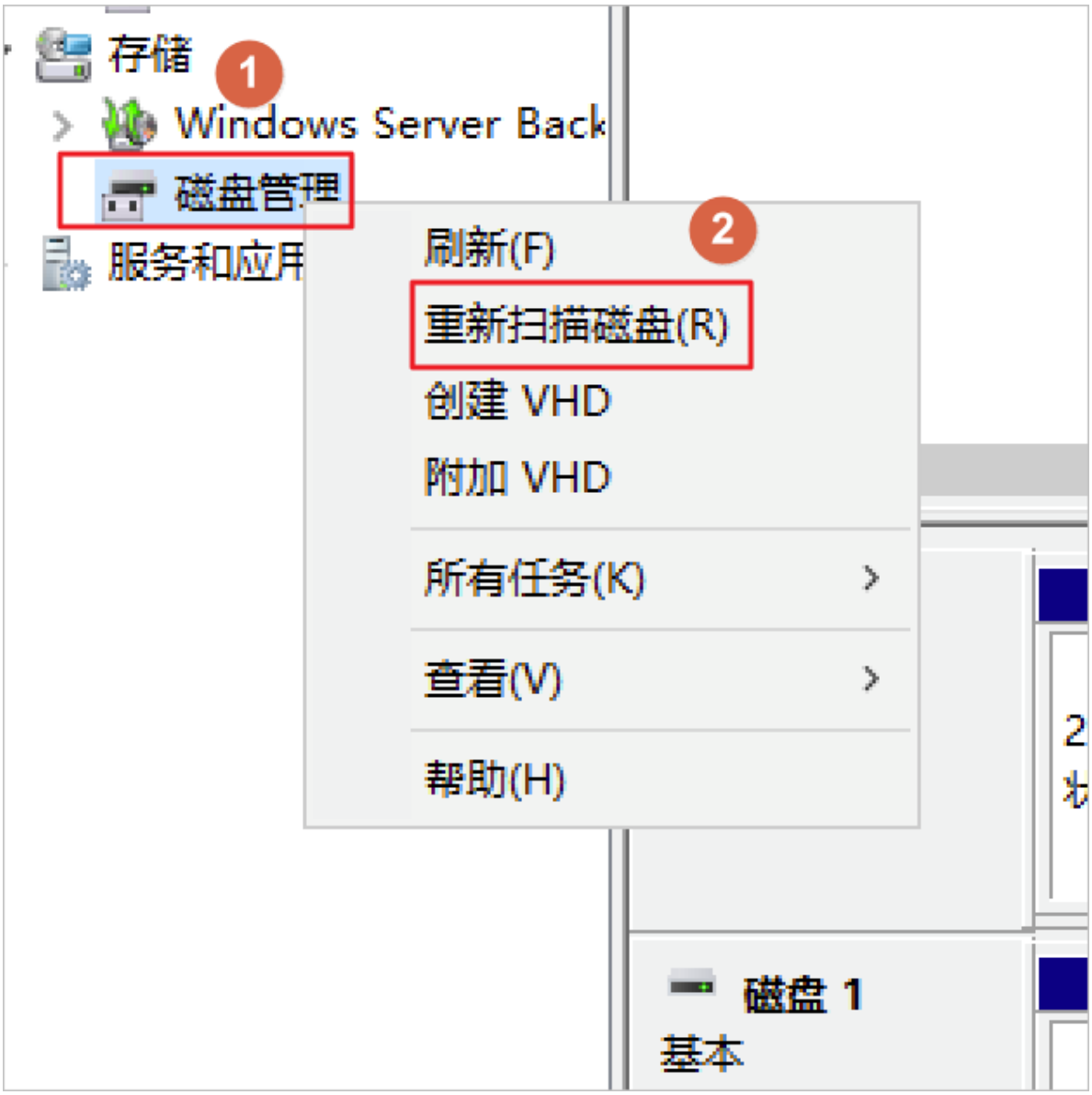


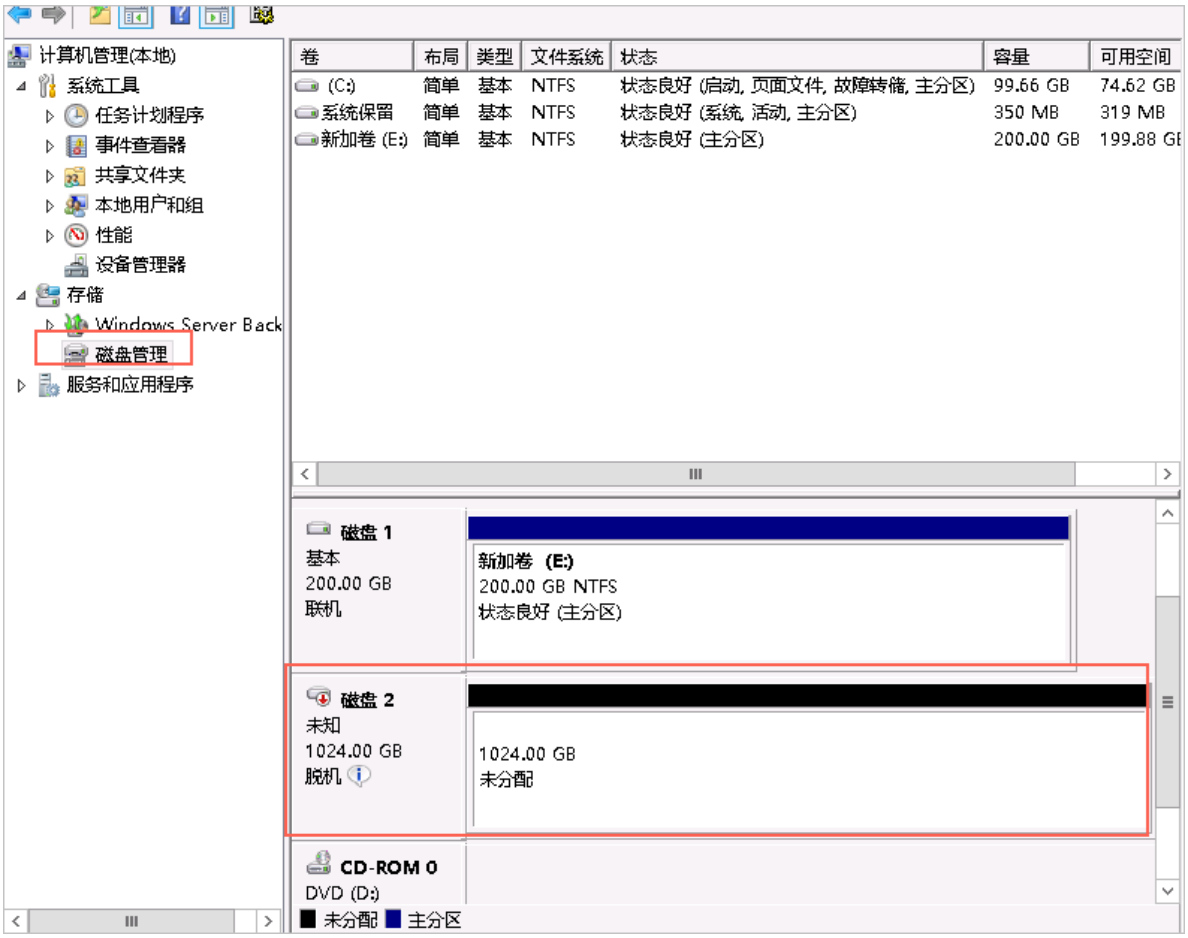
d. 确定状态为已连接后，单击确定按钮。



4. 查看磁盘

   右键单击磁盘管理选项，选择重新扫描磁盘，即能发现新添加的磁盘。


存储
1
Windows Server Back
磁盘管理
服务和应用
2
刷新(F)
重新扫描磁盘(R)
创建 VHD
附加 VHD
所有任务(K)
查看(V)
帮助(H)
磁盘 1
基本


计算机管理(本地)
系统工具
任务计划程序
事件查看器
共享文件夹
本地用户和组
性能
设备管理器
存储
Windows Server Back
磁盘管理
服务和应用程序
卷 布局 类型 文件系统 状态 容量 可用空间
(C:) 简单 基本 NTFS 状态良好 (启动, 页面文件, 故障转储, 主分区) 99.66 GB 74.62 GB
系统保留 简单 基本 NTFS 状态良好 (系统, 活动, 主分区) 350 MB 319 MB
新加卷 (E:) 简单 基本 NTFS 状态良好 (主分区) 200.00 GB 199.88 GB
磁盘 1
基本
200.00 GB
联机
新加卷 (E:)
200.00 GB NTFS
状态良好 (主分区)
磁盘 2
未知
1024.00 GB
脱机
1024.00 GB
未分配
