

Alibaba Cloud Object Storage Service

開發指南

檔案版本：20190807

法律聲明

阿里雲提醒您在閱讀或使用本文檔之前仔細閱讀、充分理解本法律聲明各條款的內容。如果您閱讀或使用本文檔，您的閱讀或使用行為將被視為對本聲明全部內容的認可。

1. 您應當通過阿里雲網站或阿里雲提供的其他授權通道下載、擷取本文檔，且僅能用於自身的合法合規的商務活動。本文檔的內容視為阿里雲的保密資訊，您應當嚴格遵守保密義務；未經阿里雲事先書面同意，您不得向任何第三方披露本手冊內容或提供給任何第三方使用。
2. 未經阿里雲事先書面許可，任何單位、公司或個人不得擅自摘抄、翻譯、複製本文檔內容的部分或全部，不得以任何方式或途徑進行傳播和宣傳。
3. 由於產品版本升級、調整或其他原因，本文檔內容有可能變更。阿里雲保留在沒有任何通知或者提示下對本文檔的內容進行修改的權利，並在阿里雲授權通道中不時發布更新後的使用者文檔。您應當即時關注使用者文檔的版本變更並通過阿里雲授權渠道下載、擷取最新版的使用者文檔。
4. 本文檔僅作為使用者使用阿里雲產品及服務的參考性指引，阿里雲以產品及服務的”現狀“、“有缺陷”和“當前功能”的狀態提供本文檔。阿里雲在現有技術的基礎上盡最大努力提供相應的介紹及操作指引，但阿里雲在此明確聲明對本文檔內容的準確性、完整性、適用性、可靠性等不作任何明示或暗示的保證。任何單位、公司或個人因為下載、使用或信賴本文檔而發生任何差錯或經濟損失的，阿里雲不承擔任何法律責任。在任何情況下，阿里雲均不對任何間接性、後果性、懲戒性、偶然性、特殊性或刑罰性的損害，包括使用者使用或信賴本文檔而遭受的利潤損失，承擔責任（即使阿里雲已被告知該等損失的可能性）。
5. 阿里雲網站上所有內容，包括但不限於著作、產品、圖片、檔案、資訊、資料、網站架構、網站畫面的安排、網頁設計，均由阿里雲和/或其關係企業依法擁有其智慧財產權，包括但不限於商標權、專利權、著作權、商業秘密等。非經阿里雲和/或其關係企業書面同意，任何人不得擅自使用、修改、複製、公開傳播、改變、散布、發行或公開發表阿里雲網站、產品程式或內容。此外，未經阿里雲事先書面同意，任何人不得為了任何營銷、廣告、促銷或其他目的使用、公布或複製阿里雲的名稱（包括但不限於單獨為或以組合形式包含”阿里雲”、Aliyun”、“萬網”等阿里雲和/或其關係企業品牌，上述品牌的附屬標誌及圖案或任何類似公司名稱、商號、商標、產品或服務名稱、網域名稱、圖案標示、標誌、標識或通過特定描述使第三方能夠識別阿里雲和/或其關係企業）。
6. 如若發現本文檔存在任何錯誤，請與阿里雲取得直接聯絡。

通用約定

格式	說明	範例
	該類警示資訊將導致系統重大變更甚至故障，或者導致人身傷害等結果。	 禁止： 重設操作將丟失使用者配置資料。
	該類警示資訊可能導致系統重大變更甚至故障，或者導致人身傷害等結果。	 警告： 重啟操作將導致業務中斷，恢復業務所需時間約10分鐘。
	用於補充說明、最佳實務、竅門等，不是使用者必須瞭解的內容。	 說明： 您也可以通過按Ctrl + A選中全部檔案。
>	多級菜單遞進。	設定 > 網路 > 設定網路類型
粗體	表示按鍵、菜單、頁面名稱等UI元素。	單擊 確定。
<code>courier</code> 字型	命令。	執行 <code>cd / d C :/ windows</code> 命令，進入Windows系統檔案夾。
##	表示參數、變數。	<code>bae log list --instanceid Instance_ID</code>
[]或者[a b]	表示可選項，至多選擇一個。	<code>ipconfig [-all -t]</code>
{ }或者{a b}	表示必選項，至多選擇一個。	<code>swich {stand slave}</code>

目錄

法律聲明.....	I
通用約定.....	I
1 使用前須知.....	1
2 基本概念介紹.....	2
3 儲存類型.....	7
3.1 儲存類型介紹.....	7
3.2 建立和使用Archive Storage類型.....	10
4 接入OSS.....	16
4.1 快速開始.....	16
4.2 基於OSS的移動開發.....	16
5 管理儲存空間.....	21
5.1 建立儲存空間.....	21
5.2 設定儲存空間讀寫權限（ACL）.....	21
5.3 獲取儲存空間資訊.....	22
5.4 查看儲存空間列表.....	22
5.5 綁定自訂網域名.....	23
5.6 設定防盜鏈.....	24
5.7 設定跨域訪問.....	25
5.8 刪除儲存空間.....	26
6 上傳檔案.....	27
6.1 簡單上傳.....	27
6.2 表單上傳.....	28
6.3 斷點續傳.....	32
6.4 追加上傳.....	35
6.5 授權給第三方上傳.....	37
6.6 上傳回調.....	38
6.7 RTMP推流上傳.....	39
7 下載檔案.....	43
7.1 簡單下載.....	43
7.2 斷點續傳下載.....	44
7.3 授權給第三方下載.....	44
8 管理檔案.....	47
8.1 設定對象/檔案元資訊.....	47
8.2 查看對象列表.....	48
8.3 拷貝對象.....	51
8.4 刪除對象.....	52
8.5 管理回源設定.....	52
8.6 SelectObject.....	56

9 訪問與控制	71
9.1 存取控制	71
9.2 基於RAM Policy的許可權控制	82
9.2.1 教程樣本：使用RAM Policy控制儲存空間和檔案夾的存取權限	82
9.2.2 如何構建RAM Policy	103
9.3 基於Bucket Policy的許可權控制	112
9.4 跨帳號授權	113
9.4.1 跨帳號授權概述	113
9.4.2 教程樣本：基於Bucket Policy實現跨帳號訪問OSS	113
10 安全管理	115
10.1 伺服器端加密編碼	115
11 靜態網站託管	119
11.1 配置靜態網站託管	119
11.2 教程樣本：使用自訂網域名設定靜態網站託管	120
12 OSS沙箱	127
13 監控服務	131
13.1 監控服務概覽	131
13.2 使用監控服務	132
13.3 使用報警服務	146
13.4 訪問監控數據	151
13.5 監控指標參考	158
13.6 監控、診斷和故障排除	165
14 雲端資料處理	180

1 使用前須知

如果您初次使用阿里雲OSS，請參閱阿里雲[OSS快速入門系列文檔](#)，幫助您了解OSS並快速使用OSS。

如果您已經充分了解OSS，您也可以通過下列資源快速使用OSS的其他各項功能：

資源	描述
阿里雲OSS開發人員指南	本文檔為您講解阿里雲OSS服務的核心概念、所有功能介紹與操作步驟，以及如何使用API和SDK的有效樣本。
阿里雲OSS最佳實踐	詳細介紹阿里雲OSS的各種使用場景與配置實踐。
阿里雲OSS SDK參考	介紹主流語言的SDK開發操作和參數。
阿里雲OSS API參考	詳細探討了阿里雲OSS支援的RESTful API操作和相關的樣本。
阿里雲OSS 控制台用戶指南	阿里雲OSS管理主控台可讓您通過介面執行OSS的部分功能。本文檔為您介紹了基於阿里雲OSS管理主控台的所有操作。
阿里雲OSS圖片處理指南	詳細探討了阿里雲OSS提供的圖片處理服務的詳細內容與操作方式。
阿里雲OSS遷移工具	您可能需要將您本地或第三方雲端儲存體服務上的檔案同步到阿里雲OSS上，阿里雲為此提供了完善的遷移解決方案。

2 基本概念介紹

本部分將向您介紹本產品中涉及的幾個基本概念，以便於您更好地理解Object Storage Service 產品。

儲存空間（Bucket）

儲存空間是您用於儲存物件（Object）的容器，所有的對象都必須隸屬於某個儲存空間。您可以設定和修改儲存空間屬性用來控制地域、存取權限、生命週期等，這些屬性設定直接作用於該儲存空間內所有對象，因此您可以通過靈活建立不同的儲存空間來完成不同的管理功能。

- 同一個儲存空間的內部是扁平的，沒有檔案系統的目錄等概念，所有的對象都直接隸屬於其對應的儲存空間。
- 每個使用者可以擁有多個儲存空間。
- 儲存空間的名稱在 OSS 範圍內必須是全域唯一的，一旦建立之後無法修改名稱。
- 儲存空間內部的對象數目沒有限制。

儲存空間的命名規範如下：

- 只能包括小寫字母、數字和短橫線（-）。
- 必須以小寫字母或者數字開頭和結尾。
- 長度必須在3-63位元組之間。

對象/檔案（Object）

對象是 OSS 儲存資料的基本單元，也被稱為 OSS 的檔案。對象由元資訊（Object Meta），使用者資料（Data）和檔案名（Key）組成。對象由儲存空間內部唯一的 Key 來標識。對象元資訊是一個鍵值對，表示了對象的一些屬性，比如最後修改時間、大小等資訊，同時使用者也可以在元資訊中儲存一些自訂的資訊。

對象的生命週期是從上傳成功到被刪除為止。在整個生命週期內，對象資訊不可變更。重複上傳同名的對象會覆蓋之前的對象，因此，OSS 不支援修改檔案的部分內容等操作。

OSS 提供了追加上傳功能，使用者可以使用該功能不斷地在Object尾部追加寫入資料。

對象的命名規範如下：

- 使用UTF-8編碼。
- 長度必須在1-1023位元組之間。

- 不能以正斜線 (/) 或者反斜線 (\) 開頭。



说明:

對象名稱需要區分大小寫。如無特殊說明，本文檔中的對象、檔案稱謂等同於 Object。

Region（地域）

Region 表示 OSS 的資料中心所在的地域，物理位置。使用者可以根據費用、請求來源等綜合選擇資料存放區的 Region。一般來說，距離使用者更近的 Region 訪問速度更快。詳情請查看[OSS 已經開通的 Region](#)。

Region 是在建立 Bucket 的時候指定的，一旦指定之後就不允許更改。該 Bucket 下所有的 Object 都儲存在對應的資料中心，目前不支援 Object 等級的 Region 設定。

Endpoint（訪問網域名稱）

Endpoint 表示 OSS 對外服務的訪問網域名稱。OSS 以 HTTP RESTful API 的形式對外提供服務，當訪問不同的 Region 的時候，需要不同的網域名稱。通過內網和外網訪問同一個 Region 所需要的 Endpoint 也是不同的。例如杭州 Region 的外網 Endpoint 是 `oss-cn-hangzhou.aliyuncs.com`，內網 Endpoint 是 `oss-cn-hangzhou-internal.aliyuncs.com`。具體的內容請參見[各個 Region 對應的 Endpoint](#)。

AccessKey（存取金鑰）

AccessKey，簡稱 AK，指的是訪問身分識別驗證中用到的 AccessKeyId 和 AccessKeySecret。OSS 通過使用 AccessKeyId 和 AccessKeySecret 對稱式加密的方法來驗證某個請求的寄件者身份。AccessKeyId 用於標識使用者，AccessKeySecret 是使用者用於加密簽名字元串和 OSS 用來驗證簽名字元串的密鑰，其中 AccessKeySecret 必須保密。對於 OSS 來說，AccessKey 的來源有：

- Bucket 的擁有者申請的 AccessKey。
- 被 Bucket 的擁有者通過 RAM 授權給第三方要求者的 AccessKey。
- 被 Bucket 的擁有者通過 STS 授權給第三方要求者的 AccessKey。

更多 AccessKey 介紹請參見[存取控制](#)。

強一致性

Object 操作在 OSS 上具有原子性，操作要麼成功要麼失敗，不會存在有中間狀態的 Object。OSS 保證使用者一旦上傳完成之後讀到的 Object 是完整的，OSS 不會返回給使用者一個部分上傳成功的 Object。

Object 操作在 OSS 上同樣具有強一致性，使用者一旦收到了一個上傳（PUT）成功的響應，該上傳的 Object 就已經立即可讀，並且資料的三份副本已經寫成功。不存在一種上傳的中間狀態，即 read-after-write 卻無法讀取到資料。對於刪除操作也是一樣的，使用者刪除指定的 Object 成功之後，該 Object 立即變為不存在。

強一致性方便了使用者架構設計，可以使用跟傳統存放裝置同樣的邏輯來使用 OSS，修改立即可見，無需考慮最終一致性帶來的各種問題。

OSS與檔案系統的對比

對比項	OSS	檔案系統
資料模型	OSS 是一個分布式的 Object Storage Service 服務，提供的是一個 Key-Value 對形式的 Object Storage Service 服務。	檔案系統是一種典型的樹狀索引結構。
資料獲取	根據 Object 的名稱（Key）唯一的獲取該 Object 的內容。 雖然使用者可以使用類似 test1/test.jpg 的名字，但是這並不表示使用者的 Object 是保存在 test1 目錄下面的。對於 OSS 來說，test1/test.jpg 僅僅只是一個字元串，和 a.jpg 這種並沒有本質的區別。因此不同名稱的 Object 之間的訪問消耗的資源是類似的。	一個名為 test1/test.jpg 的檔案，訪問過程需要先訪問到 test1 這個目錄，然後再在該目錄下尋找名為 test.jpg 的檔案。
優勢	支援海量的使用者並發訪問。	支援檔案的修改，比如修改指定位移位的內容、截斷檔案尾部等。也支援檔案夾的操作，比如重新命名目錄、刪除目錄、移動目錄等非常容易。

對比項	OSS	檔案系統
劣勢	OSS 保存的 Object 不支援修改（追加寫 Object 需要調用特定的介面，生成的 Object 也和正常上傳的 Object 類型上有差別）。使用者哪怕是僅僅需要修改一個位元組也需要重新上傳整個 Object。 OSS可以通過一些操作來類比類似檔案夾的功能，但是代價非常昂貴。比如重新命名目錄，希望將 test1 目錄重新命名成 test2，那麼 OSS 的實際操作是將所有以 test1/ 開頭的 Object 都重新複製成以 test2/ 開頭的 Object，這是一個非常消耗資源的操作。因此在使用 OSS 的時候要盡量避免類似的操作。	受限於單個裝置的效能。訪問越深的目錄消耗的資源也越大，操作擁有很多檔案的目錄也會非常慢。

因此，將 OSS 映射為檔案系統是非常低效的，也是不建議的做法。如果一定要掛載成檔案系統的話，建議盡量只做寫新檔案、刪除檔案、讀取檔案這幾種操作。使用 OSS 應該充分發揮其優點，即海量資料處理能力，優先用來儲存海量的非結構化資料，比如圖片、視頻、文檔等。

以下是OSS與檔案系統的概念對比：

Object Storage Service	檔案系統
Object	檔案
Bucket	主目錄
Region	無
Endpoint	無
AccessKey	無
無	多級目錄
GetService	獲取主目錄列表
GetBucket	獲取檔案清單
PutObject	寫檔案

Object Storage Service	檔案系統
AppendObject	追加寫檔案
GetObject	讀檔案
DeleteObject	刪除檔案
無	修改檔案內容
CopyObject（目的和源相同）	修改檔案屬性
CopyObject	複製檔案
無	重新命名檔案

3 儲存類型

3.1 儲存類型介紹

OSS提供標準、低頻訪問、歸檔三種儲存類型，全面覆蓋從熱到冷的各種資料存放區場景。

標準儲存類型（Standard）

OSS標準儲存類型提供高可靠、高可用、高性能的對象儲存服務，能夠支援頻繁的數據訪問。OSS高吞吐和低延時的服務響應能力能夠有效支援各種熱點類型數據的訪問。標準儲存類型是各種社交、分享類的圖片、音視頻應用、大型網站、大資料分析的合適選擇。

關鍵特性：

- 99.999999999%的數據可靠性
- 按照99.99%服務可用性設計
- 低延時、高吞吐的訪問性能
- 支援HTTPS加密傳輸
- 支援圖片處理

低頻訪問儲存類型（Infrequent Access）

OSS低頻訪問儲存類型適合長期保存不經常訪問的數據（平均每月訪問頻率1到2次）。儲存單價低於標準類型，適合各類行動裝置 App、智慧型裝置、企業數據的長期備份，支援即時數據訪問。低頻訪問儲存類型的Object有最短儲存時間，儲存時間短於30天的Object提前被刪除會產生一定費用。低頻訪問儲存Object有最小計量空間，Object大小低於64KB，會按照64KB計算儲存空間，數據獲取會產生費用。

關鍵特性：

- 99.999999999%的數據可靠性
- 按照99.99%服務可用性設計
- 支援即時訪問
- 支援HTTPS加密傳輸
- 支援圖片處理
- 有最短儲存時間和最小計量空間

Archive Storage類型 (Archive)

OSS Archive Storage類型在三種儲存類型中單價最低，適合需要長期保存（建議半年以上）的歸檔數據，在儲存周期內極少被訪問，數據進入到可讀取狀態需要1分鐘的解凍時間。適合需要長期保存的檔案數據、醫學影像、科學資料、影視素材。Archive Storage類型的Object有最短儲存時間，儲存時間短於60天的Object提前刪除會產生一定費用。歸檔類型儲存Object有最小計量空間，Object大小低於64KB，會按照64KB計算儲存空間，數據獲取會產生費用。

關鍵特性：

- 99.999999999%的數據可靠性
- 按照99.99%服務可用性設計
- 已經儲存的數據從冷凍狀態恢復到可讀取狀態需要1分鐘的等待時間
- 支援HTTPS加密傳輸
- 支援圖片處理，但需要先解凍
- 有最短儲存時間和最小計量空間

儲存類型對比

對比指標	標準儲存類型	低頻訪問儲存類型	Archive Storage類型
數據可靠性	99.999999999%	99.999999999%	99.999999999%
服務設計的可用性	99.99%	99.99%	99.99% (數據解凍之後)
對象最小計量大小	按照對象實際大小計算	64KB	64KB
最少儲存時間	無最短儲存實際要求	30天	60天
數據取回費用	不收取數據取回費用	按實際獲取的數據收取，單位GB	按實際解凍的數據量收取，單位GB
數據訪問特點	即時訪問 ms延遲	即時訪問 ms延遲	數據需要先解凍，解凍完成後才能讀取，解凍時間需要1分鐘時間
圖片處理	支援	支援	支援，但需要先解凍



说明：

“數據取回費用”中的數據是從底層分布式儲存系統讀取的數據量，在公網傳輸的數據量會計入到流出流量的計費項中。

儲存類型支援的API

API	標準儲存類型	低頻訪問儲存類型	Archive Storage類型
Bucket建立、刪除、查詢			
PutBucket	支援	支援	支援
GetBucket	支援	支援	支援
DeleteBucket	支援	支援	支援
Bucket ACL設定相關			
PutBucketAcl	支援	支援	支援
GetBucketAcl	支援	支援	支援
Bucket日誌功能			
PutBucketLogging	支援	支援	支援
GetBucketLogging	支援	支援	支援
Bucket預設靜態頁面設定			
PutBucketWebsite	支援	支援	不支援
GetBucketWebsite	支援	支援	不支援
Bucket Referer防盜鏈			
PutBucketReferer	支援	支援	支援
GetBucketReferer	支援	支援	支援
Bucket生命週期			
PutBucketLifecycle	支援	支援	支援，只支援數據回收
GetBucketLifecycle	支援	支援	支援
DeleteBucketLifecycle	支援	支援	支援
Bucket跨區域複製			
PutBucketReplication	支援	支援	支援
Bucket跨網域設定			
PutBucketcors	支援	支援	支援
GetBucketcors	支援	支援	支援
DeleteBucketcors	支援	支援	支援
Object操作			

API	標準儲存類型	低頻訪問儲存類型	Archive Storage類型
PutObject	支援	支援	支援
PutObjectACL	支援	支援	支援
GetObject	支援	支援	支援，需要先解凍
GetObjectACL	支援	支援	支援
GetObjectMeta	支援	支援	支援
HeadObject	支援	支援	支援
CopyObject	支援	支援	支援
OptionObject	支援	支援	支援
DeleteObject	支援	支援	支援
DeleteMultipleObjects	支援	支援	支援
PostObject	支援	支援	支援
PutSymlink	支援	支援	支援
GetSymlink	支援	支援	支援
RestoreObject	不支援	不支援	支援
Multipart操作			
InitiateMultipartUpload	支援	支援	支援
UploadPart	支援	支援	支援
UploadPartCopy	支援	支援	支援
CompleteMultipartUpload	支援	支援	支援
AbortMultipartUpload	支援	支援	支援
ListMultipartUpload	支援	支援	支援
ListParts	支援	支援	支援
圖片處理	支援	支援	支援