CD-ROM 0
DVD (D:)
未分配
主分区

5 音视频

5.1 短视频

阿里云短视频SDK提供短视频录制、导入、编辑等功能，结合上传SDK、OSS、MTS、CDN及阿里云播放器，可实现短视频的采集、上传、存储、转码、分发、播放的完整功能。

核心优势

- 快速接入，成本经济

  提供产品级SDK，最快2小时接入，节省自行开发耗费的人力物力，帮助您快速实现APP短视频功能。

- 免费人脸检测SDK

  自带由阿里自研的人脸识别技术，快速稳定且高效，助你快速低成本实现视频萌拍效果。

- 接口简单，开放强性

  接口简单易用，开放性强，专业版（UI开源）可以根据业务自由定制UI。

- 功能齐备，应用广泛

  录制功能自带断点录制、实时滤镜、高效美颜、人脸贴图接口功能，支持本地视频导入压缩裁剪，对视频添加MV、动图、字幕、音乐等高级功能。

- 迭代打磨，稳定可靠

  视频技术经钉钉、美柚、梨视频、迅雷、贝贝网、宝宝树、蚂蜂窝等1000多家应用商用验证，稳定可靠。

应用场景

短视频目前已成为APP中的标配功能，SDK适合有短视频UGC场景的APP。

专业版SDK界面

专业版SDK详细功能

功能点	功能说明
自定义UI	SDK包含一套默认的UI，布局、交互、界面可二次开发，基础版支持图标和背景颜色替换，标准版UI完全自定义。
UI开源	提供完整的UI交互源码，用户可定制UI界面。

多段录制	支持断点拍摄和连续拍摄。
自定义时长	自定义最长和最短拍摄时长。
摄像头切换	可选择使用前后摄像头进行录制。
闪光灯	支持打开、关闭、自动闪光灯模式。
实时水印	支持在录制时添加水印。
焦距调节	录制中可调节画面焦距进行放大缩小。
自定义分辨率及质量	可设定拍摄的画面尺寸、比例和质量，基础版仅支持9:16、3:4、1:1三种比例，标准版、专业版支持任意分辨率。
美颜	录制实时美颜，平滑无极调整强度。
实时滤镜	拍摄预览界面实时切换滤镜。
人脸识别	内置免费人脸检测功能，结合贴纸可实现添加实时追踪人脸挂件功能。
第三方人脸SDK	支持第三方人脸检测SDK。
实时混音和变速	支持录制界面添加音乐和变速功能。变速支持调整倍数。
相册选择	支持从相册过滤视频，也支持视频时长过滤。
照片裁剪	支持照片画面大小的裁剪，同时支持画面填充和画面裁剪。
视频裁剪	支持视频画面大小和时长裁剪，同时支持画面填充和画面裁剪。
原比例裁剪	支持保持原始视频比例裁剪视频时长。
单视频导入	支持单视频导入，跳转进入用户定义的页面。
多照片导入	支持多张照片导入，跳转进入用户定义的页面。
多视频导入	支持多视频导入，进入编辑界面。
视频和照片导入	支持多个视频多张照片混合导入，进入编辑界面。
滤镜	在编辑界面添加滤镜，切换滤镜。
动图	在编辑界面添加动图，可在任意时间点添加并支持时间调整。
MV	在编辑界面添加MV效果，切换MV。

音乐	支持将网络音乐和本地音乐合成到视频中。
静音	支持消除当前视频的原音和音乐声音。
字幕	支持普通文字字幕和气泡效果字幕，并且可更换字体。
片尾	支持在视频末尾添加片尾水印效果，可定义持续时间。
涂鸦	支持画笔尺寸和颜色调整。
播放器SDK	提供主流播放功能，支持秒开、边下边播、SEEK、安全播放、倍数播放等。

专业版SDK下载及试用

- 如需试用请发送公司名称、应用名称、申请试用的SDK版本、联系人、联系电话、应用bundleID、包名和签名信息（MD5格式小写无冒号）、阿里云的账号/UID（若没有账号请注册）至*videosdk@service.aliyun.com*，申请开通试用，试用期为一个月。
- 扫码加入SDK咨询钉钉群：

Web端上传视频到OSS

1. 注册阿里云用户，登录阿里云首页，选择对象存储产品，单击**立刻开通**，如下图所示：
2. 新建Bucket，存储类型选择低频访问，读写权限选择私有，如下图所示：
3. 登录*OSS管理控制台*，单击**Access Key**。
4. 在弹出的对话框中，选择开始使用子用户AccessKey，创建新的子用户。
5. 在步骤**选择权限**中，选择AliyunOSSFullAccess。
6. 在OSS管理控制台中，选择**访问控制 > 群组管理 > 新建群组**，如下图所示：
7. 在新建的群组，选择授权，为群组选择AliyunOSSFullAccess权限，如下图所示：
8. 在新建的群组，选择编辑组成员，将刚才创建的子用户设置为组成员，如下图所示：
9. 在*OSS管理控制台*，选择**访问控制 RAM > 角色管理 > 新建角色**。
10. 在步骤**选择角色类型**中，选择用户角色。在步骤**填写类型信息**中，选择当前云账号。
11. 选择新建的角色，选择**角色授权策略 > 编辑授权策略**，为角色选择AliyunOSSFullAccess权限，如下图所示：
12. 返回OSS管理控制台，单击**安全令牌 > 开始授权**，保存AccessKey。详细步骤，请参见设置安全令牌。
13. 参考，编写获取安全令牌的服务。

- 所需的 accessKeyId 及 secretAccessKey，请参见之前保存的AcessKey信息。
- 所需的 RoleArn 及 RoleSessionName，可在OSS管理控制台中单击安全令牌，在显示的配置页面中获取，如下图所示：

14. 通过express搭建node服务器，监听接口、执行请求，并将token返回给客户端，如下图所示：
15. 在本地启动node.js，引入`< script src="http://gosspublic.alicdn.com/aliyun-oss-sdk-4.3.0.min.js">< /script>`，具体代码如下图所示：

- 代码中的 bucket 填写最开始创建的空间名称。
- 代码中的 endpoint，可在OSS管理控制台中选择创建的bucket，在页面**Endpoint**区域获取。

5.2 音视频转码

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。

核心优势

- 高性价比
  - 无需前期投资，只按实际用量付费。
  - 窄带高清和H.265技术，同等视频质量，文件更小，更省流量。
- 强大的转码能力
  - 高速稳定的并行转码系统，按需动态调整转码资源，自动扩容/缩容，应对高并发转码需求无缝扩展集群资源。
- 专业的转码算法

- 强大的计算资源，先进的视频处理算法，业界独有的画质重生技术，将现存普通或受损的影视内容重制为超高清或画质修复的版本。

- 功能丰富、高可定制
  - 视频转码、截图、水印、剪辑、拼接等丰富的媒体转码功能满足各种应用场景。
  - 高可扩展的媒体转码模板，支持自定义转码参数，满足多样化转码需求。
- 易用的媒体工作流
  - 自定义媒体工作流，文件上传完毕自动触发执行媒体工作流转码，消息机制实时状态更新，1分钟搭建常见视频处理流程。