3.2 建立和使用Archive Storage類型

OSS提供三種儲存類型，本文介紹Archive Storage類型（Archive）的儲存空間的建立與使用。

建立Archive Storage類型的儲存空間

您可以通過控制台、API/SDK和命令列工具建立Archive Storage類型的儲存空間。

- 通過控制台建立

通過控制台建立Archive Storage類型的儲存空間，儲存類型選擇歸檔，如下圖所示。

创建 Bucket

创建存储空间

注意：Bucket 创建成功后，您所选择的存储类型、区域不支持变更。

Bucket 名称

0/63

区域

华南1（深圳）

Endpoint

oss-cn-shenzhen.aliyuncs.com

存储类型

标准存储

低频访问

归档存储

读写权限

私有

公共读

公共读写

服务器端加密

无

AES256

KMS

实时日志查询

开通

不开通

确定

取消

相同区域内的产品内网可以互通；订购后不支持更换区域，请谨慎选择

您在该区域下没有可用的流量包。建议您购买资源包享受更多优惠，点击购买。

归档：数据长期存储、基本不访问，存储单价低于低频访问型。

如何选择适合您的存储类型？

私有：对文件的所有访问操作需要进行身份验证。

将文件上传至OSS后，自动对其进行落地加密存储，KMS加密方式需要进行权限设置，当前KMS仅支持OSS默认托管的CMK，如需试用KMS单独创建的CMK加密(BYOK)，请和我们联系，更多服务器端加密指南

OSS 与日志服务深度结合，免费提供最近7天内的 OSS 实时日志查询。开通该功能后，用户可对 Bucket 的访问记录进行实时查询分析，了解详情

- 通過API/SDK建立

以Java SDK為例：

```
OSSClient ossClient = new OSSClient ( endpoint , accessKeyId , accessKeySecret );
CreateBucketRequest createBucketRequest = new CreateBucketRequest ( bucketName );
// 設定 bucket 許可權為公共讀，預設是私有讀寫 createBucketRequest .
setCannedACL ( CannedAccessControlList . PublicRead );
// 設定 bucket 儲存類型為歸檔類型，預設是標準類型
createBucketRequest . setStorageClass ( StorageClass . Archive );
ossClient . createBucket ( createBucketRequest );
```

`createBucketRequest . setStorageClass ( StorageClass . Archive );`即設定建立的儲存空間的儲存類型為Archive Storage類型。

- 通過OSS命令列工具建立

以OSSUtil為例：

```
./ ossutil mb oss ://[ bucket name ] -- storage - class = Archive
```

`[ bucket name ]`為需要建立的儲存空間名稱。指定`-- storage - class`的參數為`Archive`，用來建立Archive Storage類型的儲存空間。

使用Archive Storage類型

- 上傳數據

Archive Storage類型儲存空間支援PutObject和MultipartUpload兩種上傳方式，不支援AppendObject。基於PutObject和MultipartUpload開發的上傳應用可以直接使用Archive Storage類型。

- 下載數據

Archive Storage類型的數據讀取方式與標準儲存類型和低頻訪問類型的數據讀取方式有所區別。歸檔類型數據在讀取前需要先執行restore操作解凍到可讀取狀態，解凍過程需要1分鐘時間。

歸檔檔案的狀態變換過程如下：

1. 歸檔類型的檔案初始時處於冷凍狀態。
2. 提交解凍（restore）操作後，服務端執行解凍，檔案處於解凍中狀態。
3. 完成解凍後，可以讀取檔案。
4. 解凍狀態預設持續1天，最多延長7天，之後檔案又回到冷凍狀態。

解凍方式有如下幾種：

· 使用控制台解凍

文件名

test/Image 1(2).png

解凍

ETag

14B93614A6BB8AD

链接有效时间 (秒) ?

3600

图片样式

不使用图片样式

使用 HTTPS

☒

URL

ejjing.aliyuncs.com/test/Image%201%282%29.png?Expires=1560760786&OSSAccessKeyId=TMP.AgHpNTqGz825rlvKw8prHqzK81HqSTExDk1gkvOW578AAAwLAIUXw!LOrob7Jl87tLcCFGNe-jxnJKAPAAc0-165S1RBq9ph&Signature=yrXxf%2BTsCr4HDA%3D

下载

打开文件 URL

复制文件 URL

复制文件路径

类型

image/png

即将解冻文件 `test/Image 1(2).png`。

解冻操作预计需要 1 分钟，解冻完成后可进行文件访问。

确定解冻吗？

确定

取消

對需要讀取的檔案，執行解凍操作，解凍過程預計花費1分鐘。期間可以查詢到object處於解凍中狀態。



- 使用API/SDK解冻

以Java SDK举例，调用restoreObject方法进行object解冻：

```
ObjectMeta data objectMeta data = ossClient . getObjectM
etadate ( bucketName , key );
// check whether the object is archive class
StorageCla ss storageCla ss = objectMeta data . getObjectS
torageClas s ();
if ( storageCla ss == StorageCla ss . Archive ) {
    // restore object
```

```
ossClient.restoreObject ( bucketName , key );  
// wait for restore completed  
do {  
    Thread.sleep ( 1000 );  
    objectMeta data = ossClient.getObjectMetadata ( bucketName , key );  
} while ( ! objectMeta data.isRestoreCompleted () );  
}  
// get restored object  
OSSObject ossObject = ossClient.getObject ( bucketName , key );  
ossObject.getObjectContent ().close ();
```

- 使用OSS命令列工具解凍

以OSSUtil為例：

```
./ ossutil restore oss ://[ Bucket name ]/[ Object name ]
```

[Bucket name]和[Object name]為需要做解凍操作的bucket和object名稱。

4 接入OSS

4.1 快速開始

1. 登入[OSS管理主控台](#)，開通OSS。
2. 建立一個Bucket。
3. 上傳和下載檔案。

具體請參見[開始使用阿里雲 OSS](#)文檔。

快速了解OSS的上傳下載

開始使用SDK之前，請先參考開發人員指南關於上傳下載檔案的功能介紹。

OSS是使用RESTful API來操作的，所有的請求都是標準的HTTP請求。

- OSS提供了多種類型的上傳檔案的方法，如使用單次PUT請求完成的[簡單上傳](#)、使用網頁表單直接上傳的[表單上傳](#)、以及用於大檔案上傳的[分區上傳](#)還有適用於視頻監控等領域使用的[追加上傳](#)。
- 同樣也可以使用多種方法從OSS下載檔案，[簡單下載](#)和下載大檔案的[斷點續傳下載](#)。

基於SDK快速開始

1. 開通OSS後，從控制台上獲取AccessKeyId和AccessKeySecret。
2. 下載各種開發語言SDK。
3. 根據SDK的文檔描述，完成上傳、下載檔案等操作。

具體請參見[SDK文檔開發](#)。

4.2 基於OSS的移動開發

開發架構圖

典型的基於OSS的移動開發有四個組件：

- OSS：提供上傳、下載、上傳回調等功能。
- 開發人員的移動客戶端（App或者網頁應用），簡稱客戶端：通過開發人員提供的服務，間接使用OSS。
- 應用伺服器：客戶端互動的伺服器。也是開發人員的商務服務器。
- 阿里雲STS：頒發臨時憑證。

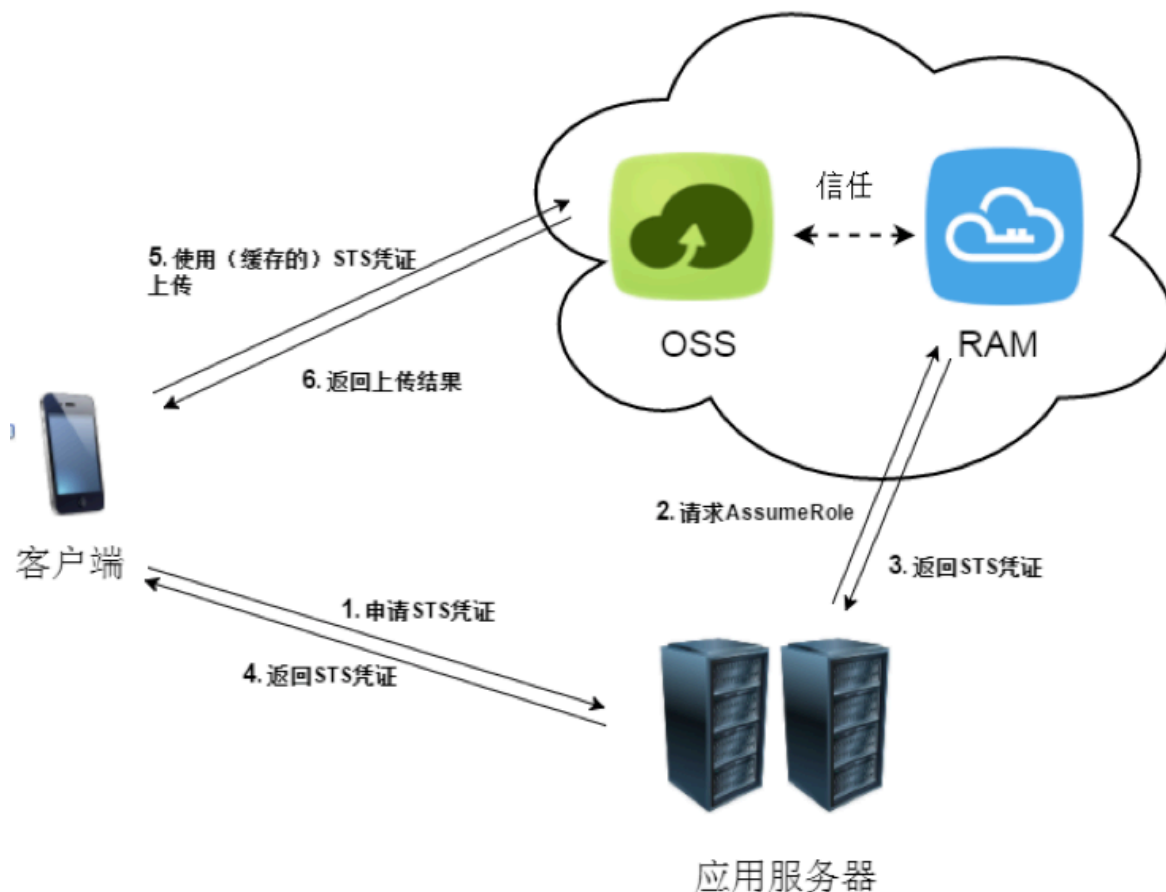
最佳實踐

- [30分鐘快速搭建行動裝置 App直傳服務](#)
- [移動端回調案例分析及代碼下載](#)
- [輕鬆搞定移動端安全性原則](#)

開發商務程序

· 臨時憑證授權的上傳

如圖所示：



具體步驟如下：

1. 客戶端嚮應用伺服器發出上傳檔案到OSS的請求。
2. 應用伺服器向STS伺服器請求一次，獲取臨時憑證。
3. 應用伺服器回覆客戶端，將臨時憑證返回給客戶端。
4. 客戶端獲取上傳到OSS的授權（STS的AccessKey以及Token），調用OSS提供的移動端SDK上傳。
5. 客戶端成功上傳數據到OSS。如果沒有設定回調，則流程結束。如果設定了回調功能，OSS會調用相關的介面。

這裡有幾個要點：

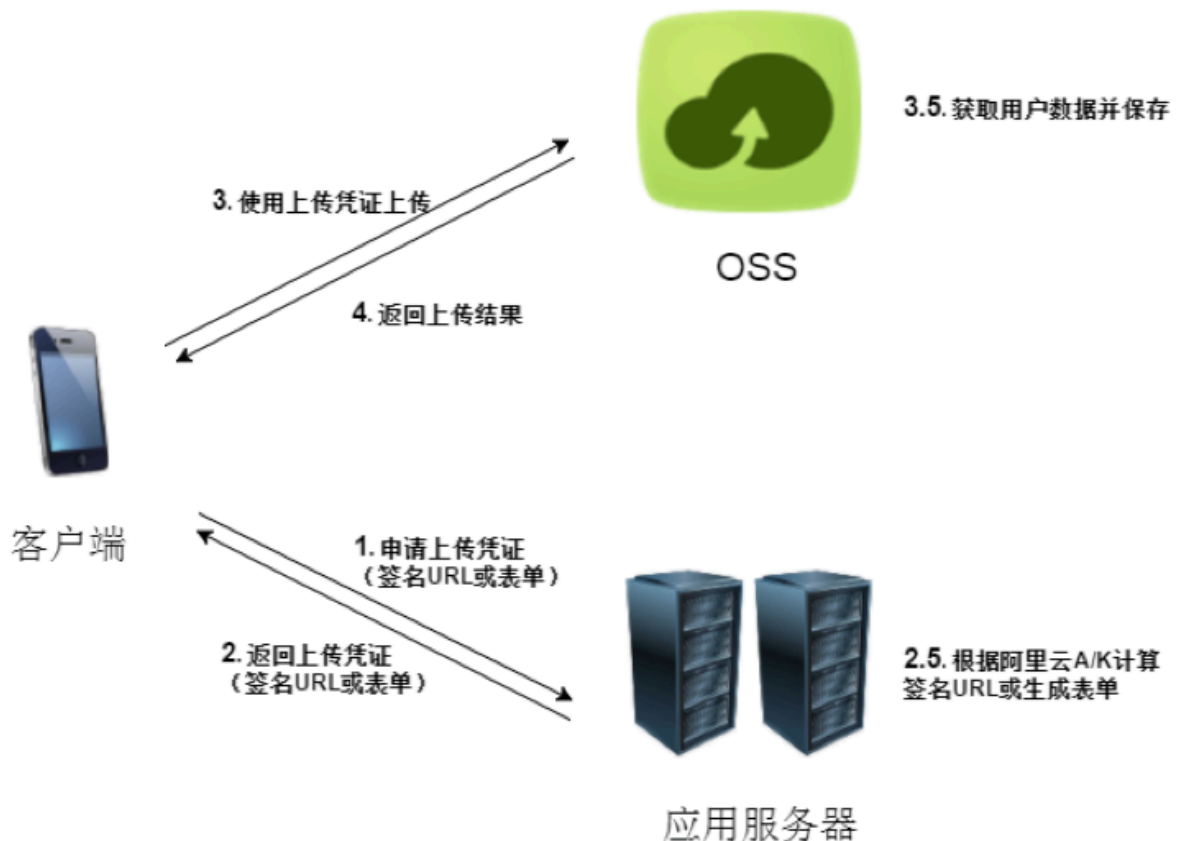
- 客戶端不需要每次都嚮應用伺服器請求授權，在第一次授權完成之後可以緩存STS返回的臨時憑證直到超過失效時間。

- STS提供了強大的許可權控制功能，可以將客戶端的存取權限限制到Object等級，從而實現不同客戶端在OSS上傳的Object的完全隔離，極大的提高了安全係數。

更多資訊請參見[授權給第三方上傳](#)。

- 簽名URL授權的上傳和表單上傳

如圖所示：



具體步驟如下：

1. 客戶端嚮應用伺服器發出上傳檔案到OSS的請求。
2. 應用伺服器回覆客戶端，將上傳憑證（簽名URL或者表單）返回給客戶端。
3. 客戶端獲取上傳到OSS的授權（簽名URL或者表單），調用OSS提供的移動端SDK上傳或者直接表單上傳。
4. 客戶端成功上傳數據到OSS。如果沒有設定回調，則流程結束。如果設定了回調功能，OSS會調用相關的介面。

更多資訊請參見[授權給第三方上傳](#)。

- 臨時憑證授權的下載

跟臨時憑證授權的上傳過程類似：

1. 客戶端嚮應用伺服器發出下載OSS檔案的請求。
2. 應用伺服器向STS伺服器請求一次，獲取臨時憑證。
3. 應用伺服器回覆客戶端，將臨時憑證返回給客戶端。
4. 客戶端獲取下載OSS檔案的授權（STS的AccessKey以及Token），調用OSS提供的移動端SDK下載。
5. 客戶端成功從OSS下載檔案。

這裡有幾個要點：

- 和上傳類似，客戶端對於臨時憑證也可以進行緩存從而提高訪問速度。
- STS同樣也提供了精確到Object的下載許可權控制，和上傳許可權控制結合在一起可以實現各移動端在OSS上儲存空間的完全隔離。

- 簽名URL授權的下載

跟簽名URL授權的上傳類似：

1. 客戶端嚮應用伺服器發出下載OSS檔案的請求。
2. 應用伺服器回覆客戶端，將簽名URL返回給客戶端。
3. 客戶端獲取下載OSS檔案的授權（簽名URL），調用OSS提供的移動端SDK下載。
4. 客戶端成功從OSS下載檔案。



说明：

客戶端不能儲存開發人員的AccessKey，只能獲取應用伺服器簽名的URL或者是通過STS頒發的臨時憑證（也就是STS的AccessKey和Token）。

最佳實踐

- [RAM和STS使用指南](#)

功能使用參考

- SDK: [Android SDK上傳檔案](#)
- SDK: [iOS SDK上傳檔案](#)

5 管理儲存空間

5.1 建立儲存空間

您可以選擇在已有的地域建立儲存空間。同時需要注意有下列限制：

- 同一用戶建立的儲存空間總數不能超過 30 個。
- 每個儲存空間的名字全域唯一，否則會建立失敗。
- 儲存空間的名稱需要符合命名規範。
- 儲存空間一旦建立成功，名稱和所處地域不能修改。

OSS 提供 ACL（Access Control List）許可權控制方法，您可以在建立儲存空間的時候設定相應的儲存空間許可權（ACL），也可以在建立之後修改 ACL。如果不設定 ACL，預設值為私有讀寫。更多資訊，請參見[設定儲存空間讀寫權限](#)。

功能使用參考

- 控制台：[建立儲存空間](#)
- SDK：Java SDK-[Bucket](#)中新建儲存空間
- API：[PutBucket](#)

5.2 設定儲存空間讀寫權限（ACL）

除了在建立儲存空間的時候能夠對儲存空間的 ACL 進行設定，也可以之後根據自己的業務需求對儲存空間的 ACL 進行修改。這個操作只有該儲存空間的建立者有許可權執行。目前儲存空間有三種存取權限：

許可權值	中文名稱	許可權對訪問者的限制
public-read-write	公共讀寫	任何人（包括匿名訪問）都可以對該儲存空間中的檔案進行讀寫操作，所有這些操作產生的費用由該儲存空間的建立者承擔，請慎用該許可權。
public-read	公共讀，私有寫	只有該儲存空間的擁有者或者授權對象可以對該儲存空間內的檔案進行寫操作，任何人（包括匿名訪問）可以對該儲存空間中的檔案進行讀操作。

許可權值	中文名稱	許可權對訪問者的限制
private	私有讀寫	只有資源擁有者或者授權對象可以對該儲存空間內的檔案進行讀寫操作，其他人在未經授權的情況下無法訪問該儲存空間內的檔案。

詳細解釋請參見存取權限。

功能使用參考

設定儲存空間 ACL

- 控制台：[設定存取權限](#)
- SDK：Java SDK-[Bucket](#)中設定Bucket ACL
- API：[Put BucketACL](#)

獲取儲存空間 ACL

- 控制台：登入後可以在儲存空間屬性中查看
- SDK：Java SDK-[Bucket](#)中獲取Bucket ACL
- API：[Get BucketACL](#)

5.3 獲取儲存空間資訊

您可以獲取儲存空間所屬的地域，即資料中心的物理位置資訊。返回的 Location 欄位顯示儲存空間所在的地域資訊，比如華東 1 (原杭州) 的Location 欄位資訊顯示為 oss-cn-hangzhou。請參見 OSS 提供的[訪問網域名稱](#)。

功能使用參考

- 控制台：進入控制台後在儲存空間屬性中直接顯示地域資訊
- API：[Get Bucket Location](#)
- SDK：Java SDK-[Bucket](#)中獲取Bucket地址

5.4 查看儲存空間列表

查看您建立的所有儲存空間列表。

功能使用參考

- 控制台：進入控制台後預設顯示您建立的儲存空間列表
- API：[GetService](#)