媒体转码详细功能

功能点	说明
输入格式全覆盖	
支持常用输出格式	• 视频：FLV、MP4、M3U8（TS） • 音频：MP3、MP4、OGG、FLAC • 图片：gif、webp
截图	• 支持对存储于OSS上的视频文件截取指定时间的JPG格式图像。 • 支持单张截图、多张截图、平均截图。
水印	• 静态水印：支持在输出的视频上覆盖最多20个静态图像，水印图片支持PNG格式。 • 动态水印：在视频开头和结尾融入动态logo，突出品牌，增加产品识别度。
媒体信息	支持获取存储于OSS上的音、视频文件的编码和内容信息。
转码模版	• 预置模版：媒体转码服务为适配一定网络带宽范围的输出视频预设了一系列转码模版。 • 自定义模版：由用户自行定义转码参数的转码模版，它是转码参数（音频、视频、容器等）的集合，可以满足用户个性化的转码需求。

功能点	说明
倍速转码	适用于30分钟以上的长视频，通过对视频分片并行转码，大幅提升转码速度，转码速度可提升5倍。
窄带高清	阿里云独家窄带高清技术，更低的带宽成本，享受更清晰的视频体验。
画质重生	• 高帧率重制服务（FRC）：对于30帧/秒以内的普通帧率高清节目，生成60帧/秒甚至120帧/秒的高帧率版本，4K大屏播放也无顿挫感。 • 2K转4K重制服务（2K转4K）：对于1080p影片，利用基于海量视频训练的超分辨率技术，生成独家高品质4K节目源。 • 标清转高清重制服务（SD转HD）：对于标清的经典老片，去除胶片颗粒和压缩噪声，加以超分辨率技术，生成720p甚至1080p的高清版本。 • 片源修复服务（PicRescue）：对于被过度压缩的网络视频，去除画面中的毛刺和马赛克，生成更高清晰度的修复重制版。
视频加密	支持“阿里云私有加密”和“HLS-AES128标准加密”两种加密方式。保护视频内容、防下载，适用于在线教育，付费观看等场景。
剪辑输出	支持指定时间点开始，截取指定时长的媒体剪辑。
视频拼接	最多支持20个视频拼接。
分辨率按比例缩放	转码输出参数中仅指定宽或者高，另一个参数留空，则媒体转码服务会按照原视频的宽高比自动设定另一个参数。
M3U8输出自定义切片时长	支持自定义设置M3U8切片时长，范围从1秒至60秒。有助于用户根据播放端带宽条件来设定切片时长，降低用户首屏加载时间。
音视频抽取	从视频文件中单独分离出音频或视频。
视频画面旋转	支持输出视频旋转视频画面一定角度。
视频转GIF	支持视频转码为GIF输出。

功能点	说明
外挂字幕	转码支持导入外部字幕文件并指定字幕编码格式。
媒资	• 支持媒资库：标题、标签、分类、描述等。 • 媒体工作流：云端自动化处理工作流，音视频上传完毕后自动执行处理流程。

操作指南

1. 登录媒体转码控制台。
2. 提交转码作业。
   a. 单击**OSS文件转码管理**。
   b. 单击**新建转码**。
   c. 单击**浏览**选择待转码文件。
   d. 填写**输出文件名**。
   e. 单击**浏览**，选择输出路径，并单击**下一步**。
   f. 在**预置静态模版**中选择一个转码模版，单击**转码**按钮以提交转码作业。
3. 查看转码进度及结果。

   单击**OSS文件转码管理 > OSS文件转码任务列表**，查看转码作业列表、转码进度及转码结果。

   - 查看作业列表及转码进度
   - 查看作业详情

   转码完成后可通过**转码作业详情 > 转码输出**，单击**复制地址**可得到输出文件地址。

更多内容

- 媒体转码产品介绍和文档地址
- 窄带高清2.0 让压缩超越极限，了解详情
- 视频安全解决方案，了解详情

6 数据安全

6.1 通过crc64校验数据传输的完整性

背景

数据在客户端和服务器之间传输时有可能会出错。OSS现在支持对各种方式上传的Object返回其crc64值，客户端可以和本地计算的crc64值做对比，从而完成数据完整性的验证。

- OSS对新上传的Object进行crc64的计算，并将结果存储为Object的元信息存储，随后在返回的response header中增加`x-oss-hash-crc64ecma`头部，表示其crc64值，该64位CRC根据*ECMA-182*标准计算得出。
- 对于crc64上线之前就已经存在于OSS上的Object，OSS不会对其计算crc64值，所以获取此类Object时不会返回其crc64值。

操作说明

- Put Object / Append Object / Post Object / Multipart upload part 均会返回对应的crc64值，客户端可以在上传完成后拿到服务器返回的crc64值和本地计算的数值进行校验。
- Multipart Complete时，如果所有的Part都有crc64值，则会返回整个Object的crc64值；否则，比如有crc64上线之前就已经上传的Part，则不返回crc64值。
- Get Object / Head Object / Get ObjectMeta 都会返回对应的crc64值（如有）。客户端可以在Get Object完成后，拿到服务器返回的crc64值和本地计算的数值进行校验。