- SDK: Java SDK-[Bucket](#)中列舉Bucket

相關參考連結

- [建立Bucket](#)

5.5 綁定自訂網域名

您的檔案上傳到OSS後，會自動生成該檔案的訪問地址。您可以使用此地址訪問OSS檔案。如果您想要通過自訂網域名訪問OSS檔案，需要將自訂網域名訪問綁定在檔案所在的Bucket上，即CNAME。按照中國《互連網管理條例》的要求，所有需要開通這項功能的用戶，必須提供工信部備案號，網域名稱持有人身份證等有效資料，經由阿里雲審批通過後才可以使用。在開通CNAME功能後，OSS將自動處理對該網域名稱的訪問請求。

應用場景

例如，用戶A擁有一個網域名為abc.com的網站，網站的網頁中的連結為 `http://img.abc.com/logo.png`。用戶A此時需要將對網站中圖片的請求遷移到OSS，並且不想修改任何網頁的代碼，也就是對外連結不變，CNAME功能特別適合這種場景。流程如下：

1. 用戶A在OSS上建立一個名為abc-img的Bucket，並上傳了其網站上的圖片。
2. 用戶A通過OSS控制台，提交將img.abc.com這個自訂的網域名稱綁定在abc-img上的申請，並提供相應的材料。
3. 通過阿里雲審核後，OSS後台會將img.abc.com做一個映射到abc-img（此處會做許可權驗證）。
4. 用戶A在自己的網域名稱伺服器上，添加一條CNAME規則，將img.abc.com映射成abc-img.oss-cn-hangzhou.aliyuncs.com（即abc-img的OSS網域名稱）。
5. `http://img.abc.com/logo.png` 請求到達OSS後，OSS找到img.abc.com和abc-img的映射，轉換變成訪問abc-img這個Bucket。也就是說，對 `http://img.abc.com/logo.png` 的訪問，經過OSS後，實際上訪問的是 `http://abc-img.oss-cn-hangzhou.aliyuncs.com/logo.png`。

	CNAME綁定前	CNAME綁定後
流程對比	1. 訪問 <code>http://img.abc.com/logo.png</code>。 2. DNS解析到使用者服務器IP。 3. 訪問使用者服務器上的logo.png。	1. 訪問 <code>http://img.abc.com/logo.png</code>。 2. DNS解析到abc-img.oss-cn-hangzhou.aliyuncs.com。 3. 訪問OSS上 abc-img 裡的 logo.png。

功能使用參考

控制台：[網域名稱管理](#)

5.6 設定防盜鏈

OSS是按使用收費的服務，為了防止您在OSS上的數據被其他人盜鏈，OSS支援基於HTTP header中表頭欄位referer的防盜鏈方法。

您可以通過[OSS管理主控台](#)或者API的方式對一個Bucket設定referer欄位的白名單和是否允許referer欄位為空的請求訪問。

例如，對於一個名為oss-example的Bucket，設定其referer白名單為 `http://www.aliyun.com/`。則所有referer為 `http://www.aliyun.com/` 的請求才能訪問oss-example這個Bucket中的Object。

細節分析

- 用戶只有通過URL簽名或者匿名訪問Object時，才會做防盜鏈驗證。請求的Header中有“Authorization”欄位的，不會做防盜鏈驗證。
- 一個Bucket可以支援多個referer參數。通過控制台設定時通過回車作為分行符號分隔，通過API設定時通過逗號“,”分隔。
- Referer參數支援萬用字元“*”和“?”。
- 用戶可以設定是否允許referer欄位為空的請求訪問。
- 白名單為空時，不會檢查referer欄位是否為空（不然所有的請求都會被拒絕）。
- 白名單不為空，且設定了不允許referer欄位為空的規則；則只有referer屬於白名單的請求被允許，其他請求（包括referer為空的請求）會被拒絕。
- 如果白名單不為空，但設定了允許referer欄位為空的規則；則referer為空的請求和符合白名單的請求會被允許；其他請求都會被拒絕。

- Bucket的三種許可權（private, public-read, public-read-write）都會檢查referer欄位。

萬用字元詳解

- 星號 “*”：可以使用星號代替0個或多個字元。如果正在尋找以AEW開頭的一個檔案，但不記得檔案名其餘部分，可以輸入AEW*，尋找以AEW開頭的所有檔案類型的檔案，如AEWT.txt、AEWU.EXE、AEWI.dll等。要縮小範圍可以輸入AEW*.txt，尋找以AEW開頭的所有檔案類型並以.txt為副檔名的檔案如AEWIP.txt、AEWDF.txt。
- 問號 “?”：可以使用問號代替一個字元。如果輸入love?,尋找以love開頭的一個字元結尾檔案類型的檔案，如lovey、lovei等。要縮小範圍可以輸入love?.doc，尋找以love開頭的一個字元結尾檔案類型並以.doc為副檔名的檔案如lovey.doc、loveh.doc。

功能使用參考

- API: [Put Bucket Referer](#)
- 控制台: [設定防盜鏈](#)

5.7 設定跨域訪問

跨域訪問，或者說JavaScript的跨域訪問問題，是瀏覽器出於安全考慮而設定的一個限制，即同源策略。當來自於A網站的頁面中的JavaScript代碼希望訪問B網站的時候，瀏覽器會拒絕該訪問，因為A、B兩個網站是屬於不同的域。

在實際應用中，經常會有跨域訪問的需求，比如用戶的網站www.a.com，後端使用了OSS。在網頁中提供了使用JavaScript實現的上傳功能，但是在該頁面中，只能向www.a.com發送請求，向其他網站發送的請求都會被瀏覽器拒絕。這樣就導致用戶上傳的數據必須從www.a.com中轉。如果設定了跨域訪問的話，用戶就可以直接上傳到OSS而無需從www.a.com中轉。

跨域資源共用的實現

跨域資源共用（Cross-Origin Resource Sharing），簡稱CORS，是HTML5提供的標準跨域解決方案，OSS支援CORS標準來實現跨域訪問。具體的CORS規則可以參考[W3C CORS規範](#)。其實現如下：

1. CORS通過HTTP請求中附帶Origin的Header來表明自己來源域，比如上面那個例子，Origin的Header就是www.a.com。
2. 伺服器端接收到這個請求之後，會根據一定的規則判斷是否允許該來源域的請求。如果允許的話，伺服器在返回的響應中會附帶上Access-Control-Allow-Origin這個Header，內容為www.a.com來表示允許該次跨域訪問。如果伺服器允許所有的跨域請求，將Access-Control-Allow-Origin的Header設定為*即可，

3. 瀏覽器根據是否返回了對應的Header來決定該跨域請求是否成功，如果沒有附加對應的Header，瀏覽器將會攔截該請求。如果是非簡單請求，瀏覽器會先發送一個OPTIONS請求來獲取伺服器的CORS配置，如果伺服器不支援接下來的操作，瀏覽器也會攔截接下來的請求。

OSS提供了CORS規則的配置從而根據需求允許或者拒絕相應的跨域請求。該規則配置在Bucket等級的。詳情可以參考[PutBucketCORS](#)。

這裡有幾個要點：

- CORS相關的Header附加等都是瀏覽器自動完成的，用戶不需要有任何額外的操作。CORS操作也只在瀏覽器環境下有意義。
- CORS請求的通過與否和OSS的身分識別驗證是完全獨立的，即OSS的CORS規則僅僅是用來決定是否附加CORS相關的Header的一個規則。是否攔截該請求完全由瀏覽器決定。
- 使用跨域請求的時候需要關注瀏覽器是否開啟了Cache功能。當運行在同一個瀏覽器上分別來源於www.a.com和www.b.com的兩個頁面都同時請求同一個跨域資源的時候，如果www.a.com的請求先到達伺服器，伺服器將資源帶上Access-Control-Allow-Origin的Header為www.a.com返回給用戶。這個時候www.b.com又發起了請求，瀏覽器會將Cache的上一次請求返回給用戶，此時Header的內容和CORS的要求不匹配，就會導致後面的請求失敗。

功能使用參考

- API: [跨域資源共用](#)
- SDK: Java SDK-[跨域資源共用](#)
- 控制台: [跨域資源共用](#)

5.8 刪除儲存空間

您可以刪除您建立的儲存空間。如果儲存空間不為空（儲存空間中有檔案或者是尚未完成的分區上傳），則儲存空間無法刪除，必須刪除儲存空間中的所有檔案和未完成的分區檔案後，儲存空間才能成功刪除。如果想刪除儲存空間內部所有的檔案，推薦使用[生命週期管理](#)。

功能使用參考

- API: [Delete Bucket](#)
- SDK: Java SDK-[Bucket](#)中刪除Bucket
- 控制台: [刪除儲存空間](#)

6 上傳檔案

6.1 簡單上傳

適用場景

簡單上傳指的是用戶使用OSS API中的Put Object方法上傳單個Object，可以適用在任何一次HTTP請求互動即可完成上傳的場景，比如小檔案的上傳。

上傳檔案時設定Object Meta

在使用簡單上傳的情況下，可以攜帶Object Meta資訊對Object進行描述，比如可以設定Content-Type等標準HTTP頭，也可以設定自訂資訊。具體請參考[Object Meta](#)。

上傳限制

- 大小限制：Object的大小不能超過 5GB。
- 命名限制
 - 使用UTF-8編碼。
 - 長度必須在1-1023位元組之間。
 - 不能以“/”或者“\”字元開頭。

大檔案上傳

因為使用的是單次HTTP請求，Object過大會導致上傳時間長。在這段時間出現網路原因造成逾時或者連結斷開等錯誤的時候，上傳容易失敗，可以考慮斷點續傳上傳（分區上傳）。當Object大於5GB，這種情況下只能使用斷點續傳上傳（分區上傳），具體參考[斷點續傳上傳](#)。

上傳的安全及授權

為了防止第三方往開發人員的Bucket未經授權上傳，OSS提供了Bucket和Object等級的存取權限控制，詳細解釋見存取控制。為了授權給第三方上傳，OSS除了Bucket和Object等級的存取權限外，還提供了帳號等級的授權，見[上傳安全之授權第三方](#)。

上傳後續操作

在檔案上傳到OSS上後，開發人員可以使用[上傳後回調](#)來向指定的應用伺服器發起回調請求，進行下一步操作。如果上傳的是圖片需要處理，可以使用[上傳圖片後雲端處理](#)。如果上傳的是音頻或者視頻檔案也可以使用[媒體轉碼](#)。

功能使用參考

- API: [PutObject](#)
- SDK: Java SDK-[Object](#)中PutObject
- 控制台: [上傳檔案](#)

最佳實踐

- [RAM和STS使用指南](#)
- [Web端直傳實踐及上傳回調](#)

相關文檔

- [上傳後回調](#)
- [移動端開發上傳場景介紹](#)
- [上傳後下載](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [上傳安全之存取控制](#)
- [上傳安全之授權第三方](#)
- [上傳後對檔案進行複製、刪除等管理操作](#)

6.2 表單上傳

適用場景

指的是用戶使用OSS API中的Post Object請求來完成Object的上傳，上傳的Object不能超過5GB。非常適合嵌入在HTML網頁中來上傳Object，比較常見的應用場景是網站應用程式，以招聘網站為例子：

	不使用表單上傳	表單上傳
流程對比	1. 網站用戶上傳簡曆 2. 網站伺服器回應上傳頁面 3. 簡曆被上傳到網站伺服器 4. 網站伺服器再將簡曆上傳到OSS	1. 網站用戶上傳簡曆 2. 網站伺服器回應上傳頁面 3. 簡曆上傳到OSS

上傳限制

- 大小限制：使用表單上傳時，Object不能超過5GB。

- 命名限制：
 - 使用UTF-8編碼
 - 長度必須在1-1023位元組之間
 - 不能以 “/” 或者 “\” 字元開頭

使用表單上傳的好處

- 從流程上，少了一步轉寄。
- 從架構上來說，原來的上傳都統一走網站伺服器，上傳量過大時，瓶頸在網站伺服器，可能需要擴容網站伺服器。採用表單上傳後，上傳都是直接從客戶端發送到OSS。上傳量過大時，壓力都在OSS上，由OSS來保障服務品質。

上傳的安全及授權

- 為了授權給第三方上傳，這裡使用的是Post Policy的機制，詳情請參見[PostObject](#)。

使用表單上傳的基本步驟

1. 構建一個Post Policy。

Post請求的Policy表單域用於驗證請求的合法性。例如可以指定上傳的大小，可以指定上傳的Object名字等，上傳成功後客戶端跳轉到的URL，上傳成功後客戶端收到的狀態碼。具體請參考[Post Policy](#)。

例如如下Policy，網站用戶能上傳的過期時間是2115-01-27T10:56:19Z（這裡為了測試成功寫的過期時間很長，實際使用中不建議這樣設定），能上傳的檔案最大為104857600位元組。

```
以 Python 代碼為例子，Policy 是 json 格式的字元串。
policy = "{\" expiration \": \" 2115 - 01 - 27T10 : 56 : 19Z \", \"
conditions \": [[ \" content - length - range \", 0 , 104857600
]]}"
```

2. 將Policy字元串進行base64編碼。
3. 用OSS的AccessKeySecret對base64編碼後的Policy進行簽名。
4. 構建上傳的HTML頁面。
5. 開啟HTML頁面，選擇檔案上傳。

完整Python程式碼範例：

```
# coding = utf8
import md5
import hashlib
import base64
import hmac
from optparse import OptionParser
def convert_base64 ( input ):
    return base64 . b64encode ( input )
def get_signature ( key , policy ):
```

```

    return base64 . b64encode ( hmac . new ( key , policy ,
hashlib . sha1 ). digest () )
def get_form ( bucket , endpoint , access_key _id , access_key
_secret , out ):
    # 1 構建一個 Post Policy
    policy = "{ \" expiration \": \" 2115 - 01 - 27T10 : 56 : 19Z \", \"
conditions \": [ [ \" content - length - range \", 0 , 1048576 ] ] }"
    print ( " policy : % s " % policy )
    # 2 將 Policy 字元串進行 base64 編碼
    base64poli cy = convert_base64 ( policy )
    print ( " base64_encode_policy : % s " % base64poli cy )
    # 3 用 OSS 的 AccessKeySecret 對編碼後的 Policy 進行簽名
    signature = get_sign_policy ( access_key _secret ,
base64poli cy )
    # 4 構建上傳的 HTML 頁面
    form = '''
    < html >
    < meta http - equiv = content - type content = " text /
html ; charset = UTF - 8 ">
    < head >< title > OSS 表單上傳( PostObject )</ title ></ head >
    < body >
    < form action = " http :// % s . % s " method = " post "
enctype = " multipart / form - data ">
    < input type = " text " name = " OSSAccessKey Id "
value = "% s ">
    < input type = " text " name = " policy " value = "%
s ">
    < input type = " text " name = " Signature " value
= "% s ">
    < input type = " text " name = " key " value = "
upload / ${ filename } ">
    < input type = " text " name = " success_ac
tion_redirect " value = " http :// oss . aliyun . com ">
    < input type = " text " name = " success_ac
tion_status " value = " 201 ">
    < input name = " file " type = " file " id = " file
">
    < input name = " submit " value = " Upload " type = "
submit ">
    </ form >
    </ body >
    </ html >
    ''' % ( bucket , endpoint , access_key _id , base64poli cy ,
signature )
    f = open ( out , " wb " )
    f . write ( form )
    f . close ()
    print ( " form is saved into % s " % out )
if __name__ == '__main__':
    parser = OptionParser ()
    parser . add_option ( "", "-- bucket ", dest = " bucket ", help
= " specify " )
    parser . add_option ( "", "-- endpoint ", dest = " endpoint ",
help = " specify " )
    parser . add_option ( "", "-- id ", dest = " id ", help = "
access_key _id " )
    parser . add_option ( "", "-- key ", dest = " key ", help = "
access_key _secret " )
    parser . add_option ( "", "-- out ", dest = " out ", help = " out
put form " )
    ( opts , args ) = parser . parse_args ()
    if opts . bucket and opts . endpoint and opts . id
and opts . key and opts . out :

```

```

        get_form ( opts . bucket , opts . endpoint , opts . id ,
opts . key , opts . out )
    else :
        print " python %s -- bucket = your - bucket -- endpoint
= oss - cn - hangzhou . aliyuncs . com -- id = your - access - key -
id -- key = your - access - key - secret -- out = out - put - form
- name " % __file__

```

將此段代碼保存為post_object.py，然後用python post_object.py來運行。

用法：

```
python post_object.py -- bucket =您的 Bucket -- endpoint
= Bucket 對應的 OSS 網域名稱 -- id =您的 AccessKeyId -- key =您的
AccessKeySecret -- out =輸出的檔案名
```

樣本：

```
python post_object.py -- bucket = oss - sample -- endpoint
= oss - cn - hangzhou . aliyuncs . com -- id = tphxp -- key =
ZQNJzf4QJR krH4 -- out = post . html
```



说明：

- 構建的表單中 success_action_redirect value = http://oss.aliyun.com 表示上傳成功後跳轉的頁面。可以替換成您自己的頁面。
- 構建的表單中 success_action_status value = 201 表示的是上傳成功後返回的狀態碼為201。可以替換。
- 假如指定生成的HTML檔案為post.html，開啟post.html，選擇檔案上傳。在這個例子中如果成功則會跳轉到OSS的首頁面。

功能使用參考

- API: [PostObject](#)

最佳實踐

- [Web端直傳實踐](#)
- [跨域資源共用（CORS）使用指南](#)

相關參考連結

- [上傳後回調](#)
- [移動端開發上傳場景介紹](#)
- [上傳後下載](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [上傳安全之存取控制](#)
- [上傳安全之授權第三方](#)

- 上傳後對檔案進行複製，刪除等管理操作

6.3 斷點續傳

適用場景

當使用簡單上傳（PutObject）功能來上傳較大的檔案到OSS的時候，如果上傳的過程中出現了網路錯誤，那麼此次上傳失敗。重試必須從檔案起始位置上傳。針對這種情況，OSS提供了分區上傳（Multipart Upload）來達到斷點續傳的效果。顧名思義，分區上傳就是將要上傳的檔案分成多個資料區塊（OSS裡又稱之為Part）來分別上傳，上傳完成之後再調用OSS的介面將這些Part組合成一個Object。

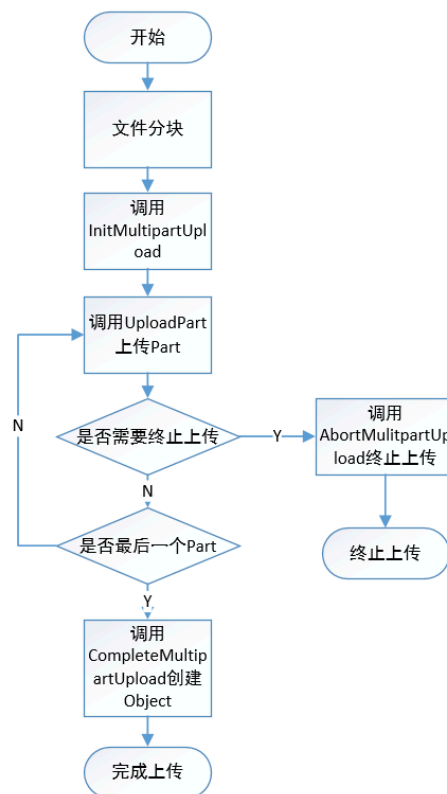
相對於其他的上傳方式，分區上傳適用於以下場景：

- 惡劣的網路環境：如手機端，當出現上傳失敗的時候，可以對失敗的Part進行獨立的重試，而不需要重新上傳其他的Part。
- 斷點續傳：中途暫停之後，可以從上次上傳完成的Part的位置繼續上傳。
- 加速上傳：要上傳到OSS的本地檔案很大的時候，可以並行上傳多個Part以加快上傳。
- 流式上傳：可以在需要上傳的檔案大小還不確定的情況下開始上傳。這種場景在視頻監控等行業應用中比較常見。

基本流程

流程如下：

1. 將要上傳的檔案按照一定的大小分區。
2. 初始化一個分區上傳任務（InitiateMultipartUpload）。
3. 逐個或並行上傳分區（UploadPart）。
4. 完成上傳（CompleteMultipartUpload）。



該過程需注意以下幾點：

- 除了最後一塊Part，其他Part的大小不能小於100KB，否則會導致調用 **CompleteMultipartUpload** 介面時失敗。
- 要上傳的檔案切成Part之後，檔案順序是通過上傳過程中指定的partNumber來確定的，實際執行中並沒有順序要求，因此可以實現並發上傳。具體的並發個數並不是越多速度越快，要結合用戶自身的網路情況和裝置負載綜合考慮。
- 預設情況下，已經上傳但還沒有調用 **CompleteMultipartUpload** 的Part是不會自動回收的，因此如果要終止上傳並刪除佔用的空間請調用 **AbortMultipartUpload**。如果需要自動回收上傳的Part，請參考Object [生命週期管理](#)。

斷點續傳

因為已經上傳的Part的生命週期是永久的，因此很容易可以實現斷點續傳的功能。

在使用分區上傳的過程中，如果系統意外崩潰，可以在重啟的時候通過 **ListMultipartUploads** 和 **ListParts** 兩個介面來獲取某個Object上的所有的分區上傳任務和每個分區上傳任務中上傳成功的Part列表。這樣就可以從最後一塊成功上傳的Part開始繼續上傳，從而達到斷點續傳的效果。暫停和恢復上傳實現原理也是一樣的。

斷點續傳功能在行動裝置和大檔案上傳中的優勢尤為明顯。

上傳限制

- 大小限制：在這種上傳方式下，Object的大小是由Part來決定的，最大支援10000塊Part。每塊Part最小100KB（最後一塊可以比100KB小），最大5GB。Object的大小不能超過48.8TB。
 -
- 命名限制
 - 使用UTF-8編碼
 - 長度必須在1-1023位元組之間
 - 不能以“/”或者“\”字元開頭

上傳的安全及授權

為了防止第三方往開發人員的Bucket未經授權上傳，OSS提供了Bucket和Object等級的存取權限控制，詳細解釋見存取控制。為了授權給第三方上傳，OSS除了Bucket和Object等級的存取權限外，還提供了帳號等級的授權，見[上傳安全之授權第三方](#)。

上傳後續操作

- 在檔案上傳到OSS上後，開發人員可以使用[上傳後回調](#)來向指定的應用伺服器發起回調請求，進行下一步操作。
- 如果上傳的是圖片，可以使用[圖片服務](#)進行後續處理。
- 如果上傳的是音頻或者視頻檔案，可以使用[媒體轉碼](#)進行後續處理。

功能使用參考：

- API：
 - [MultipartUpload](#)
 - [InitiateMultipartUpload](#)
 - [UploadPart](#)
 - [UploadPartCopy](#)
 - [CompleteMultipartUpload](#)
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [ListParts](#)
- SDK：Java SDK-[MultipartUpload](#) 中的分區上傳

最佳實踐

- [RAM和STS使用指南](#)
- [Web端直傳實踐](#)

相關參考連結：

- [上傳後回調](#)
- [移動端開發上傳場景介紹](#)
- [上傳後下載](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [上傳安全之存取控制](#)
- [上傳安全之授權第三方](#)
- [上傳後對檔案進行複製、刪除等管理操作](#)

6.4 追加上傳

適用場景

之前提到的上傳方式，比如[簡單上傳](#)，[表單上傳](#)，[斷點續傳上傳](#)等，建立的Object都是Normal類型，這種Object在上傳結束之後內容就是固定的，只能讀取，不能修改。如果Object內容發生了改變，只能重新上傳同名的Object來覆蓋之前的內容，這也是OSS和普通檔案系統使用的一個重大區別。

正因為這種特性，在很多應用場景下會很不方便，典型比如視頻監控、ApsaraVideo for Live領域等，視頻數據在即時的不斷產生。如果使用其他上傳方式，只能將視頻流按照一定規律切分成小塊然後不斷的上傳新的Object。這種方式在實際使用上存在很明顯的缺點：

- 軟體架構比較複雜，需要考慮檔案分塊等細節問題。
- 需要有位置保存元數據，比如已經生成的Object列表等，然後每次請求都重複讀取元數據來判斷是否有新的Object生成。這樣對伺服器的壓力很大，而且客戶端每次都需要發送兩次網路請求，延時上也會有一定的影響。
- 如果Object切分的比較小的話，延時比較低，但是眾多Object會導致管理起來很複雜。如果Object切分的比較大的話，數據的延時又會很高。

為了簡化這種場景下的開發成本，OSS提供了用戶通過追加上傳（Append Object）的方式在一個Object後面直接追加內容的功能。通過這種方式操作的Object的類型為Appendable Object，而其他方式上傳的Object類型為Normal Object。每次追加上傳的數據都能夠即時可讀。

如果使用追加上傳，那麼上述場景的架構就變得很簡單。視頻數據產生之後即時地通過追加上傳到同一個Object，而客戶端只需要定時獲取該Object的長度與上次讀取的長度進行對比，如果發現有新的數據可讀，那麼就觸發一次讀操作來獲取新上傳的數據部分即可。通過這種方式可以很大的簡化架構，增強擴充性。

不僅在視頻場景，在日誌追加上傳的場景下，追加上傳也能發揮作用。

上傳限制

- 大小限制：在這種上傳方式下，Object不能超過5GB。
- 命名限制
 - 使用UTF-8編碼
 - 長度必須在1-1023位元組之間
 - 不能以“/”或者“\”字元開頭
- 檔案類型：只有通過追加上傳建立的檔案才可以後續繼續被追加上傳。也就是說，其他通過簡單上傳、表單上傳、分區上傳得到的檔案，不能在這些檔案後面追加上傳新的內容。
- 後續操作限制：通過追加上傳的檔案，不能被複製，可以修改檔案本身的meta資訊。

上傳的安全及授權

為了防止第三方往開發人員的Bucket未經授權上傳，OSS提供了Bucket和Object等級的存取權限控制，詳細解釋見存取控制。為了授權給第三方上傳，OSS除了Bucket和Object等級的存取權限外，還提供了帳號等級的授權，見[上傳安全之授權第三方](#)。

上傳後續操作

如果上傳的是圖片需要處理，可以使用[上傳圖片後雲端處理](#)。如果上傳的是音頻或者視頻檔案也可以使用[媒體轉碼](#)。

功能使用參考

- API: [Append Object](#)
- SDK: Java SDK-[追加檔案樣本](#)



说明:

追加上傳不支援上傳回調操作。

最佳實踐

- [RAM和STS使用指南](#)

相關參考連結

- [上傳後下載](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [上傳安全之存取控制](#)
- [上傳安全之授權第三方](#)

6.5 授權給第三方上傳

適用場景

在典型的C/S系統架構中，伺服器端負責接收並處理客戶端的請求。那麼考慮一個使用OSS作為後端的儲存服務，客戶端將要上傳的檔案發送給伺服器端，然後伺服器端再將數據轉寄上傳到OSS。在這個過程中，一份數據需要在網路上傳輸兩次，一次從客戶端到伺服器端，一次從伺服器端到OSS。當訪問量很大的時候，伺服器端需要有足夠的頻寬資源來滿足多個客戶端的同時上傳的需求，這對架構的伸縮性提出了挑戰。

為了解決這種場景帶來的挑戰，OSS提供了授權給第三方上傳的功能。使用這個功能，每個客戶端可以直接將檔案上傳到OSS而不是通過伺服器端轉寄，節省了自建伺服器的成本，並且充分利用了OSS的海量資料處理能力，無需考慮頻寬和並發限制等，可以讓客戶專心於業務處理。

目前授權上傳可以由兩種實現方式：URL簽名和臨時訪問憑證。

URL簽名

URL簽名是授權訪問的一種方式，即在請求的URL中帶OSS AccessKeyId和Signature欄位，這樣用戶就可以直接使用該URL來進行上傳。每個URL簽名中攜帶有過期時間以保證安全。具體的做法可以參考[在URL中包含簽名](#)。

臨時訪問憑證

臨時訪問憑證是通過阿里雲Security Token Service(STS)來實現授權的一種方式。其實現請參見[STS Java SDK](#)。臨時訪問憑證的流程如下：

1. 客戶端向伺服器端發起獲得授權的請求。伺服器端先驗證客戶端的合法性。如果是合法客戶端，那麼伺服器端會使用自己的AccessKey來向STS發起一個請求授權的請求，具體可以參考存取控制。
2. 伺服器端獲取臨時憑證之後返回給客戶端。
3. 客戶端使用獲取的臨時憑證來發起向OSS的上傳請求，更詳細的請求構造可以參考[臨時授權訪問](#)。客戶端可以緩存該憑證用來上傳，直到憑證失效再向伺服器端請求新的憑證。

最佳實踐

- [RAM和STS使用指南](#)
- [Web端直傳實踐及上傳回調](#)

相關參考連結

- [上傳後回調](#)
- [移動端開發上傳場景介紹](#)

- [上傳後下載](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [上傳安全之存取控制](#)
- [表單上傳](#)

6.6 上傳回調

適用場景

OSS在上傳檔案完成的時候可以提供回調（Callback）給應用伺服器。您只需要在發送給OSS的請求中攜帶相應的Callback參數，即能實現回調。現在支援Callback的API 介面有：PutObject、PostObject、CompleteMultipartUpload。

上傳回調的一種典型應用場景是與授權第三方上傳同時使用，客戶端在上傳檔案到OSS的時候指定到伺服器端的回調，當客戶端的上傳任務在OSS執行完畢之後，OSS會向應用伺服器端主動發起HTTP請求進行回調，這樣伺服器端就可以及時得到上傳完成的通知從而可以完成諸如資料庫修改等操作，當回調請求接收到伺服器端的響應之後OSS才會將狀態返回給客戶端。

OSS在向應用伺服器發送POST回調請求的時候，會在POST請求的body中包含一些參數來攜帶特定的資訊，這些參數有兩種，一種是系統定義的參數，如Bucket名稱、Object名稱等；另外一種就是自訂的參數，您可以在發送帶回調的請求給OSS的時候根據應用邏輯的需要指定這些參數。您可以通過使用自訂參數來攜帶一些和應用邏輯相關的資訊，比如發起請求的用戶id等。具體使用自訂參數的方法可以參考[Callback](#)。

通過適當的使用上傳回調機制，能很好的降低客戶端的邏輯複雜度和網路消耗。流程如下：



說明：

- 支援區域包括中國大陸區域（暫不支援華北3、華北5）、香港區域、亞太東南1、亞太東南2、美東、美西、亞太東北1、歐洲中部1、中東東部1
- 目前只有簡單上傳(PutObject)、表單上傳(PostObject)、分區上傳完成（Complete Multipart Upload）操作支援上傳回調功能。

功能使用參考

- API: [Callback](#)
- SDK: iOS [上傳後回調通知](#)

最佳實踐

- [Web端直傳實踐及上傳回調](#)
- [如何搭建一個回調應用伺服器（包括範例程式碼下載）](#)

相關參考連結

- [移動端直傳服務](#)
- [移動端許可權管理](#)
- [移動端開發上傳場景介紹](#)
- [上傳後下載](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [上傳安全之存取控制](#)
- [上傳安全之授權第三方](#)
- [上傳後對檔案進行複製、刪除等管理操作](#)

6.7 RTMP推流上傳

OSS支援使用RTMP協議推送H264編碼的視頻流+AAC編碼的音頻流到OSS。推送到OSS的音視頻數據可以點播播放；在對延遲不敏感的應用場景，也可以做直播用途。

通過RTMP協議上傳音視頻數據目前有以下限制：

- 只能使用RTMP推流的方式，不支援拉流。
- 必須包含視頻流，且視頻流格式為H264。
- 音頻流是可選的，並且只支援AAC格式，其他格式的音頻流會被丟棄。
- 轉儲只支援HLS協議。
- 一個LiveChannel同時只能有一個客戶端向其推流。

下面會分別介紹如何推送音視頻流到OSS，以及如何點播，直播播放。

向OSS推送音視頻數據

- 獲得推流地址

使用SDK調用PutLiveChannel介面，建立一個LiveChannel，並獲取對應的推流地址。如果Bucket的許可權控制（ACL）為公共讀寫（public-read-write），那麼可以直接使用得到的推流地址進行推流；否則需要進行簽名操作。

以使用python SDK為例，獲取未簽名以及簽名推流地址的代碼如下：

```
# 以使用 python sdk 為例
from oss2 import *
from oss2.models import *
host = "oss-cn-hangzhou.aliyuncs.com" # just for example
accessid = "your-access-id"
accesskey = "your-access-key"
bucket_name = "your-bucket"
channel_name = "test-channel"
auth = Auth(accessid, accesskey)
bucket = Bucket(auth, host, bucket_name)
channel_config = LiveChannelInfo(target=LiveChannelInfoTarget())
channel = bucket.create_live_channel(channel_name, channel_config)
publish_url = channel.publish_url
signed_publish_url = bucket.sign_rtmp_url("test-channel", "playlist.m3u8", 3600)
```

獲得的推流地址樣本如下：

```
publish_url = rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/test-channel
signed_publish_url = rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/your-channel?OSSAccessKeyId=LGarWxxxxx xjKWg6 & playlistName=t.m3u8 & Expires=1472201595 & Signature=bjKraZTTyz z9 % 2FpYoomDx4 Wgh % 2FlM % 3D "
```

- 使用ffmpeg進行推流

可以使用ffmpeg推送本地的視頻檔案到OSS，命令如下：

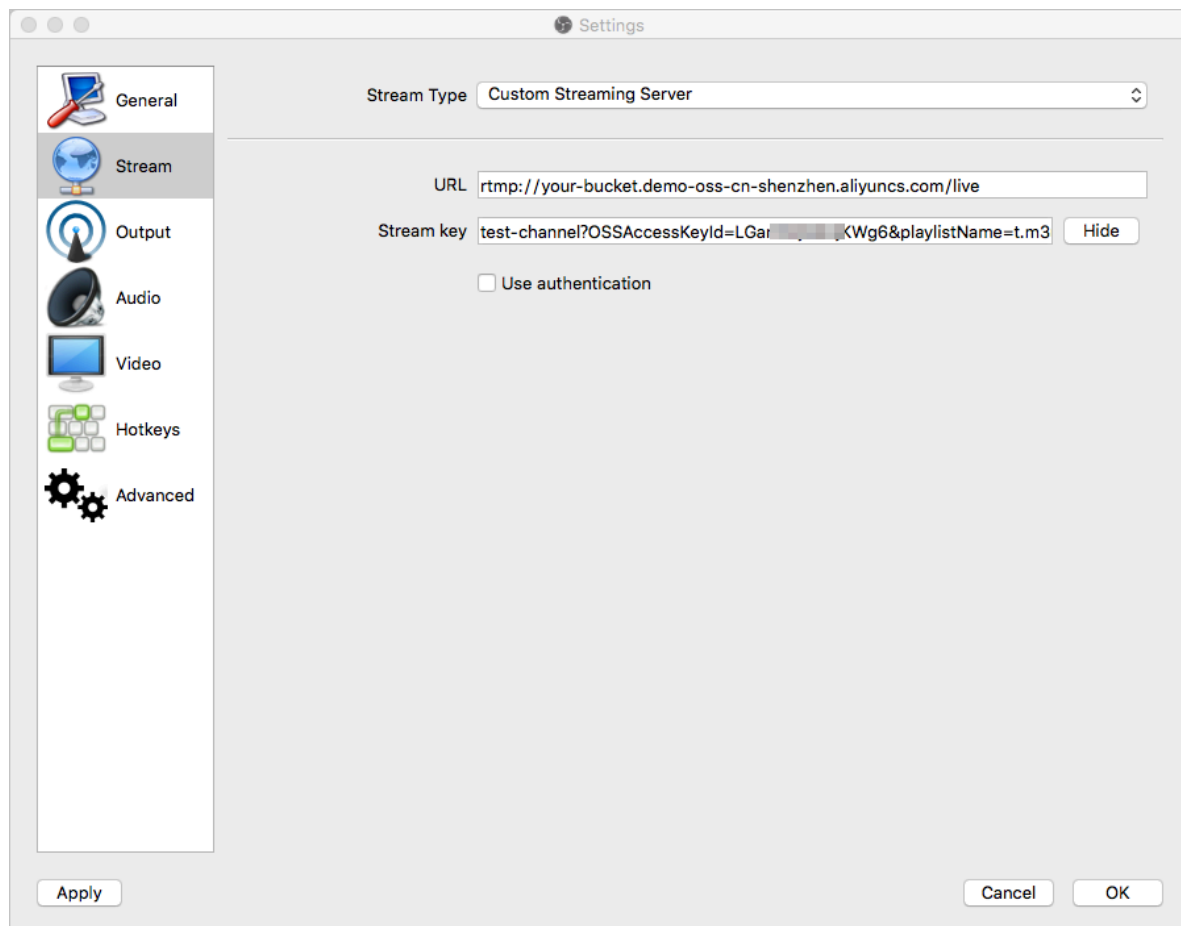
```
ffmpeg -i 1.flv -c copy -f flv "rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/test-channel?OSSAccessKeyId=LGarWxxxxx xjKWg6 & Expires=
```

```
1472199095 & Signature =% 2FAvRo7FTs s1InBKgwn7 Gz % 2FUlp9w % 3D  
"
```

- 使用OBS進行推流

首先點擊Settings，在 URL 框中輸入前面步驟獲取的推流地址，然後點擊OK開始推流即可。

如下圖所示，請注意推流地址的拆分方式：



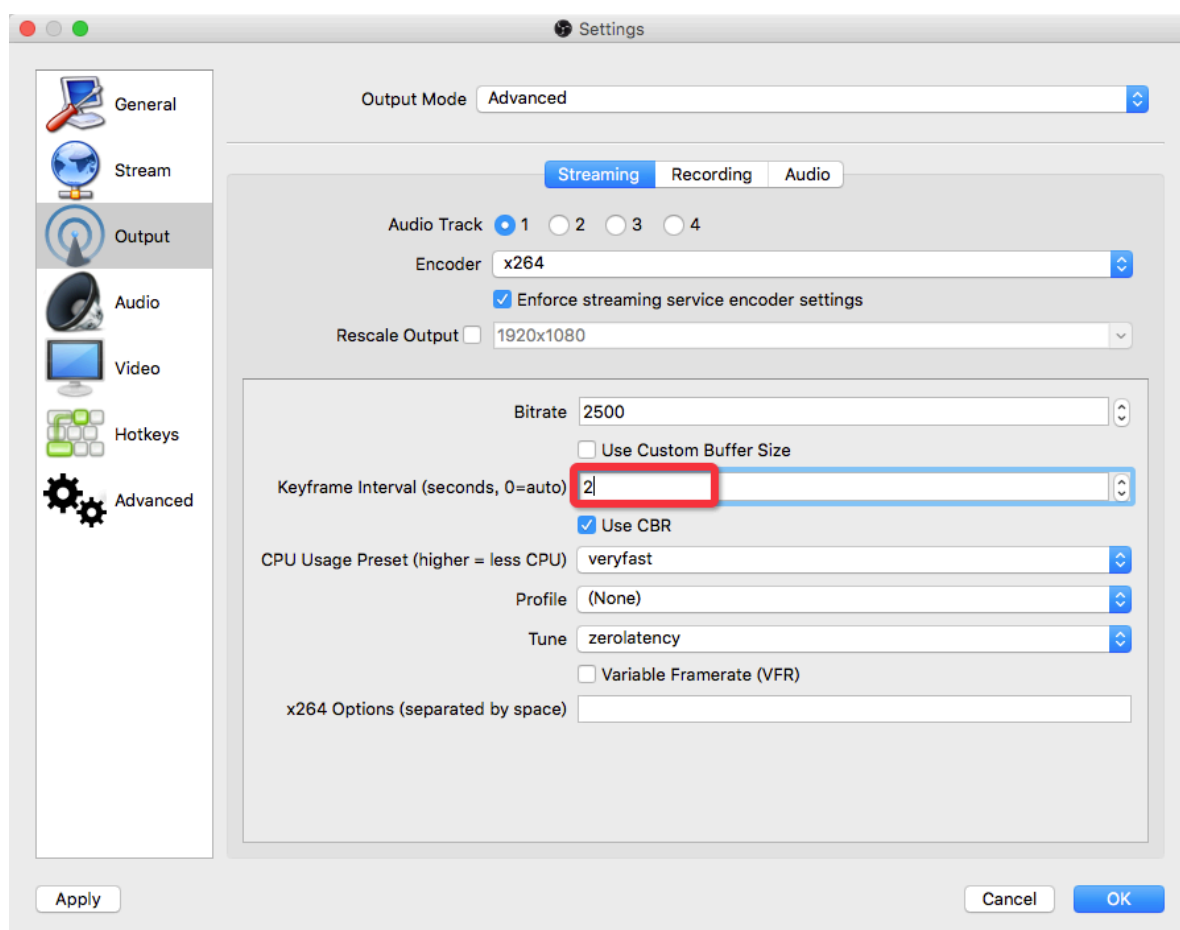
播放推送到OSS的音視頻數據

· 直播場景

在推流的過程中，可以通過HLS協議播放正在推送的內容，各個平台的播放方法如下：

- 在Android、iOS等移動平台，直接在瀏覽器輸入LiveChannel對應的播放地址即可。
- Mac OS可以使用safari瀏覽器進行播放。
- PC端可以安裝vlc播放器進行播放。