  说明：

  range get请求返回的将会是整个Object的crc64值。
- Copy相关的操作，如Copy Object / Upload Part Copy，新生成的Object/Part不保证具有crc64值。

应用示例

以下为完整的Python示例代码，演示如何基于crc64值验证数据传输的完整性。

1. 计算crc64。

```
import oss2
from oss2.models import PartInfo
import os
import crcmod
import random
import string
```

```
do_crc64 = crcmod.mkCrcFun(0x142F0E1EBA9EA3693L, initCrc=0L, xorOut=
0xffffffffffffffffL, rev=True)
def check_crc64(local_crc64, oss_crc64, msg="check crc64"):
if local_crc64 != oss_crc64:
print "{0} check crc64 failed. local:{1}, oss:{2}.".format(msg,
local_crc64, oss_crc64)
return False
else:
print "{0} check crc64 ok.".format(msg)
return True
def random_string(length):
return ''.join(random.choice(string.lowercase) for i in range(length
))
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
```

2. 验证Put Object。

```
content = random_string(1024)
 key = 'normal-key'
 result = bucket.put_object(key, content)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(content))
 check_crc64(local_crc64, oss_crc64, "put object")
```

3. 验证Get Object。

```
result = bucket.get_object(key)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(result.resp.read()))
 check_crc64(local_crc64, oss_crc64, "get object")
```

4. 验证Upload Part 和 Complete。

```
part_info_list = []
 key = "multipart-key"
 result = bucket.init_multipart_upload(key)
 upload_id = result.upload_id
 part_1 = random_string(1024 * 1024)
 result = bucket.upload_part(key, upload_id, 1, part_1)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(part_1))
 #check 上传的 part 1数据是否完整
 check_crc64(local_crc64, oss_crc64, "upload_part object 1")
 part_info_list.append(PartInfo(1, result.etag, len(part_1)))
 part_2 = random_string(1024 * 1024)
 result = bucket.upload_part(key, upload_id, 2, part_2)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(part_2))
 #check 上传的 part 2数据是否完整
 check_crc64(local_crc64, oss_crc64, "upload_part object 2")
 part_info_list.append(PartInfo(2, result.etag, len(part_2)))
 result = bucket.complete_multipart_upload(key, upload_id,
part_info_list)
 oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
 local_crc64 = str(do_crc64(part_2, do_crc64(part_1)))
 #check 最终oss上的object和本地文件是否一致
```

```
check_crc64(local_crc64, oss_crc64, "complete object")
```

OSS SDK支持

部分OSS SDK已经支持上传、下载使用crc64进行数据校验，用法见下表中的示例：

SDK	是否支持CRC	示例
Java SDK	是	*CRCSample.java*
Python SDK	是	*object_check.py*
PHP SDK	否	无
C# SDK	否	无
C SDK	是	*oss_crc_sample.c*
JavaScript SDK	否	无
Go SDK	是	*crc_test.go*
Ruby SDK	否	无
iOS SDK	是	*OSSCrc64Tests.m*
Android SDK	是	*CRC64Test.java*

6.2 通过客户端加密保护数据

客户端加密是指用户数据在发送给远端服务器之前就完成加密，而加密所用的密钥的明文只保留在本地，从而可以保证用户数据安全，即使数据泄漏别人也无法解密得到原始数据。

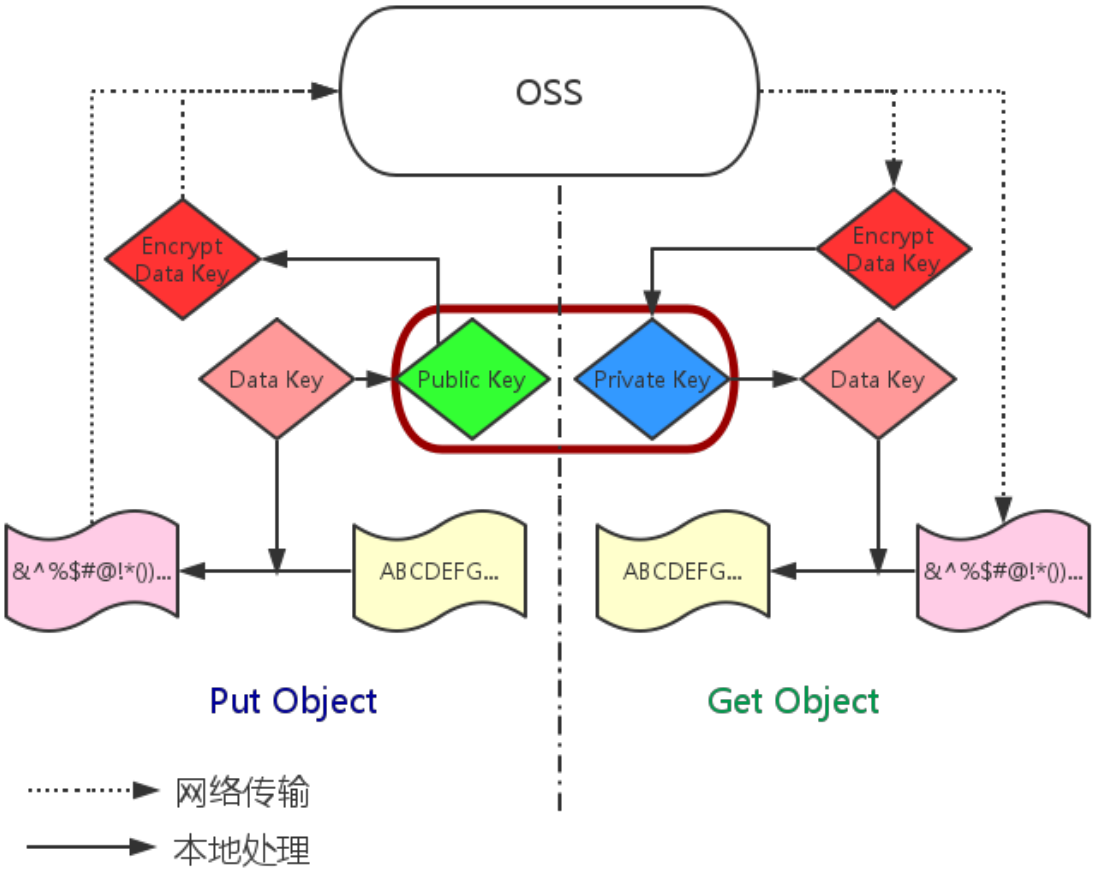
本文介绍如何基于OSS的现有Python SDK版本，通过客户端加密来保护数据。