為了直播流暢，可以設定比較小的FragDuration，例如2s；另外，GOP的大小最好固定且與LiveChannel的FragDuration配置一致。OBS的GOP（即 keyframe Interval）設定方法如下：



· 點播場景

推流的過程中，OSS總是以直播的方式更新m3u8檔案。所以對於點播的場景，需要在推流結束後，調用PostVodPlaylist介面來組裝一個點播用的m3u8檔案，然後使用該檔案地址來播放。

對於點播的場景，可以設定較大的GOP來減少ts檔案數，降低碼率。

7 下載檔案

7.1 簡單下載

簡單下載即下載已經上傳的檔案（Object），Object下載是使用HTTP的GET請求來完成的。Object的URL建置規則請參考[OSS的訪問](#)。如果需要使用自訂網域名來訪問Object，請參考[自訂網域名訪問OSS](#)。

當用戶需要訪問某個Object的時候，有兩種情況：

- 該Object沒有匿名讀的許可權，用戶擁有對應的AccessKey，那麼可以使用AccessKey對GET請求進行簽名來訪問。
- 該Object擁有匿名讀的許可權，那麼所有的用戶都可以直接使用GET請求來進行訪問。

具體的Object和Bucket的存取權限控制請參考[存取控制](#)。

如果希望將私有Bucket的Object提供給第三方進行下載，請參考[授權給第三方下載](#)。

如果需要實現斷點下載請參考[斷點續傳下載](#)。

功能使用參考

- API: [Get Object](#)
- SDK: [Java SDK-Object](#)
- 控制台: [獲取檔案訪問地址](#)

最佳實踐

- [RAM和STS使用指南](#)

相關參考連結

- [上傳檔案方式](#)
- [上傳後回調](#)
- [移動端開發下載場景介紹](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [下載安全之存取控制](#)
- [下載安全之授權第三方](#)
- [上傳後對檔案進行複製，刪除等管理操作](#)

7.2 斷點續傳下載

OSS提供了從Object指定的位置開始下載的功能，這樣在下載很大的Object的時候，可以分為多次下載。如果中途下載中斷，重啟的時候也可以從上次最後完成的地點開始繼續下載。

和簡單上傳類似，您也需要對該Object有讀許可權。通過設定參數Range來支援斷點續傳，對於比較大的Object建議使用該功能。Range的定義可參考HTTP RFC。如果在要求標頭中使用Range參數則返回消息中會包含整個檔案的長度和此次返回的範圍。例如：Content-Range: bytes 0-9/44，表示整個檔案長度為44，此次返回的範圍為0-9。如果不符合範圍規範，則傳送整個檔案，並且不在結果中提及Content-Range。返回碼為206。

功能使用參考

- API: [Get Object](#)
- SDK: Java SDK-[分段讀取檔案](#)

相關參考連結

- [上傳檔案方式](#)
- [上傳後回調](#)
- [移動端開發下載場景介紹](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [下載安全之存取控制](#)
- [下載安全之授權第三方](#)
- [上傳後對檔案進行複製，刪除等管理操作](#)

7.3 授權給第三方下載

當您希望將私有Bucket內部的Object授權給第三方下載的時候，不應該直接將AccessKey提供給下載者，而應該使用以下兩種方法。

URL簽名

OSS提供了簽名下載的方法。在實現中，開發人員在URL中加入簽名資訊，把該URL轉給第三方實現授權訪問。第三方用戶只需要使用HTTP的GET請求訪問此URL即可下載Object。

- 實現方式

URL中包含簽名樣本如下：

```
http ://< bucket >.< region >. aliyuncs . com /< object >?  
OSSAccessKeyId =< user   access_key _id >& Expires =< unix  
time >& Signature =< signature_ string >
```

在URL中實現簽名，必須至少包含Signature，Expires，OSSAccessKeyId三個參數。

- OSSAccessKeyId：開發人員的AccessKeyId。
- Expires：開發人員期望URL過期的時間。
- Signature：開發人員簽名的字元串，具體請參考[在URL中包含簽名](#)。



说明：

此串連需要進行URL編碼。

- 功能使用參考

- API：[Get Object](#)
- SDK：Java SDK-[使用URL簽名授權訪問](#)
- 控制台：[獲取檔案訪問地址](#)



说明：

在控制台中只有當Bucket處於私有讀寫權限的時候，獲取的訪問地址才是這種URL中籤名的形式。否則則為不帶簽名的URL形式。

臨時訪問憑證

OSS通過STS (Security Token Service) 提供了臨時憑證給第三方用戶，第三方用戶以在要求標頭部中帶簽名的方式去訪問Object。這種授權方式適合移動場景的下載。臨時訪問憑證實現見[STS Java SDK](#)。

實現方式

第三方用戶向APP伺服器請求，獲取了STS頒發的AccessKeyId, AccessKeySecret以及STS Token。第三方用戶以STS AccessKeyId和AccessKeySecret以及STS Token簽名，去請求開發人員的Object的資源。

功能使用參考

- API：[臨時訪問憑證](#)
- SDK：Java SDK-[Object](#)中使用STS服務臨時授權
- 控制台：[獲取檔案訪問地址](#)

最佳實踐

- [RAM和STS使用指南](#)

相關參考連結

- [上傳檔案方式](#)
- [上傳後回調](#)
- [移動端開發下載場景介紹](#)
- [上傳圖片後雲端處理](#)
- [上傳音頻視頻檔案後雲端處理](#)
- [下載安全之存取控制](#)
- [下載安全之授權第三方](#)
- [上傳後對檔案進行複製，刪除等管理操作](#)

8 管理檔案

8.1 設定對象/檔案元資訊

對象/檔案元資訊（Object Meta）是對上傳到OSS的檔案的屬性描述，分為兩種：HTTP標準屬性（HTTP Headers）和 User Meta（用戶自訂元資訊）。檔案元資訊可以在各種方式上傳時或者拷貝檔案時進行設定。

- HTTP標準屬性

名稱	描述
Cache-Control	指定該Object被下載時的網頁的緩存行為
Content-Disposition	指定該Object被下載時的名稱
Content-Encoding	指定該Object被下載時的內容編碼格式
Content-Language	指定該Object被下載時的內容語言編碼
Expires	過期時間
Content-Length	該Object大小
Content-Type	該Object檔案類型
Last-Modified	最近修改時間

- User Meta

為了便於用戶對Object進行更多描述，OSS中規定所有以x-oss-meta-為首碼的參數視為User Meta，比如x-oss-meta-location。一個Object可以有多個類似的參數，但所有的User Meta總大小不能超過8KB。這些User Meta資訊會在下載GetObject或者HeadObject的時候在HTTP頭部中返回。

上傳Object時設定Object Meta

當上傳Object的時候，可以設定Object Meta。

功能使用參考：

- API: [Put Object](#)
- SDK: Java SDK-[上傳](#)文檔中的設定Object的Http Header和用戶自訂元資訊

分區上傳（斷點續傳）的時候也可以設定Object Meta。

功能使用參考：

- API: [InitiateMultipartUpload](#)
- SDK: Java SDK-[初始化Multipart Upload](#)中的初始化Multipart Upload

上傳Object後修改Object Meta

如果需要修改Object Meta而不修改Object本身的數據，那麼應該使用拷貝Object的介面來實現這個功能，只需要將新的Meta資訊（注意這個Meta必須是全量）放在HTTP頭部中，然後將拷貝的源地址和目標地址都設為目標Object的地址即可。

功能使用參考：

- API: [拷貝Object](#)
- SDK: Java SDK-[通過CopyObjectRequest拷貝Object](#)

獲取Object Meta

如果需要獲取Object Meta而並不需要Object本身的數據，可以使用此功能。

功能使用參考：

- API: [Head Object](#)
- SDK: Java SDK-[獲取檔案元資訊](#)

8.2 查看對象列表

查看對象列表即列出您在Bucket中上傳的檔案（Object），您可以通過OSS的介面調用一次性得到某一Bucket下最多1000個的Object列表。通過下面的四個參數，您可以完成多種拓展功能：

名稱	作用
Delimiter	用於對Object名字進行分組的字元。所有名字包含指定的首碼且第一次出現Delimiter字元之間的Object作為一組元素: CommonPrefixes。
Marker	設定結果從Marker之後按字母排序的第一個開始返回。
MaxKeys	限定此次返回Object的最大數，如果不設定，預設為100，MaxKeys取值不能大於1000。
Prefix	限定返回的Object key必須以Prefix作為首碼。注意使用prefix查詢時，返回的key中仍會包含Prefix。

檔案夾類比功能

OSS服務是沒有檔案夾這個概念的，所有元素都是以Object來儲存。建立類比檔案夾本質上來說是建立了一個size為0的Object。對於這個Object照樣可以上傳下載，只是控制台會對以“/”結尾的Object以檔案夾的方式展示。所以您可以使用上述方式來實現建立類比檔案夾。

您可以通過 Delimiter 和 Prefix 參數的配合類比出檔案夾功能。Delimiter 和 Prefix 的組合效果是這樣的：

- 如果把 Prefix 設為某個檔案夾名，就可以羅列以此 Prefix 開頭的檔案，即該檔案夾下遞迴的所有檔案和子檔案夾（目錄）。檔案名在Contents中顯示。
- 如果再把 Delimiter 設定為“/”時，傳回值就只羅列該檔案夾下的檔案和子檔案夾（目錄），該檔案夾下的子檔案名（目錄）返回在 CommonPrefixes 部分，子檔案夾下遞迴的檔案和檔案夾不被顯示。

舉個例子：

假如在 OSS 的 Bucket：oss - sample 下有如下 Object：

```
檔案 D
目錄 A / 檔案 C
目錄 A / 檔案 D
目錄 A / 目錄 B / 檔案 B
目錄 A / 目錄 B / 目錄 C / 檔案 A
目錄 A / 目錄 C / 檔案 A
目錄 A / 目錄 D / 檔案 B
目錄 B / 檔案 A
```

1. 列出第一層目錄和檔案

根據 API 中請求約定，需要設定 Prefix 為“”，Delimiter 為“/”：

返回結果如下：

```
<? xml version =" 1 . 0 " encoding =" UTF - 8 "?>
< ListBucket Result >
  < Name > oss - sample </ Name >
  < Prefix ></ Prefix >
  < Marker ></ Marker >
  < MaxKeys > 1000 </ MaxKeys >
  < Delimiter ></ Delimiter >
  < IsTruncate d > false </ IsTruncate d >
  < Contents >
    < Key >檔案 D </ Key >
    < LastModifi ed > 2015 - 11 - 06T10 : 07 : 11 . 000Z </
LastModifi ed >
    < ETag >" 8110930DA5 E04B1ED5D8 4D6CC4DC90 80 "</ ETag >
    < Type > Normal </ Type >
    < Size > 3340 </ Size >
    < StorageCla ss > Standard </ StorageCla ss >
    < Owner >
      < ID > oss </ ID >
      < DisplayNam e > oss </ DisplayNam e >
    </ Owner >
  </ Contents >
  < CommonPref ixes >
    < Prefix >目錄 A </ Prefix >
  </ CommonPref ixes >
  < CommonPref ixes >
    < Prefix >目錄 B </ Prefix >
  </ CommonPref ixes >
</ ListBucket Result >
```

可以看到：

Contents 返回的是第一層的檔案：“檔案 D”。

CommonPrefixes 返回的是第一層的目錄：“目錄 A /”和“目錄 B /”，而“目錄 A /”和“目錄 B /”下的檔案名不顯示。

2. 列出第二層目錄 A 底下的目錄和檔案

根據 API 中請求約定，需要設定 Prefix 為“目錄 A”，Delimiter 為“/”：

返回結果如下：

```
<? xml version = " 1 . 0 " encoding = " UTF - 8 " ?>
< ListBucket Result >
  < Name > oss - sample </ Name >
  < Prefix >目錄 A </ Prefix >
  < Marker ></ Marker >
  < MaxKeys > 1000 </ MaxKeys >
  < Delimiter >/</ Delimiter >
  < IsTruncated > false </ IsTruncated >
  < Contents >
    < Key >目錄 A /檔案 C </ Key >
    < LastModified > 2015 - 11 - 06T09 : 36 : 00 . 000Z </
LastModified >
    < ETag >" B026324C69 04B2A9CB4B 88D6D61C81 D1 "</ ETag >
    < Type > Normal </ Type >
    < Size > 2 </ Size >
    < StorageClass > Standard </ StorageClass >
    < Owner >
      < ID > oss </ ID >
      < DisplayName > oss </ DisplayName >
    </ Owner >
  </ Contents >
  < Contents >
    < Key >目錄 A /檔案 D </ Key >
    < LastModified > 2015 - 11 - 06T09 : 36 : 00 . 000Z </
LastModified >
    < ETag >" B026324C69 04B2A9CB4B 88D6D61C81 D1 "</ ETag >
    < Type > Normal </ Type >
    < Size > 2 </ Size >
    < StorageClass > Standard </ StorageClass >
    < Owner >
      < ID > oss </ ID >
      < DisplayName > oss </ DisplayName >
    </ Owner >
  </ Contents >
  < CommonPrefixes >
    < Prefix >目錄 A /目錄 B </ Prefix >
  </ CommonPrefixes >
  < CommonPrefixes >
    < Prefix >目錄 A /目錄 C </ Prefix >
  </ CommonPrefixes >
  < CommonPrefixes >
    < Prefix >目錄 A /目錄 D </ Prefix >
  </ CommonPrefixes >
</ ListBucket Result >
```

可以看到：

Contents 返回的是第二層的檔案：“目錄 A /檔案 C”，“目錄 A /檔案 D”。

CommonPrefixes 返回的是第一層的目錄：“目錄 A /目錄 B /”，“目錄 A /目錄 C /”和“目錄 A /目錄 D /”。而目錄下的檔案名不會被顯示。

功能使用參考

- API: [Get Bucket](#)
- SDK: Java SDK-[列出儲存空間中的檔案](#)

8.3 拷貝對象

拷貝對象即複製Bucket中的檔案。在有些情況下，您可能需要僅僅只是將一些Object從一個Bucket複製到另外一個Bucket，不改變內容。這種情況一般的做法是將Object重新下載然後上傳。但是因為數據實際上都是一樣的，因此浪費了很多網路頻寬。因此OSS提供了CopyObject的功能來實現OSS的內部拷貝，這樣在用戶和OSS之間就無需傳輸大量的數據。

另外，由於OSS不提供重新命名功能，因此如果需要對Object進行重新命名的話，最佳的方法就是調用OSS的CopyObject介面先將原來的數據拷貝成新的檔案名，然後刪除原Object。如果用戶僅僅是需要修改某個Object的一些Object Meta資訊，同樣可以調用CopyObject的介面，將源Object地址和目標Object地址設定成相同的，這樣OSS就會僅更新該Object Meta資訊。Object Meta資訊可以參考[Object Meta](#)。

該操作有以注意事項：

- 您需要有源Object的操作許可權，否則會無法完成操作。
- 該操作不支援跨Region拷貝數據。比如：不支援將杭州Bucket裡的Object拷貝到青島。
- 該操作支援的最大Object大小為1GB。
- 該操作不能對追加上傳產生的Object進行拷貝。

功能使用參考：

- API: [Copy Object](#)
- SDK: Java SDK-[Object](#)

拷貝大檔案

當用戶的Object很大的時候，就必須採用其他的方法來完成。和上傳操作一樣，OSS提供了類似[斷點續傳上傳](#)的功能來完成大檔案的拷貝。

基本操作步驟和[斷點續傳上傳](#)內描述的基本一致，唯一需要注意的就是將[UploadPart](#)替換成[UploadPartCopy](#)。

[UploadPartCopy](#)的語義和[UploadPart](#)基本一致，只是資料來源從HTTP請求直接上傳改成從源Object中獲取。

功能使用參考：

- API: [UploadPartCopy](#)
- SDK: Java SDK-[拷貝大檔案](#)

8.4 刪除對象

刪除對象即刪除上傳在Bucket中的檔案（Object），OSS允許您做如下的刪除動作：

- 單個刪除。指定某個Object刪除。
- 大量刪除。大量刪除一次最多可指定1000個Object。
- 自動刪除。如果需要刪除的Object數目很多，而且刪除的Object有一定的規律，比如定期刪除某些天之前的Object，或者是要清空整個Bucket，這個時候推薦使用[生命週期管理](#)來完成。一旦設定了之後，OSS會根據規則自動刪除已到期的Object，能大大減少您發送刪除請求的次數，提高刪除速度。

功能使用參考

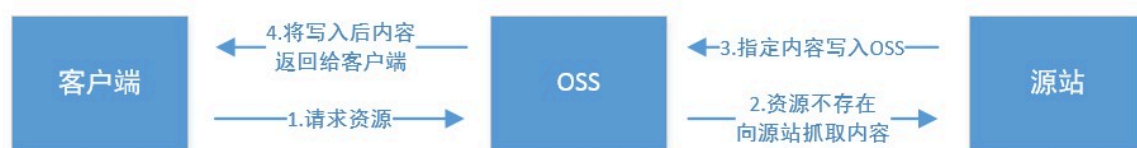
- API: [Delete Object](#)和[Delete Multiple Objects](#)
- SDK: Java SDK[刪除檔案](#)
- 控制台: [刪除檔案](#)

8.5 管理回源設定

通過回源設定，對於獲取數據的請求以多種方式進行回源讀取，滿足您對於數據熱遷移、特定請求重新導向等需求。

通過規則的方式，對每條到OSS的Get請求的url進行匹配，然後按照特定的方式進行回源。規則最多配置5條，順序匹配，直到匹配到有效規則。回源類型分為鏡像方式和重新導向的方式。

鏡像方式



如果配置了鏡像回寫，則當Get一個不存在的檔案時，會向源地址請求這個檔案，返回給用戶，並同時寫入到OSS。

使用場景

鏡像回寫主要用於無縫遷移數據到OSS，即服務已經在自己建立的源站或者在其他雲產品上運行，需要遷移到OSS上，但是又不能停止服務，此時可利用鏡像回寫功能實現。具體場景分析如下：

- 源站有一批冷數據，同時在不斷的生成新的熱數據。

可以先通過遷移工具將冷資料移轉到OSS上，遷移工具為[ossimport2](#)，同時配置鏡像回寫，將源站的地址配置到OSS上。當將網域名稱切換到OSS上（或者阿里雲的CDN，回源到OSS），就算有一部分新生成的數據沒有遷移過來，依然可以在OSS上正常訪問到，且訪問一次後檔案就會存入到OSS。網域名稱切換後，源站已經沒有新的數據產生了，此時再掃描一次，將還沒有導過來的數據一次性匯入到OSS，然後將鏡像回寫配置刪除。

如果配置的源站是IP，那麼將網域名稱遷移到OSS後還可以繼續鏡像到源站，但是如果配置的是一個網域名稱，由於網域名稱本身會解析到OSS或者CDN，那麼鏡像就失去作用了，在這種情況下，可以另外申請一個網域名稱作為鏡像的源站，這個網域名稱與正在服務的網域名稱解析到同一個IP，這樣服務網域名稱遷移的時候就可以繼續鏡像到源站了。

- 只切換源站的部分流量到OSS或者CDN，源站本身還在不斷的產生數據。

遷移方式與上述方式類似，只是流量切換到OSS後，不要將鏡像回寫的配置刪掉，這樣可以保證切換到OSS或者CDN的流量還是能夠拿到源站的數據。

使用細則

- 只有當GetObject()本應該返回404的情況下，OSS才會執行鏡像回寫，向源站請求檔案。
- 向源站請求的url為 `MirrorURL + object`，回寫到OSS的檔案名為“object”，例如bucket為 `example - bucket`，配置了鏡像回寫，MirrorURL為 `http://www.example-domain.com/`，檔案 `image/example_object.jpg` 不在這個bucket裡面，此時去下載這個檔案時，OSS將向 `http://www.example-domain.com/image/example_object.jpg` 發起GET請求，並將結果同時返回給用戶以及寫入到OSS，當下載完成後，這個檔案就已經存在OSS上了，檔案名為 `image/example_object.jpg`，此時相當於將源站的檔案同名的遷移到了OSS上。如果MirrorURL帶有path資訊，比如 `http://www.example-domain.com/dir1/`，其他與上例相同，那麼OSS回源的url為 `http://www.example-domain.com/dir1/image/example_object.jpg`，寫入到OSS的object依然是 `image/example_object.jpg`，此時相當於將源站的某一個目錄下的檔案遷移到OSS上。
- 傳給OSS的header資訊不會傳遞給源站，querystring資訊是否會傳遞給源站取決於控制台回源規則中的配置。
- 如果源站是chunked編碼返回，那麼OSS返回給用戶的也是chunked編碼。
- OSS會將源站的以下頭資訊返回並存為OSS的頭資訊：

```
Content - Type
Content - Encoding
```

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

- 通過鏡像回寫的檔案會添加一個回應頭x-oss-tag，值為“MIRROR” + 空格 + url_decode（回源URL），以上例為例，則為

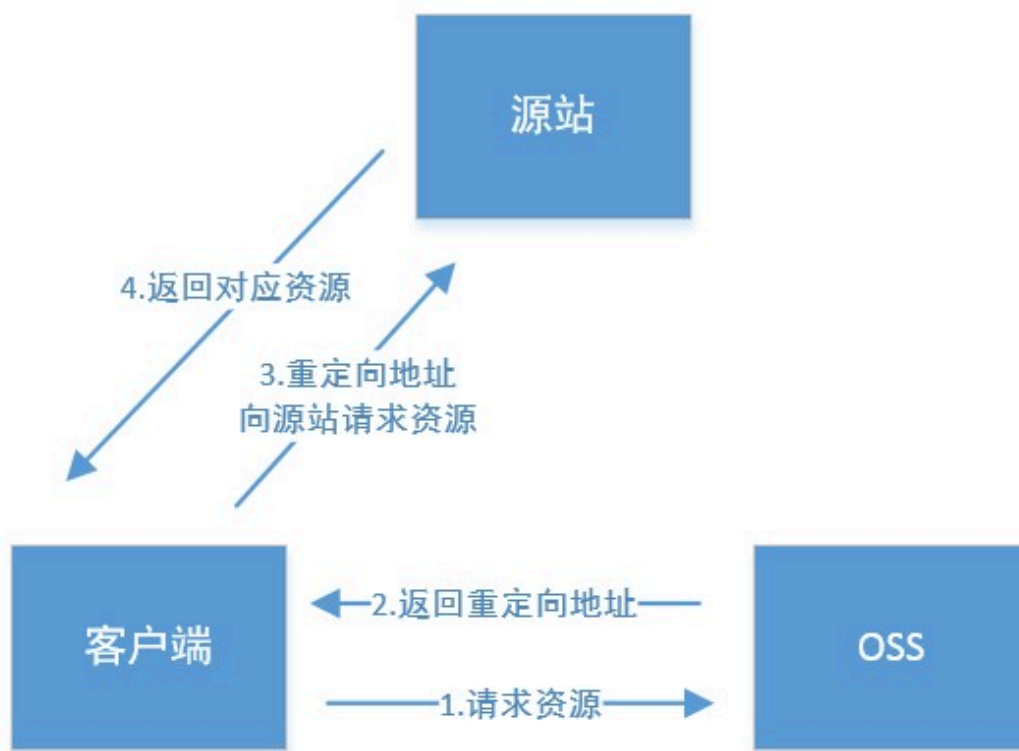
```
x - oss - tag : MIRROR    http % 3a % 2f % 2fwww . example - domain
. com % 2fdir1 % 2fimage % 2fexample_  object . jpg
```

。檔案回寫到OSS上後，只要檔案不被重新覆蓋，每次下載這個檔案都會添加這個頭部，用於表示這個檔案來源於鏡像。

- 假設檔案已經通過鏡像回寫到了OSS，如果源站的相同檔案發生了變化，那OSS不會更新已經存在於OSS上的檔案，因為此時檔案已經在OSS上，不符合鏡像回寫的條件了。
- 如果鏡像源也不存在此檔案，即鏡像源返回給OSS的http status為404，那麼OSS也返回404給用戶，如果是其他非200的狀態碼（包括因為網路原因等獲取不到檔案的錯誤情況），OSS將返回用戶424，錯誤碼為“MirrorFailed”。

重新導向

URL重新導向功能的作用是根據使用者佈建的條件，以及相應的跳轉的配置，向用戶返回一個3xx跳轉。用戶可以利用這種跳轉的功能對檔案做重新導向以及在此基礎之上的各種業務。其流程如下：



使用場景

- 其他資料來源向OSS的無縫遷移。

用戶非同步從自己的資料來源向OSS遷移數據，在此過程中未遷移到OSS的數據通過URL rewrite的方式返回給用戶一個302重新導向請求，用戶的客戶端根據302中的Location從自己的資料來源讀回數據。

- 配置頁面跳轉功能。

比如用戶希望隱藏自己的某些首碼開頭的object，給訪問者返回一個特殊的頁面。

- 配置發生404、或者500錯誤時的跳轉頁面。

發生以上錯誤的時候用戶可以看到一個預先設定的頁面，不至於系統發生錯誤的時候向用戶完全暴露OSS的錯誤。

功能使用參考

- 控制台：[管理回源規則](#)

8.6 SelectObject

SelectObject典型的應用情境是做記錄檔分析，還可以和巨量資料產品結合使用。本文主要介紹如何使用Python SDK以及Java SDK實現以上應用情境。

功能介紹

Object Storage Service (Object Storage Service, 簡稱OSS) 是基於阿里雲飛天分布式系統的海量、安全和高可靠的雲端儲存體服務，是一種面向互連網的大規模、低成本、通用儲存，提供RESTful API，具備容量和處理的彈性擴充能力。OSS不僅非常適合儲存海量的媒體檔案，也適合作為資料倉儲儲存海量的資料檔案。目前Hadoop 3.0已經支援OSS，在EMR上運行Spark、Hive、Presto等服務，以及阿里自研的MaxCompute、HybridDB以及新上線的Data Lake Analytics都支援從OSS直接處理資料。

然而，目前OSS提供的GetObject介面決定了大資料平台只能把OSS資料全部下載到本地然後進行分析過濾，在很多查詢情境下浪費了大量頻寬和用戶端資源。

SelectObject介面是對上述問題的解決方案。其核心思想是大資料平台將條件、Projection下推到OSS層，讓OSS做基本的過濾，從而只返回有用的資料。用戶端一方面可以減少網路頻寬，另一方面也減少了資料的處理量，從而節省了CPU和記憶體用來做其他更多的事情。這使得基於OSS的資料倉儲、資料分析成為一種更有吸引力的選擇。

SelectObject提供了Java和Python 的SDK，其他語言SDK也會很快推出。目前支援RFC 4180標準的CSV（包括TSV等類CSV檔案，檔案的行資料行分隔符號以及Quote字元都可自訂）和Json檔案，且檔案編碼為UTF-8。支援標準儲存類型和低頻訪問儲存類型的檔案（歸檔檔案需要先取回）。支援加密檔案（OSS完全託管加密、KMS託管主要金鑰）。

JSON支援DOCUMENT和LINES兩種檔案。DOCUMENT是指整個檔案是單一的JSON對象，LINES表示整個檔案由一行行的JSON對象組成，每一行是一個JSON對象（但整個檔案本身並不是一個合法的JSON對象），行與行之間以換行分隔字元隔開。OSS Select可以支援常見的\n, \r\n等分隔字元，且無需使用者指定。

· 支援的SQL文法

- SQL 陳述式: Select From Where
- 資料類型: string、int(64bit)、double(64bit)、decimal(128)、timestamp、bool
- 操作: 邏輯條件 (AND,OR,NOT), 算術運算式 (+-*/%), 比較操作(>,<,>=,<=,!=), String 操作 (LIKE, ||)

- 分區查詢

和GetObject提供了基於Byte的分區下載類似，SelectObject也提供了分區查詢的機制，包括兩種分區方式：按行分區、按Split分區。按行分區是常用的分區方式，然而對於稀疏資料來說，按行分區可能會導致分區時負載不均衡。Split是OSS用於分區的一個概念，一個Split包含多行資料，每個Split的資料大小大致相等。按Split分區比按行分區要更加高效。

- 資料類型

關於資料類型，OSS中的CSV資料預設都是String類型，使用者可以使用CAST函數實現資料轉換，比如下面的SQL查詢將_1和_2轉換為int後進行比較。

```
Select * from OSSObject where cast ( _1 as int ) > cast ( _2 as int )
```

同時，對於SelectObject支援在Where條件中進行隱式的轉換，比如下面的語句中第一列和第二列將被轉換成int：

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

對於JSON檔案，如果在SQL中未指定cast函數，則其類型根據Json資料的實際類型而定，標準Json內建的資料類型包括null、bool、int64、double、string等類型。

Python SDK 範例

```
import os
import oss2

def select_callback ( consumed_bytes , total_bytes = None ):
    print ( ' Consumed Bytes : ' + str ( consumed_bytes ) + ' \n ' )

# 首先初始化 AccessKeyId 、 AccessKeySecret 、 Endpoint 等資訊。
# 通過環境變數擷取，或者把諸如 “< yourAccess KeyId >” 替換成真實的 AccessKeyId 等。
#
# 以杭州地區為例， Endpoint 可以是：
# http://oss-cn-hangzhou.aliyuncs.com
# https://oss-cn-hangzhou.aliyuncs.com

access_key_id = os . getenv ( ' OSS_TEST_ACCESS_KEY_ID ' , '< yourAccess KeyId >' )
access_key_secret = os . getenv ( ' OSS_TEST_ACCESS_KEY_SECRET ' , '< yourAccess KeySecret >' )
bucket_name = os . getenv ( ' OSS_TEST_BUCKET ' , '< yourBucket >' )
endpoint = os . getenv ( ' OSS_TEST_ENDPOINT ' , '< yourEndpoint >' )

# 建立儲存空間執行個體，所有檔案相關的方法都需要通過儲存空間執行個體來調用。
bucket = oss2 . Bucket ( oss2 . Auth ( access_key_id , access_key_secret ) , endpoint , bucket_name )
key = ' python_select . csv '
```

```

content = ' Tom   Hanks , USA , 45 \ r \ n '* 1024
filename = ' python_select . csv '
# 上傳 CSV 檔案
bucket . put_object ( key , content )
# Select API 的參數
csv_meta_params = {' CsvHeaderInfo ': ' None ',
' RecordDelimiter ': '\ r \ n ' }
select_csv_params = {' CsvHeaderInfo ': ' None ',
' RecordDelimiter ': '\ r \ n ',
' LineRange ': ( 500 , 1000 ) }

csv_header = bucket . create_select_object_meta ( key ,
csv_meta_params )
print ( csv_header . rows )
print ( csv_header . splits )
result = bucket . select_object ( key , " select * from
ossobject where _3 > 44 ", select_cal_l_back , select_csv
_params )
select_content = result . read ()
print ( select_content )

result = bucket . select_object_to_file ( key , filename ,
" select * from ossobject where _3 > 44 ", select_cal
l_back , select_csv_params )
bucket . delete_object ( key )

### JSON DOCUMENT
key = ' python_select . json '
content = "{ \" contacts \": [{ \" key1 \": 1 , \" key2 \": \" hello
world1 \" }, { \" key1 \": 2 , \" key2 \": \" hello world2 \" } ] }"
filename = ' python_select . json '
# 上傳 JSON DOCUMENT
bucket . put_object ( key , content )
select_json_params = {' JsonType ': ' DOCUMENT ' }
result = bucket . select_object ( key , " select s . key2
from ossobject . contacts [*] s where s . key1 = 1 ", None
, select_json_params )
select_content = result . read ()
print ( select_content )

result = bucket . select_object_to_file ( key , filename ,
" select s . key2 from ossobject . contacts [*] s where s .
key1 = 1 ", None , select_json_params )

bucket . delete_object ( key )
### JSON LINES
key = ' python_select_lines . json '
content = "{ \" key1 \": 1 , \" key2 \": \" hello world1 \" } \ n { \"
key1 \": 2 , \" key2 \": \" hello world2 \" }"
filename = ' python_select . json '
# 上傳 JSON LINE
bucket . put_object ( key , content )
select_json_params = {' JsonType ': ' LINES ' }
json_header = bucket . create_select_object_meta ( key ,
select_json_params )
print ( json_header . rows )
print ( json_header . splits )

result = bucket . select_object ( key , " select s . key2
from ossobject s where s . key1 = 1 ", None , select_json
_params )
select_content = result . read ()
print ( select_content )
result = bucket . select_object_to_file ( key , filename ,

```

```
" select s . key2 from ossobject s where s . key1 = 1
", None , select_json_params )

bucket . delete_object ( key )
```

Python Select API詳解

· select_object

- select_object 樣本：

```
def select_object ( self , key , sql ,
                    progress_callback = None ,
                    select_params = None ):
```

上述樣本的功能：對指定Key的檔案運行SQL，返回查詢結果。

- 參數SQL是原始的SQL字串，無需做base64編碼。
- Progress_callback是可選的用來彙報進度的回呼函數。
- select_params是該API的重點，它指定了select執行的各種參數以及行為。
- headers可以讓使用者指定請求中附帶的header資訊，其行為和get-object一致。比如，對於CSV檔案，在某些情況下可以用bytes來指定SQL查詢在檔案中的範圍。

- select_params支援的參數

參數名稱	描述
Json_Type	■ 當Json_Type沒有指定時，預設為CSV檔案。 ■ 當Json_Type為DOCUMENT時，該檔案為JSON檔案。 ■ 當Json_Type為LINES時，該檔案為JSON LINE檔案。
CsvHeaderInfo	CSV的header資訊，合法值為 None 、 Ignore 、 Use 。 ■ None：該檔案沒有header資訊。 ■ Ignore：該檔案有header資訊但未在SQL中使用。 ■ Use：該檔案有Header資訊且在sql語句中使用了Header中的列名。
CommentCharacter	CSV中的注釋字元。僅支援一個字元，預設為None表示沒有注釋字元。
RecordDelimiter	CSV中的行分隔字元，僅支援一個或兩個字元。預設為\ n 。
OutputRecordDelimiter	Select輸出結果中的行分隔字元，預設為\ n 。
FieldDelimiter	CSV的資料行分隔符號，僅支援一個字元，預設為逗號 (,) 。

參數名稱	描述
OutputFieldDelimiter	Select輸出結果中的資料行分隔符號，預設為逗號 (,)。
QuoteCharacter	CSV 列的引號字元，只支援一個字元，預設為雙引號。引號內的行資料行分隔符號被當做一般字元處理。
SplitRange	使用Split做分區查詢。格式為(start, end)，此處為閉區間，表示查詢範圍從Split start#到end#。
LineRange	使用行做分區查詢。格式為(start, end)，此處為閉區間，表示查詢範圍從行號start#到end#。
CompressionType	壓縮類型，預設為None，可以為GZIP。
KeepAllColumns	該參數為true時表示原CSV檔案中未在select列中出現的列將以空值輸出（但保留列的位置）。預設為False。 樣本如下： 如果CSV檔案中的列為 firstname, lastname, age。SQL 為 <code>select firstname , age from ossobject</code>。如果KeepAllColumns為true，則輸出為firstname,,age（中間多了一個逗號）。如果KeepAllColumns為false，則輸出為firstname,age。引入該選項的原因是讓原本處理Get-Object返回資料的代碼可以不用修改的情況下平移切換到用Select object。
OutputRawData	■ 該參數為True時表示輸出為Select資料，沒有Frame的封裝，也意味著當很長時間不返回資料時會引起逾時。 ■ 該參數為False時表示輸出為Frame封裝的資料，預設是False。
EnablePayloadCrc	為每個Frame計算CRC校正值，預設為 False。
OutputHeader	僅用於CSV檔案，表示輸出結果中第一行是Header資訊。
SkipPartialDataRecord	■ 該參數為True時，當CSV中某一列值不存在或者Json中某一個Key不存在時，直接跳過整個記錄。 ■ 該參數為False時，則把該列當做null來處理。 樣本如下： 假如某一行的列為firstname,lastname,age。SQL為 <code>select _1 , _4 from ossobject</code>。如果為True，則該行被完全跳過，如果為False，則返回firstname,\n。

參數名稱	描述
MaxSkippedRecordsAllowed	允許跳過的行的最大值。預設為0，表示一旦有一行跳過就返回錯誤。
ParseJsonNumberAsString	當該參數為True時表示Json檔案中的數字都解析為字串；為False時則仍按照整數或者浮點數進行解析，False為預設值。 當Json檔案中含有高精度的浮點數時，直接解析為浮點數會丟失精度。如果想保留原始的精度，則可以設定該參數為True，並且在SQL語句中將該列Cast為decimal類型即可。

- `select_object` 傳回值：返回SelectObjectResult對象，該對象支援read()函數以獲得所有select結果。同時也支援__iter__方法。如果Select結果非常大，用read()函數則不是最佳的辦法，因為該調用會阻塞直到select結果完全返回，同時佔用過多的記憶體。

推薦的做法是用__iter__方法（foreach chunk in result），然後對每個chunk進行處理。此做法的優勢是一方面降低記憶體使用量，另一方面OSS伺服器端每處理一個請求chunk用戶端就能及時處理，不必等到全部返回後才處理。

- `select_object_to_file`

```
def select_object_to_file ( self , key , filename , sql ,
                           progress_callback = None ,
                           select_params = None
                           ):
```

上述樣本的功能：對指定Key的檔案運行SQL，將查詢結果寫入指定的檔案。

其他參數同[select_object](#)相同。

- create_select_object_meta
- select_meta_params 的文法結構

```
def create_select_object_meta ( self , key , select_meta_params = None ):
```

上述樣本的功能：對指定Key的檔案進行建立或者擷取select meta。Select meta是指該檔案的總行數、總列數（CSV檔案）、總的Split數。

如果該檔案已經建立過meta，則調用該函數不會重新建立，除非在參數中指定OverwriteIfExists為true。

建立select meta需要掃描整個檔案。

- select_meta_params 中支援的參數

參數名稱	描述
Json_Type	當Json_Type沒有指定時，預設為CSV檔案。若指定，必須為LINES，表示檔案是Json LINES。JSON DOCUMENT不支援該操作。
RecordDelimiter	CSV檔案分行符號。
FieldDelimiter	CSV檔案資料行分隔符號。
QuoteCharacter	CSV檔案列引號符。引號符內的行資料行分隔符號按一般字元處理。
CompressionType	壓縮類型，若指定，只能為None。
OverwriteIfExists	覆蓋原有的Select Meta。正常情況無需使用該選項。

- create_select_object_meta 傳回值：GetSelectObjectMetaResult對象，它包括rows、splits 兩個屬性。對於CSV檔案，其內部的select_resp對象還包括columns值，表示CSV檔案的列數。

Java SDK 範例

```
package    samples ;

import    com . aliyun . oss . ClientBuilderConfiguration ;
import    com . aliyun . oss . model . * ;
import    com . aliyun . oss . OSS ;
import    com . aliyun . oss . OSSClientBuilder ;

import    java . io . IOException ;
import    java . util . ArrayList ;
import    java . util . List ;
import    java . util . concurrent . Callable ;
import    java . util . concurrent . ExecutorService ;
import    java . util . concurrent . Executors ;
import    java . util . concurrent . Future ;
```

```

import com . aliyun . oss . common . auth . * ;
import com . aliyuncs . DefaultAcs Client ;
import com . aliyuncs . exceptions . ClientException ;
import com . aliyuncs . http . MethodType ;
import com . aliyuncs . http . ProtocolType ;
import com . aliyuncs . profile . DefaultProfile ;
import com . aliyuncs . profile . IClientProfile ;
import com . aliyuncs . sts . model . v20150401 . AssumeRole
Request ;
import com . aliyuncs . sts . model . v20150401 . AssumeRole
Response ;

import java . text . SimpleDateFormat ;
import java . util . Calendar ;

/**
 * Examples of create select object metadata and select
 * object .
 */
class MultipartSelector implements Callable < Integer > {

    private OSS client ;
    private String bucket ;
    private String key ;
    private int start ;
    private int end ;
    private String sql ;

    public MultipartSelector ( OSS client , String bucket ,
String key , int start , int end , String sql ){
        this . client = client ;
        this . bucket = bucket ;
        this . key = key ;
        this . start = start ;
        this . end = end ;
        this . sql = sql ;
    }

    @Override
    public Integer call () throws Exception {
        SelectObjectRequest selectObjectRequest =
            new SelectObjectRequest ( bucket , key )
                . withInputStream (
                    new InputSerialization ()
                        . withCsvInputFormat (
                            new CSVFormat ()
                                . withHeaderInfo ( CSVFormat . Header . None )
                                . withRecordDelimiter ( "\n" )
                                . withFieldDelimiter ( "|" ) ) )
                . withSplitRange ( start , end )
                . withOutputSerialization ( new
                    OutputSerialization () . withCsvOutputFormat ( new CSVFormat ( ) )
                        . withCrcEnabled ( true ) ) ;

        selectObjectRequest . setExpression ( sql ) ;
        OSSObject ossObject = client . selectObject (
            selectObjectRequest ) ;
        byte [] buffer = new byte [ 4096 ] ;
        int bytesRead ;
        int totalSize = 0 ;
        try {