原理介绍

1. 用户本地维护一对RSA密钥(`rsa_private_key`和`rsa_public_key`)。
2. 每次上传Object时，随机生成一个AES256类型的对称密钥`data_key`，然后用`data_key`加密原始content得到encrypt_content。
3. 用`rsa_public_key`加密`data_key`，得到`encrypt_data_key`，作为用户的自定义meta放入请求头部，和encrypt_content一起发送到OSS。
4. Get Object时，首先得到encrypt_content以及用户自定义meta中的`encrypt_data_key`。
5. 用户使用`rsa_private_key`解密`encrypt_data_key`得到`data_key`，然后用`data_key`解密encrypt_content得到原始content。

> 说明：
> 本文用户的密钥为非对称的RSA密钥，加密Object content时用的AES256-CTR算法，详情可参考 *PyCrypto Document*。本文旨在介绍如何通过Object的自定义meta来实现客户端加密，加密密匙类型及加密算法，用户可以根据自己的需要进行选择。

架构图



准备工作

1. 安装PyCrypto库。

```
pip install pycrypto
```

完整 **Python** 示例代码

```
# -*- coding: utf-8 -*-
import os
import shutil
import base64
import random
```

```
import oss2
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from Crypto import Random
from Crypto.Util import Counter
# aes 256, key always is 32 bytes
_AES_256_KEY_SIZE = 32
_AES_CTR_COUNTER_BITS_LEN = 8 * 16
class AESCipher:
    def __init__(self, key=None, start=None):
        self.key = key
        self.start = start
        if not self.key:
            self.key = Random.new().read(_AES_256_KEY_SIZE)
        if not self.start:
            self.start = random.randint(1, 10)
        ctr = Counter.new(_AES_CTR_COUNTER_BITS_LEN, initial_value=
self.start)
        self.cipher = AES.new(self.key, AES.MODE_CTR, counter=ctr)
    def encrypt(self, raw):
        return self.cipher.encrypt(raw)
    def decrypt(self, enc):
        return self.cipher.decrypt(enc)
# 首先初始化AccessKeyId、AccessKeySecret、Endpoint等信息。
# 通过环境变量获取，或者把诸如“<您的AccessKeyId>”替换成真实的AccessKeyId等。
#
# 以杭州区域为例，Endpoint可以是：
#   http://oss-cn-hangzhou.aliyuncs.com
#   https://oss-cn-hangzhou.aliyuncs.com
# 分别以HTTP、HTTPS协议访问。
access_key_id = os.getenv('OSS_TEST_ACCESS_KEY_ID', '<您的AccessKeyId
>')
access_key_secret = os.getenv('OSS_TEST_ACCESS_KEY_SECRET', '<您的
AccessKeySecret>')
bucket_name = os.getenv('OSS_TEST_BUCKET', '<您的Bucket>')
endpoint = os.getenv('OSS_TEST_ENDPOINT', '<您的访问域名>')
# 确认上面的参数都填写正确了
for param in (access_key_id, access_key_secret, bucket_name, endpoint
):
    assert '<' not in param, '请设置参数：' + param
##### 0 prepare ########
# 0.1 生成rsa key文件并保存到disk
rsa_private_key_obj = RSA.generate(2048)
rsa_public_key_obj = rsa_private_key_obj.publickey()
encrypt_obj = PKCS1_OAEP.new(rsa_public_key_obj)
decrypt_obj = PKCS1_OAEP.new(rsa_private_key_obj)
# save to local disk
file_out = open("private_key.pem", "w")
file_out.write(rsa_private_key_obj.exportKey())
file_out.close()
file_out = open("public_key.pem", "w")
file_out.write(rsa_public_key_obj.exportKey())
file_out.close()
# 0.2 创建Bucket对象，所有Object相关的接口都可以通过Bucket对象来进行
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
obj_name = 'test-sig-1'
content = "test content"
#### 1 Put Object  ####
```

```
# 1.1 生成加密这个object所用的一次性的对称密钥 encrypt_cipher, 其中的key 和
start为随机生成的value
encrypt_cipher = AESCipher()
# 1.2 将辅助解密的信息用公钥加密后存到object的自定义meta中. 后续当我们get
object时，就可以根据自定义meta，用私钥解密得到原始content
headers = {}
headers['x-oss-meta-x-oss-key'] = base64.b64encode(encrypt_obj.encrypt
(encrypt_cipher.key))
headers['x-oss-meta-x-oss-start'] = base64.b64encode(encrypt_obj.
encrypt(str(encrypt_cipher.start)))
# 1.3. 用 encrypt_cipher 对原始content加密得到encrypt_content
encryt_content = encrypt_cipher.encrypt(content)
# 1.4 上传object
result = bucket.put_object(obj_name, encryt_content, headers)
if result.status / 100 != 2:
    exit(1)
#### 2 Get Object ####
# 2.1 下载得到加密后的object
result = bucket.get_object(obj_name)
if result.status / 100 != 2:
    exit(1)
resp = result.resp
download_encrypt_content = resp.read()
# 2.2 从自定义meta中解析出之前加密这个object所用的key 和 start
download_encrypt_key = base64.b64decode(resp.headers.get('x-oss-meta-x
-oss-key', ''))
key = decrypt_obj.decrypt(download_encrypt_key)
download_encrypt_start = base64.b64decode(resp.headers.get('x-oss-meta
-x-oss-start', ''))
start = int(decrypt_obj.decrypt(download_encrypt_start))
# 2.3 生成解密用的cipher, 并解密得到原始content
decrypt_cipher = AESCipher(key, start)
download_content = decrypt_cipher.decrypt(download_encrypt_content)
if download_content != content:
    print "Error!"
else:
    print "Decrypt ok. Content is: %s" % download_content
```