```

```

        while (( bytesRead = ossObject . getObjectContent
        (). read ( buffer )) != - 1 ) {
            totalSize += bytesRead ;
        }
        String result = new String ( buffer , 0 ,
totalSize - 1 );
        return new Integer ( Integer . parseInt ( result ));
    }
    catch ( IOException e ){
        System . out . println ( e . toString ());
        return new Integer ( 0 );
    }
}

class RoleCredentialProvider {
    public static final String REGION_CN_ HANGZHOU = " cn -
hangzhou ";
    // 當前 STS API 版本
    public static final String STS_API_VERSION = " 2015 -
04 - 01 ";
    public static final String serviceAccessKeyId = "<
access Key Id that can do assume role >";
    public static final String serviceAccessKeySecret =
"< access key secret >";

    public static final long DurationSeconds = 15 * 60
;

    private Credentials credential ;
    private Calendar expireTime ;

    private String roleArn ;
    private DefaultAcsClient client ;

    public RoleCredentialProvider ( String roleArn ) throws
InvalidCredentialsException {
        this . roleArn = roleArn ;
    }

    private AssumeRoleResponse assumeRole ( String
accessKeyId , String accessKeySecret , String roleArn ,
String roleSessionName , String policy , ProtocolType
protocolType , long durationSeconds ) throws ClientExce
ption {
        try {
            // create Aliyun Acs Client for sending
OpenAPI requests
            if ( this . client == null ) {
                IClientProfile profile = DefaultProfile .
getProfile ( REGION_CN_ HANGZHOU , accessKeyId , accessKeyS
ecret );
                this . client = new DefaultAcsClient ( profile
);
            }
            // create an instance of AssumeRole Request
and set properties
            final AssumeRoleRequest request = new
AssumeRoleRequest ();
            request . setVersion ( STS_API_VERSION );
            request . setMethod ( MethodType . POST );
            request . setProtocol ( protocolType );
            request . setRoleArn ( roleArn );
            request . setRoleSessionName ( roleSessionName );
            request . setPolicy ( policy );

```

```

        request . setDurationSeconds ( durationSeconds );
        // send request and get the response
        final AssumeRole Response response = client .
getAcsResponse ( request );
        return response ;
    } catch ( ClientException e ) {
        throw e ;
    }
}

public Credential sProvider GetCredentialProvider ()
    throws IOException {
    // AssumeRole API 請求參數: RoleArn , RoleSessionName ,
    Policy , and DurationSeconds
    // RoleArn 需要在 RAM 控制台上擷取
    // RoleSessionName 是臨時 Token 的會話名稱, 自己指定用於標識你的
    使用者, 主要用於審計, 或者用於區分 Token 頒發給誰
    // 但是注意 RoleSessionName 的長度和規則, 不要有空格, 只能有'-' '_'
    ' 字母和數字等字元
    // 具體規則請參考 API 文檔中的格式要求
    SimpleDateFormat timeFormat = new SimpleDateFormat
(" yyyy - MM - dd ");
    String roleSessionName = " AssumingRole " +
timeFormat . format ( Calendar . getInstance (). getTime ());
    // read OSS data
    String policy = "{\n " +
        "     \" Version \": \" 1 \",\n " +
        "     \" Statement \": [\n " +
        "         {\n " +
        "             \" Action \": \" oss :*\",\n " +
        "             \" Resource \": [\n " +
        "                 \" acs : oss :*:~:*\",\n " +
        "             ],\n " +
        "             \" Effect \": \" Allow \"\n " +
        "         }\n " +
        "     ]\n " +
        " }";
    // 此處必須為 HTTPS
    ProtocolType protocolType = ProtocolType . HTTPS ;
    try {
        final AssumeRole Response response = assumeRole (
serviceAccessKeyId , serviceAccessKeySecret ,
        roleArn , roleSessionName , policy ,
protocolType , DurationSeconds );
        String ossAccessId = response . getCredentials
(). getAccessKeyId ();
        String ossAccessKey = response . getCredentials
(). getAccessKeySecret ();
        String ossSts = response . getCredentials ().
getSecurityToken ();

        return new DefaultCredentialProvider (
ossAccessId , ossAccessKey , ossSts );

    } catch ( ClientException e ) {
        throw new InvalidCredentialsException (" Unable
to get the temporary AK : " + e );
    }
}

public void setClient ( DefaultAcsClient client ) {
    this . client = client ;
}

```

```

        public void setCredentials ( Credentials creds ) {
            this . credential = creds ;
        }

        public Credentials getCredentials () {
            if ( credential != null && expireTime . after ( Calendar
                . getInstance () ) ) {
                return credential ;
            }

            try {
                Credential sProvider provider = GetCredential
            ialProvider ();
                credential = provider . getCredentials ();
                expireTime = Calendar . getInstance ();
                expireTime . add ( Calendar . SECOND , ( int )
            DurationSeconds - 60 );
                return credential ;
            } catch ( IOException e ) {
                throw new InvalidCredentialsException ( " Unable
            tp get the temporary AK : " + e );
            }
        }
    }

    public class SelectObjectSample {
        private static String endpoint = "< oss endpoint >";
        private static String bucketName = "< bucket >";
        private static String key = "< object >";
        private static String roleArn = "< service role ' s
    ARN >"; // 存取控制台中可以獲得一個 RAM 角色的 ARN , 該角色必須有訪問 OSS 的許可權
        private static String recordDelimiter = "\n";
        private static int threadCount = 10 ;

        public static void main ( String [] args ) throws
        Exception {
            ClientBuilderConfiguration config = new ClientBuilderConfiguration ();
            RoleCredentialProvider provider = new RoleCredentialProvider ( roleArn );
            Credentials credential s = provider . getCredentials ();
            // OSS client = new OSSClientBuilder (). build (
            endpoint , accessKeyId , accessKeySecret , config );
            System . out . println ( " Id " + credential s .
            getAccessKeyId () );
            System . out . println ( " Key " + credential s .
            getSecretAccessKey () );
            System . out . println ( " Token " + credential s .
            getSecurityToken () );
            OSS client = new OSSClientBuilder (). build (
            endpoint , credential s . getAccessKeyId (), credential s .
            getSecretAccessKey (), credential s . getSecurityToken (),
            config );
            int totalSplits = 1 ;
            try {
                SelectObjectMetadata selectObjectMetadata =
            client . createSelectObjectMetadata (
                new CreateSelectObjectMetadataRequest (
                    bucketName , key )
                    . withInputSerialization (
                        new InputSerialization ()
            withCsvInputFormat (

```

```

        new CSVFormat ().
withHeader Info ( CSVFormat . Header . None ). withRecord Delimiter
( recordDeli miter ))));

        totalSplit s = selectObje ctMetadata . getCsvObje
ctMetadata (). getSplits ();
        System . out . println ( selectObje ctMetadata .
getCsvObje ctMetadata (). getTotalLi nes ());
        System . out . println ( totalSplit s );

    }
    catch ( Exception e )
    {
        e . printStack Trace ();
    }

    String sql = " select count (*) from ossobject ";

    ExecutorSe rvice executor = Executors . newFixedTh
readPool ( threadCoun t );
    long startTime = System . currentTim eMillis ();
    List < Future < Integer >> list = new ArrayList < Future
< Integer >>();
    int n = threadCoun t < totalSplit s ? threadCoun t
: totalSplit s ;
    for ( int i = 0 ; i < n ; i ++ ) {
        int start = i * totalSplit s / n ;
        int end = i == n - 1 ? totalSplit s - 1 : ( i
+ 1 ) * totalSplit s / n - 1 ;
        System . out . println ( " start : " + start + " end : "
+ end );
        Callable < Integer > task = new MulipartSe lector (
client , bucketName , key , start , end , sql );
        Future < Integer > future = executor . submit ( task
);
        list . add ( future );
    }

    long totalLines = 0 ;
    for ( Future < Integer > task : list ){
        totalLines += task . get (). longValue ();
    }
    long endTime = System . currentTim eMillis ();
    System . out . println ( " total lines : " + totalLines );
    System . out . printf ( " Total time % dms \ n " , (
endTime - startTime ));
}
}

```

常見的SQL用例

· 常見的SQL用例 (Csv)

應用情境	SQL語句
返回前10行資料	select * from ossobject limit 10
返回第1列和第3列的整數，並且第1列大於第3列	select _1, _3 from ossobject where cast(_1 as int) > cast(_3 as int)

應用情境	SQL語句
返回第1列以'陳'開頭的記錄的個數(註：此處like後的中文需要用UTF-8編碼)	<code>select count(*) from ossobject where _1 like '陳%'</code>
返回所有第2列時間大於2018-08-09 11:30:25且第3列大於200的記錄	<code>select * from ossobject where _2 > cast('2018-08-09 11:30:25' as timestamp) and _3 > 200</code>
返回第2列浮點數的平均值，總和，最大值，最小值	<code>select AVG(cast(_2 as double)), SUM(cast(_2 as double)), MAX(cast(_2 as double)), MIN(cast(_2 as double))</code>
返回第1列和第3列串連的字串中以'Tom'為開頭以'Anderson'結尾的所有記錄	<code>select * from ossobject where (_1 _3) like 'Tom%Anderson'</code>
返回第1列能被3整除的所有記錄	<code>select * from ossobject where (_1 % 3) == 0</code>
返回第1列大小在1995到2012之間的所有記錄	<code>select * from ossobject where _1 between 1995 and 2012</code>
返回第5列值為N,M,G,L的所有記錄	<code>select * from ossobject where _5 in ('N', 'M', 'G', 'L')</code>
返回第2列乘以第3列比第5列大100以上的所有記錄	<code>select * from ossobject where _2 * _3 > _5 + 100</code>

· 常見的SQL用例（Json）

假如我們有如下Json文檔。

```
{
  " contacts ":[
    {
      " firstName ": " John ",
      " lastName ": " Smith ",
      " isAlive ": true ,
      " age ": 27 ,
      " address ": {
        " streetAddress ": " 21 2nd Street ",
        " city ": " New York ",
        " state ": " NY ",
        " postalCode ": " 10021 - 3100 "
      },
      " phoneNumbers ": [
        {
          " type ": " home ",
          " number ": " 212 555 - 1234 "
        },
        {
          " type ": " office ",
          " number ": " 646 555 - 4567 "
        },
        {
          " type ": " mobile ",
```

```

    " number ": " 123    456 - 7890 "
  }
],
" children ": [],
" spouse ": null
}, ..... #此處省略其他類似的節點
]}

```

SQL用例如下：

應用情境	SQL語句
返回所有age是27的記錄	select * from ossobject.contacts[*] s where s.age = 27
返回所有的家庭電話	select s.number from ossobject.contacts[*].phoneNumbers[*] s where s.type = "home"
返回所有單身的記錄	select * from ossobject s where s.spouse is null
返回所有沒有孩子的記錄	select * from ossobject s where s.children[0] is null



说明：

目前沒有專用的空數組的表示方法，用以上語句代替。

最佳實務

· 大檔案分區查詢

如果確定該CSV檔案列中不含有分行符號，則基於Bytes的分區由於不需要建立Meta，其使用最為簡便。如果列中包含分行符號或者是Json檔案時，則使用以下步驟：

1. 調用Create Select Object Meta API獲得該檔案的總的Split數。理想情況下如果該檔案需要用SelectObject，則該API最好在查詢前進行非同步呼叫，這樣可以節省掃描時間。
 2. 根據用戶端資源情況選擇合適的並發度n，用總的Split數除以並發度n得到每個分區查詢應該包含的Split個數。
 3. 在請求Body中用諸如split-range=1-20的形式進行分區查詢。
 4. 如果需要最後可以合併結果。
- SelectObject和Normal類型檔案配合效能更佳。Multipart 以及Appendable類型的檔案由於其內部結構差異導致效能較差。

- 查詢Json檔案時，在SQL的From 語句中儘可能的縮小From後的Json Path 範圍。

樣本：有如下Json檔案。

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

如果要尋找所有postalCode為 10021 開頭的streetAddress，可以這樣寫SQL：

```
select  s . address . streetAddress  from  ossobject . contacts
[*] s  where  s . address . postalCode  like  ‘ 10021 %’
```

也可以這樣寫SQL：

```
select  s . streetAddress  from  ossobject . contacts [*].
address s  where  s . postalCode  like  ‘ 10021 %’
```

第二個SQL由於Json Path更加精確，所以效能會更好。

- 在Json檔案中處理高精度浮點數

在Json檔案中需要進行高精度浮點數的數值計算時，推薦的做法是使用

ParseJsonNumberAsString選項為true，同時將該值cast成Decimal。比如一個屬性a值為123456789.123456789，用 `select s . a from ossobject s where cast ( s . a as decimal ) > 123456789 . 12345` 就可以保持未經處理資料的精度不丟失。

9 訪問與控制

9.1 存取控制

發送訪問OSS的請求

您可以直接使用OSS提供的RESTful API介面訪問或者使用對API介面進行完整封裝的SDK開發包。而每一次向OSS的請求根據當前Bucket許可權和操作不同要求使用者進行身分識別驗證或者直接匿名訪問。對OSS的資源訪問的分類如下：

- 按訪問者的角色可分為擁有者訪問和第三方使用者訪問。這裡的擁有者指的是Bucket的Owner，也稱為開發人員。第三方使用者是指訪問Bucket裡資源的使用者。
- 按訪問者的身份資訊可分為匿名訪問和帶簽名訪問。對於OSS來說，如果請求中沒有攜帶任何和身份相關的資訊即為匿名訪問。帶簽名訪問指的是按照OSS API文檔中規定的在要求標頭部或者在請求URL中攜帶簽名的相關資訊。

AccessKey 類型

目前訪問 OSS 使用的 AK ([AccessKey](#)) 有三種類型，具體如下：

- 阿里雲帳號AccessKey

阿里雲帳號AK特指Bucket擁有者的AK，每個阿里雲帳號提供的AccessKey對擁有的資源有完全的許可權。每個阿里雲帳號能夠同時擁有不超過5個active或者inactive的AK對（AccessKeyId和AccessKeySecret）。

使用者可以登入[AccessKey管理主控台](#)，申請新增或刪除AK對。

每個AK對都有active/inactive兩種狀態。

- Active 表明使用者的 AK 處於啟用狀態，可以在身分識別驗證的時候使用。
- Inactive 表明使用者的 AK 處於非啟用狀態，不能在身分識別驗證的時候使用。



说明：

請避免直接使用阿里雲帳戶的 AccessKey。

- RAM子帳號AccessKey

RAM (Resource Access Management) 是阿里雲提供的資源存取控制服務。RAM帳號AK指的是通過RAM被授權的AK。這組AK只能按照RAM定義的規則去訪問Bucket裡的資源。通過RAM，您可以集中管理您的使用者（比如員工、系統或應用程式），以及控制使用者可以訪問

您名下哪些資源的許可權。比如能夠限制您的使用者只擁有對某一個Bucket的讀許可權。子帳號是從屬於主帳號的，並且這些帳號下不能擁有實際的任何資源，所有資源都屬於主帳號。

- STS帳號AccessKey

STS（Security Token Service）是阿里雲提供的臨時訪問憑證服務。STS帳號AK指的是通過STS頒發的AK。這組AK只能按照STS定義的規則去訪問Bucket裡的資源。

身分識別驗證具體實現

目前主要有三種身分識別驗證方式：

- AK驗證
- RAM驗證
- STS驗證

當使用者以個人身份向OSS發送請求時，其身分識別驗證的實現如下：

1. 使用者將發送的請求按照OSS指定的格式生成簽名字元串。
2. 使用者使用AccessKeySecret對簽名字元串進行加密產生驗證碼。
3. OSS收到請求以後，通過AccessKeyId找到對應的AccessKeySecret，以同樣的方法提取簽名字元串和驗證碼。
 - 如果計算出來的驗證碼和提供的一樣即認為該請求是有效。
 - 否則，OSS將拒絕處理這次請求，並返回HTTP 403錯誤。

對於使用者來說可以直接使用OSS提供的SDK，配合不同類型的AccessKey即可實現不同的身分識別驗證。

許可權控制

針對存放在Bucket的Object的訪問，OSS提供了多種許可權控制，主要有：

- Bucket等級許可權
- Object等級許可權
- 帳號等級許可權（RAM）
- 臨時帳號許可權（STS）

Bucket等級許可權

- Bucket權限類別型

OSS提供ACL（Access Control List）許可權控制方法，OSS ACL提供Bucket等級的許可權存取控制，Bucket目前有三種存取權限：public-read-write，public-read和private，它們的含義如下：

許可權值	中文名稱	許可權對訪問者的限制
public-read-write	公共讀寫	任何人（包括匿名訪問）都可以對該Bucket中的Object進行讀/寫/刪除操作；所有這些操作產生的費用由該Bucket的Owner承擔，請慎用該許可權。
public-read	公共讀，私有寫	只有該Bucket的Owner或者授權對象可以對存放在其中的Object進行寫/刪除操作；任何人（包括匿名訪問）可以對Object進行讀操作。
private	私有讀寫	只有該Bucket的Owner或者授權對象可以對存放在其中的Object進行讀/寫/刪除操作；其他人在未經授權的情況下無法訪問該Bucket內的Object。

- Bucket許可權設定和讀取方法

功能使用參考：

- API: [Put BucketACL](#)
- SDK: Java SDK-[設定Bucket ACL](#)
- 控制台: [建立Bucket使用權限設定](#)
- API: [Get BucketACL](#)
- SDK: Java SDK-[獲取Bucket ACL](#)

Object等級許可權

· Object權限類別型

OSS ACL也提供Object等級的許可權存取控制。目前Object有四種存取權限：private, public-read, public-read-write, default。Put Object ACL操作通過Put請求中的“x-oss-object-acl”頭來設定，這個操作只有Bucket Owner有許可權執行。

許可權值	中文名稱	許可權對訪問者的限制
public-read-write	公共讀寫	該ACL表明某個Object是公共讀寫資源，即所有使用者擁有對該Object的讀寫權限。
public-read	公共讀，私有寫	該ACL表明某個Object是公共讀資源，即非Object Owner只有該Object的讀許可權，而Object Owner擁有該Object的讀寫權限。
private	私有讀寫	該ACL表明某個Object是私有資源，即只有該Object的Owner擁有該Object的讀寫權限，其他的使用者沒有許可權操作該Object。
default	預設許可權	該ACL表明某個Object是遵循Bucket讀寫權限的資源，即Bucket是什麼許可權，Object就是什麼許可權。



说明:

- 如果沒有設定Object的許可權，即Object的ACL為default，Object的許可權和Bucket許可權一致。
- 如果設定了Object的許可權，Object的許可權大於Bucket許可權。舉個例子，如果設定了Object的許可權是public-read，無論Bucket是什麼許可權，該Object都可以被身分識別驗證訪問和匿名訪問。

- Object許可權設定和讀取方法

功能使用參考：

- API: [Put Object ACL](#)
- SDK: Java SDK-[ObjectACL](#) 中設定Object ACL
- API: [Get Object ACL](#)
- SDK: Java SDK-[ObjectACL](#) 中讀取Object ACL

帳號等級許可權（RAM）

- 使用場景

如果您購買了雲資源，您的組織裡有多個使用者需要使用這些雲資源，這些使用者只能共用使用您的雲帳號AccessKey。這裡有兩個問題：

- 您的密鑰由多人共用，泄露的風險很高。
- 您無法控制特定使用者能訪問哪些資源（比如Bucket）的許可權。

解決方法：在您的阿里雲帳號下面，通過RAM可以建立具有自己AccessKey的子使用者。您的阿里雲帳號被稱為主帳號，建立出來的帳號被稱為子帳號，使用子帳號的AccessKey只能使用主帳號授權的操作和資源。