7 OSS资源的监控与报警

云监控服务能够监控OSS服务资源。借助云监控服务，您可以全面了解您在阿里云上的资源使用情况、性能和运行状况。借助报警服务，您可以及时做出反应，保证应用程序顺畅运行。本章介绍如何监控OSS资源、设置报警规则以及自定义监控大盘。

前提条件

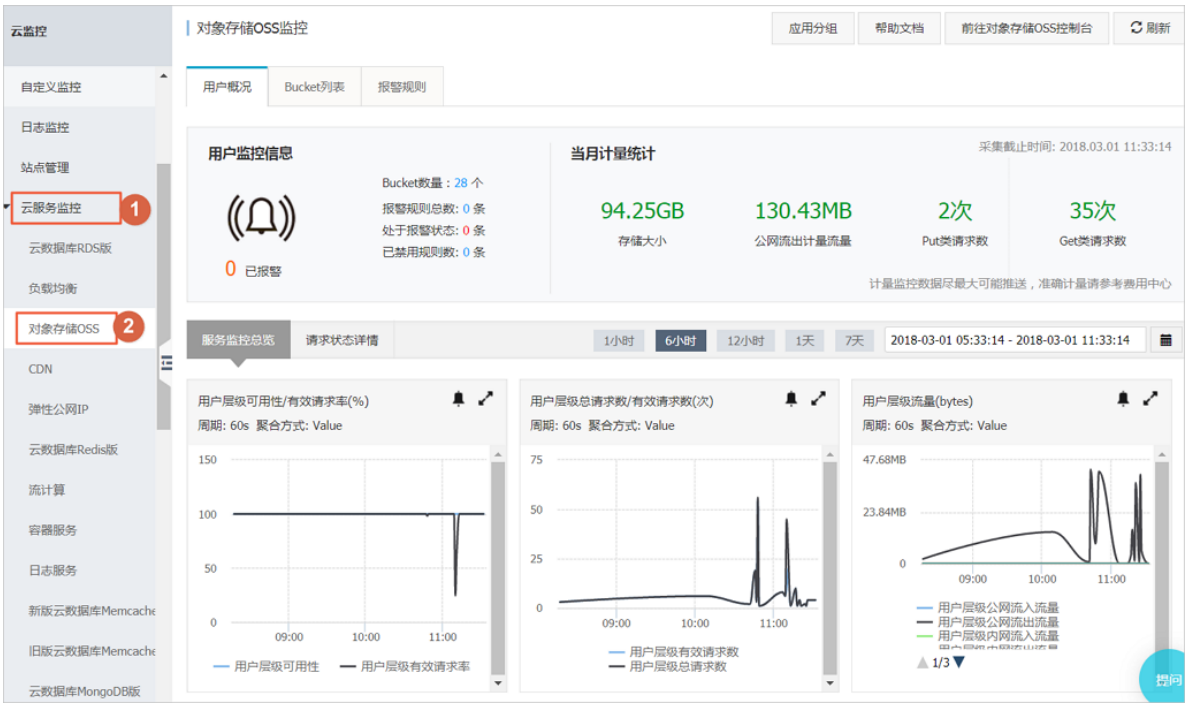
- 已开通OSS服务。
- 已开通云监控服务。

监控OSS资源

1. 登录云监控控制台。
2. 在左侧导航栏选择 **云服务监控 > 对象存储OSS**，进入OSS监控页面，如下图所示。

   在OSS监控页面，您可以获取各类监控数据。

> **说明：**
> 用户层级指用户级别的数据，即该用户下所有bucket的数据。



设置报警规则

1. 在OSS监控页面的报警规则页签下，单击创建报警规则。

2. 填写配置项。

   配置项说明参见管理报警规则。

3. 完成配置后，即生成一条报警规则，您可以使用测试数据来检测该条规则是否生效，确认能否顺利接收报警信息（邮件、短信、旺旺、钉钉等）。

自定义监控大盘

您可以参考如下步骤，在云监控控制台上自定义配置OSS资源监控图。

1. 登录云监控控制台。
2. 在左侧导航栏单击**Dashboard**。
3. 单击创建监控大盘。

4. 输入大盘名称后，单击添加图表。

5. 根据需求完成配置，并单击**发布**。

   配置项说明参见监控指标参考手册。

9 OSS性能与扩展性最佳实践

分区与命名约定

OSS按照文件名UTF-8编码的顺序对用户数据进行自动分区，从而能够处理海量文件，以及承载高速率的客户请求。不过，如果您在上传大量对象时，在命名上使用了顺序前缀（如时间戳或字母顺序），可能会导致大量文件索引集中存储于某个特定分区。这样，当您的请求速率超过2000操作/秒时（下载、上传、删除、拷贝、获取元数据信息等操作算1次操作，批量删除N个文件、列举N个文件等操作算N次操作），会带来如下后果：

- 该分区成为热点分区，导致分区的I/O能力被耗尽，或被系统自动限制请求速率。
- 热点分区的存在会触发系统进行持续的分区数据再均衡，这个过程可能会延长请求处理时间。

从而降低OSS的水平扩展效果，导致客户的请求速率受限。

要解决这个问题，根本上，要消除文件名中的顺序前缀。我们可以在文件名前缀中引入某种随机性，这样文件索引（以及 I/O 负载）就会均匀分布在多个分区。

下面提供了几个将顺序前缀改为随机性前缀的方法案例。

- 示例 1：向文件名添加十六进制哈希前缀

  如下例所示，用户使用了日期与客户ID生成文件名，包含了顺序时间戳前缀：

```
sample-bucket-01/2017-11-11/customer-1/file1
sample-bucket-01/2017-11-11/customer-2/file2
sample-bucket-01/2017-11-11/customer-3/file3
...
sample-bucket-01/2017-11-12/customer-2/file4
sample-bucket-01/2017-11-12/customer-5/file5
sample-bucket-01/2017-11-12/customer-7/file6
...
```

  针对这种情况，我们可以对客户ID计算哈希，即MD5 (customer-id)，并取若干字符的哈希前缀作为文件名的前缀。假如取4个字符的哈希前缀，结果如下所示：

```
sample-bucket-01/2c99/2017-11-11/customer-1/file1
sample-bucket-01/7a01/2017-11-11/customer-2/file2
sample-bucket-01/1dbd/2017-11-11/customer-3/file3
...
sample-bucket-01/7a01/2017-11-12/customer-2/file4
sample-bucket-01/b1fc/2017-11-12/customer-5/file5
sample-bucket-01/2bb7/2017-11-12/customer-7/file6
...
```

  加入4个字符组成的十六进制哈希作为前缀，每个字符有0-f共16种取值，因此4个字符共有16^4=65536种可能的字符组合。那么在存储系统中，这些数据理论上会被持续划分至最多65536个

分区，以每个分区2000操作/秒的性能瓶颈标准，再结合您的业务的请求速率，以此您可以评估hash桶的个数是否合适。

如果您想要列出文件名中带有特定日期的文件，例如列出sample-bucket-01里带有2017-11-11的文件，您只要对sample-bucket-01进行列举（即通过多次调用List Object接口，分批次地获得sample-bucket-01下的所有文件），然后合并带有该日期的文件即可。

- 示例 2：反转文件名

  如下例所示，用户使用了毫秒精度的UNIX时间戳生成文件名，同样属于顺序前缀：

```
sample-bucket-02/1513160001245.log
sample-bucket-02/1513160001722.log
sample-bucket-02/1513160001836.log
sample-bucket-02/1513160001956.log
...
sample-bucket-02/1513160002153.log
sample-bucket-02/1513160002556.log
sample-bucket-02/1513160002859.log
...
```

  由前面的分析，我们知道，这种顺序前缀命名，在请求速率超过一定阈值时，会引发性能问题。我们可以通过反转时间戳前缀来避免，这样文件名就不包含顺序前缀了。反转后结果如下：

```
sample-bucket-02/5421000613151.log
sample-bucket-02/2271000613151.log
sample-bucket-02/6381000613151.log
sample-bucket-02/6591000613151.log
...
sample-bucket-02/3512000613151.log
sample-bucket-02/6552000613151.log
sample-bucket-02/9582000613151.log
...
```

  由于文件名中的前3位数字代表毫秒时间，会有1000种取值。而第4位数字，每1秒钟就会改变一次。同理第5位数字每10秒钟就会改变一次…以此类推，反转文件名后，极大地增强了前缀的随机性，从而将负载压力均匀地分摊在各个分区上，避免出现性能瓶颈。