- 具體實現

有關RAM詳情，請參考[RAM使用者手冊](#)。

對於授權中需要的Policy的配置方式可以參考本章最後一節：RAM和STS授權策略（Policy）配置。

臨時帳號許可權（STS）

- 使用場景

對於您本地身份系統所管理的使用者，比如您的App的使用者、您的企業本地帳號、第三方App，也有直接存取OSS資源的可能，將這部分使用者稱為同盟使用者。此外，使用者還可以是您建立的能訪問您的阿里雲資源的應用程式。

對於這部分同盟使用者，通過阿里雲STS (Security Token Service) 服務為阿里雲帳號（或RAM使用者）提供短期存取權限管理。您不需要透露雲帳號（或RAM使用者）的長期密鑰（如登入密碼、AccessKey），只需要生成一個短期訪問憑證給同盟使用者使用即可。這個憑證的

存取權限及有效期間限都可以由您自訂。您不需要關心許可權撤銷問題，訪問憑證過期後會自動失效。

使用者通過STS生成的憑證包括安全性權杖(SecurityToken)、臨時存取金鑰(AccessKeyId, AccessKeySecret)。使用AccessKey方法與您在使用阿里雲帳戶或RAM使用者AccessKey發送請求時的方法相同。此外還需要注意的是在每個向OSS發送的請求中必須攜帶安全性權杖。

- 具體實現

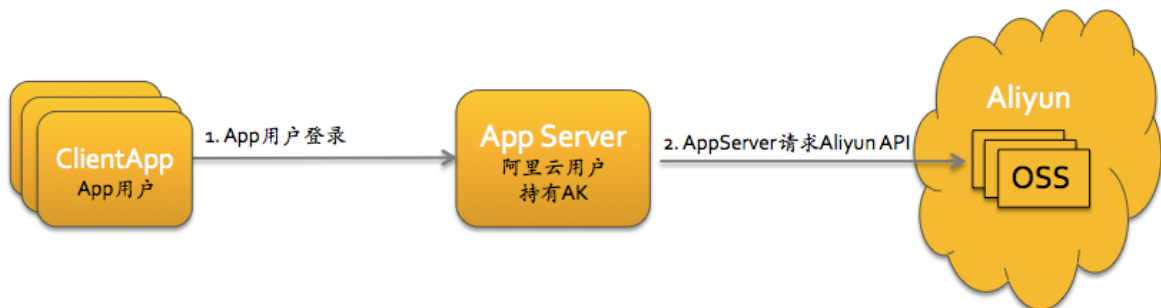
STS安全性權杖、角色管理和使用相關內容詳情，請參考RAM使用者指南中的[角色管理](#)。關鍵是調用STS服務介面AssumeRole來獲取有效訪問憑證即可，也可以直接使用STS SDK來調用該方法。

RAM和STS應用場景實踐

對於不同的應用場景，涉及到的訪問身分識別驗證方式可能存在差異。下面以幾種典型的應用場景來說明訪問身分識別驗證中幾種使用方式。

以一個移動App舉例。假設您是一個移動App開發人員，打算使用阿里雲OSS服務來保存App的終端使用者資料，並且要保證每個App使用者之間的資料隔離，防止一個App使用者獲取到其它App使用者的資料。

- 方式一：使用AppServer來做資料中轉和資料隔離



如上圖所示，您需要開發一個AppServer。只有AppServer能訪問雲服務，ClientApp的每次讀寫資料都需要通過AppServer，AppServer來保證不同使用者資料的隔離訪問。

對於該種使用方式，使用阿里雲帳號或者RAM帳號提供的密鑰來進行簽名驗證訪問。建議您盡量不要直接使用阿里雲帳號（主帳號）的密鑰訪問OSS，避免出現安全問題。

- 方式二：使用STS讓使用者直接存取OSS

STS方案描述如下圖所示：



方案的詳細描述如下：

1. App使用者登入。App使用者和雲帳號無關，它是App的終端使用者，AppServer支援App使用者登入。對於每個有效App使用者來說，需要AppServer能定義出每個App使用者的最小存取權限。
2. AppServer請求STS服務獲取一個安全性權杖（SecurityToken）。在調用STS之前，AppServer需要確定App使用者的最小存取權限（用Policy文法描述）以及授權的過期時間。然後通過扮演角色（AssumeRole）來獲取一個代表角色身份的安全性權杖。
3. STS返回給AppServer一個有效訪問憑證，包括一個安全性權杖（SecurityToken）、臨時存取金鑰（AccessKeyId, AccessKeySecret）以及過期時間。
4. AppServer將訪問憑證返回給ClientApp。ClientApp可以緩存這個憑證。當憑證失效時，ClientApp需要向AppServer申請新的有效訪問憑證。比如，訪問憑證有效期間為1小時，那麼ClientApp可以每30分鐘向AppServer請求更新訪問憑證。
5. ClientApp使用本機快取的訪問憑證去請求Aliyun Service API。雲服務會感知STS訪問憑證，並會依賴STS服務來驗證訪問憑證，正確響應使用者請求。

RAM和STS授權策略（Policy）配置

對於RAM或者STS授權中使用Policy，詳細規則如下。

- 樣本

先看下面的一個Policy樣本：

```

{
  "Version": " 1 ",
  "Statement": [
    {
      "Action": [
        "oss : GetBucketA  cl ",
        "oss : ListObject  s "
      ],
      "Resource": [

```

```

        "acs : oss :*: 1775305056 529849 : mybucket "
      ],
      "Effect ": " Allow ",
      "Condition ": {
        "StringEquals ": {
          "acs : UserAgent ": " java - sdk ",
          "oss : Prefix ": " foo "
        },
        "IpAddress ": {
          "acs : SourceIp ": " 192 . 168 . 0 . 1 "
        }
      }
    },
    {
      "Action ": [
        "oss : PutObject ",
        "oss : GetObject ",
        "oss : DeleteObject "
      ],
      "Resource ": [
        "acs : oss :*: 1775305056 529849 : mybucket / file
* "
      ],
      "Effect ": " Allow ",
      "Condition ": {
        "IpAddress ": {
          "acs : SourceIp ": " 192 . 168 . 0 . 1 "
        }
      }
    }
  ]
}

```

這是一個授權的Policy，使用者用這樣的一個Policy通過RAM或STS服務向其他使用者授權。Policy當中有一個Statement（一條Policy當中可以有多條Statement）。Statement裡面規定了相應的Action、Resource、Effect和Condition。

這條Policy把使用者自己名下的 `mybucket` 和 `mybucket / file *`這些資源授權給相應的使用者，並且支援GetBucketAcl、GetBucket、PutObject、GetObject和DeleteObject這幾種操作。Condition中的條件表示UserAgent為java-sdk，源IP為192.168.0.1的時候鑒權才能通過，被授權的使用者才能訪問相關的資源。Prefix這個Condition是在GetBucket（ListObjects）的時候起作用的，關於這個欄位的解釋詳見OSS的API文檔。

· 配置細則

- Version

Version定義了Policy的版本，本文檔中sw2q的配置方式，設定為 1 。

- Statement

通過Statement描述授權語義，其中可以根據業務場景包含多條語義，每條包含對Action、Effect、Resource和Condition的描述。每次請求系統會逐條依次匹配檢查，所有匹配成功的Statement會根據Effect的設定不同分為通過（Allow）、禁止（Deny），其中禁止

(Deny) 的優先。如果匹配成功的都為通過，該條請求即鑒權通過。如果匹配成功有一條禁止，或者沒有任何條目匹配成功，該條請求被禁止訪問。

- Action

Action分為三大類：Service等級操作，對應的是GetService操作，用來列出所有屬於該使用者的Bucket列表。

■ Bucket等級操作，對應類似於oss:PutBucketAcl、oss:GetBucketLocation之類的操作，操作的對象是Bucket，它們的名稱和相應的介面名稱一一對應。

■ Object等級操作，分為oss:GetObject、oss:PutObject、oss:DeleteObject和oss:AbortMultipartUpload，操作對象是Object。

如想授權某一類的Object的操作，可以選擇這幾種的一種或幾種。另外，所有的Action前面都必須加上 `oss :`，如上面例子所示。Action是一個列表，可以有多个Action。具體的Action和API介面的對應關係如下：

■ Service等級

API	Action
GetService (ListBuckets)	oss:ListBuckets

■ Bucket等級

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

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

■ Object等級

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

API	Action
GetObjectAcl	oss:GetObjectAcl
PutObjectAcl	oss:PutObjectAcl

- Resource

Resource指代的是OSS上面的某個具體的資源或者某些資源（支援*通配），resource的規則是 `acs : oss :{ region }:{ bucket_owner }:{ bucket_name }/{ object_name }`。對於所有Bucket等級的操作來說不需要最後的斜杠和{object_name}，即 `acs : oss :{ region }:{ bucket_owner }:{ bucket_name }`。Resource也是一個列表，可以有多個Resource。其中的region欄位暫時不做支援，設定為*。

- Effect

Effect代表本條的Statement的授權的結果，分為Allow和Deny，分別指代通過和禁止。多條Statement同時匹配成功時，禁止（Deny）的優先順序更高。

例如，期望禁止使用者對某一目錄進行刪除，但對於其他檔案有全部許可權：

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

```
}

```

- Condition

Condition代表Policy授權的一些條件，上面的樣本裡面可以設定對於acs:UserAgent的檢查、acs:SourceIp的檢查、還有oss:Prefix這項用來在GetBucket的時候對資源進行限制。

OSS支援的Condition如下：

condition	功能	合法取值
acs:SourceIp	指定ip網段	普通的ip，支援*通配
acs:UserAgent	指定http useragent頭	字元串
acs:CurrentTime	指定合法的訪問時間	ISO8601格式
acs:SecureTransport	是否是https協議	“true” 或者 “false”
oss:Prefix	用作ListObjects時的prefix	合法的object name

更多樣本

針對具體場景更多的授權策略配置樣本，可以參考[教程樣本：控制儲存空間和檔案夾的存取權限和OSS授權常見問題](#)。

Policy線上圖形化便捷組態工具，請單擊[這裡](#)。

最佳實務

- [RAM和STS使用指南](#)

9.2 基於RAM Policy的許可權控制

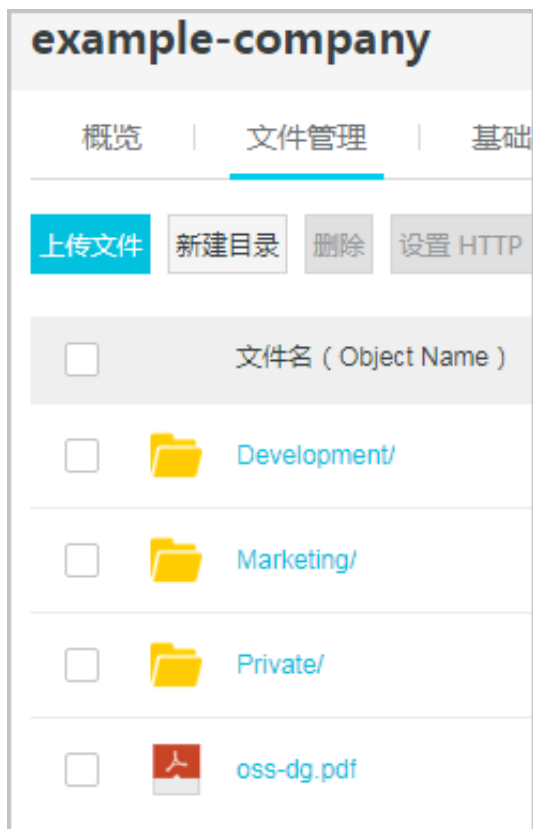
9.2.1 教程樣本：使用RAM Policy控制儲存空間和檔案夾的存取權限

本教程樣本詳細示範了如何使用 RAM Policy 控制使用者對 OSS 儲存空間和檔案夾的訪問。

在樣本中，我們首先建立一個儲存空間和檔案夾，然後使用阿里雲主帳號建立訪問管理（RAM）使用者，並通過構建不同的 RAM Policy 為這些使用者授予對所建立 OSS 儲存空間及檔案夾的增量許可權。

儲存空間和檔案夾的基本概念

阿里雲 OSS 的資料模型為扁平型結構，所有檔案都直接隸屬於其對應的儲存空間。因此，OSS 缺少檔案系統中類似於目錄與子檔案夾的階層。但是，您可以在 OSS 控制台上類比資料夾階層。在該控制台中，您可以按檔案夾對相關檔案進行分組、分類和管理，如下圖所示。



OSS 提供使用索引值（key）對格式的分布式Object Storage Service服務。使用者根據其唯一的key（對象名）檢索對象的內容。例如，名為 example-company 的儲存空間有三個檔案夾：*Development* # *Marketing* 和 *Private*，以及一個對象 *oss - dg . pdf*。

- 在建立 *Development* 檔案夾時，控制台會建立一個 key 為 *Development /* 的對象。注意，檔案夾的 key 包括分隔字元 */*。

- 當您將名為 `ProjectA.docx` 的對象上傳到 `Development` 檔案夾中時，控制台會上傳該對象並將其key設定為 `Development / ProjectA.docx`。

在該key中，`Development` 為首碼，而`/`為分隔字元。您可以從儲存空間中擷取具有特定首碼和分隔字元的所有對象的列表。在控制台中，單擊 `Development` 檔案夾時，控制台會列出檔案夾中的對象，如下圖所示。



说明:

當控制台列出 `example-company` 儲存空間中的 `Development` 檔案夾時，它會向 OSS 發送一個用於指定首碼 `Development` 和分隔字元`/`的請求。控制台的響應與檔案系統類似，會顯示資料夾清單。上例說明，儲存空間 `example-company` 有三個對象，其 key 分別為 `Development / Alibaba Cloud.pdf`、`Development / ProjectA.docx` 及 `Development / ProjectB.docx`。

控制台通過對象的 key 推斷邏輯階層。當您建立對象的邏輯階層時，您可以管理對個別檔案夾的訪問，如本教程後面描述的那樣。

在本教程開始之前，您還需要知道“根級”儲存空間內容的概念。假設 `example-company` 儲存空間包含以下對象：

- `Development/Alibaba Cloud.pdf`
- `Development/ProjectA.docx`

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

這些對象的key構建了一個以 `Development`、`Marketing` 和 `Private` 作為根級檔案夾並以 `oss-dg.pdf` 作為根級對象的邏輯階層。當您單擊 OSS 控制台中的儲存空間名時，控制台會將一級首碼和一個分隔字元（`Development/`、`Marketing/` 和 `Private/`）顯示為根級檔案夾。對象 `oss-dg.pdf` 沒有首碼，因此顯示為根層級項。



OSS 的請求和響應邏輯

在授予許可權之前，我們需要清楚，當使用者單擊某個儲存空間的名字時控制台向 OSS 發送的是什麼請求、OSS 返回的是什麼響應，以及控制台如何解析該響應。

當使用者單擊某個儲存空間名時，控制台會將 `GetBucket` 請求發送至 OSS。此請求包括以下參數：

- `prefix`，其值為空白字串。
- `delimiter`，其值為 `/`。

請求樣本如下所示：

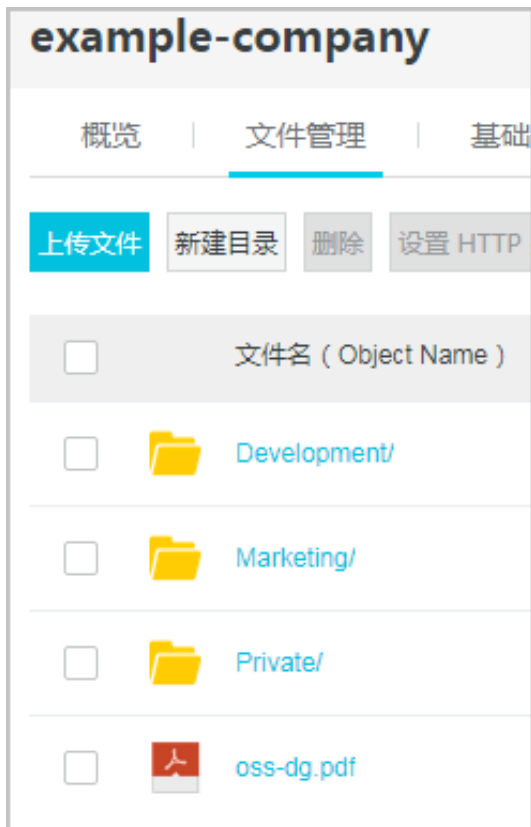
```
GET /? prefix =& delimiter =/ HTTP / 1 . 1
Host : example - company . oss - cn - hangzhou . aliyuncs . com
Date : Fri , 24 Feb 2012 08 : 43 : 27 GMT
Authorization : OSS qn6qrrqxo2 oawuk53otf jbyc : DNrnX7xHk3
sgysx7I8U9 I9IY1vY =
```

OSS 返回的響應包括 `ListBucket Result` 元素：

```
HTTP / 1 . 1 200 OK
x - oss - request - id : 534B371674 E88A4D8906 008B
Date : Fri , 24 Feb 2012 08 : 43 : 27 GMT
Content - Type : applicatio n / xml
Content - Length : 712
Connection : keep - alive
Server : AliyunOSS
<? xml version =" 1 . 0 " encoding =" UTF - 8 ">
< ListBucket Result xmlns =± http :// doc . oss - cn - hangzhou .
  aliyuncs . com ±>
  < Name > example - company </ Name >
  < Prefix ></ Prefix >
  < Marker ></ Marker >
  < MaxKeys > 100 </ MaxKeys >
  < Delimiter ></ Delimiter >
    < IsTruncate d > false </ IsTruncate d >
    < Contents >
      < Key > oss - dg . pdf </ Key >
      ...
    </ Contents >
    < CommonPref ixes >
      < Prefix > Developmen t </ Prefix >
    </ CommonPref ixes >
      < CommonPref ixes >
        < Prefix > Marketing </ Prefix >
      </ CommonPref ixes >
      < CommonPref ixes >
        < Prefix > Private </ Prefix >
      </ CommonPref ixes >
    </ ListBucket Result >
```

由於 `oss - dg . pdf` 不包含/分隔字元，因此 OSS 在 `< Contents />` 元素中返回該 key。儲存空間 `example-company` 中的所有其他 key 都包含/分隔字元，因此 OSS 會將這些 key 分組，並為每個首碼值 `Developmen t /# Marketing /` 和 `Private /` 返回一個 `< CommonPref ixes />` 元素。該元素是一個字串，包含從這些 key 的第一個字元開始到第一次出現指定的/分隔字元之間的字元。

控制台會解析此結果並顯示如下的根層級項：



現在，如果使用者單擊 *Development* 檔案夾，控制台會將 [GetBucket](#) 請求發送至 OSS。此請求包括以下參數：

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

請求樣本如下所示：

```
GET /? prefix = Development/& delimiter =/ HTTP / 1 . 1
Host : oss - example . oss - cn - hangzhou . aliyuncs . com
Date : Fri , 24 Feb 2012 08 : 43 : 27 GMT
Authorization : OSS qn6qrrqxo2 oawuk53otf jbyc : DNrnX7xHk3
sgysx7I8U9 I9IY1vY =
```

作為響應，OSS 返回以指定首碼開頭的 `key`：

```
HTTP / 1 . 1 200 OK
x - oss - request - id : 534B371674 E88A4D8906 008B
Date : Fri , 24 Feb 2012 08 : 43 : 27 GMT
Content - Type : application / xml
Content - Length : 712
Connection : keep - alive
Server : AliyunOSS
<? xml version =" 1 . 0 " encoding =" UTF - 8 ">
< ListBucket Result xmlns =;± http :// doc . oss - cn - hangzhou .
aliyuncs . com ;±>
< Name > example - company </ Name >
< Prefix > Development /</ Prefix >
< Marker ></ Marker >
< MaxKeys > 100 </ MaxKeys >
```

```
< Delimiter >/</ Delimiter >
  < IsTruncate d > false </ IsTruncate d >
  < Contents >
    < Key > ProjectA . docx </ Key >
    ...
  </ Contents >
  < Contents >
    < Key > ProjectB . docx </ Key >
    ...
  </ Contents >
</ ListBucket Result >
```

控制台會解析此結果並顯示如下的 key：



教程樣本

本教程樣本如下所示：

- 建立一個儲存空間 **example-company**，然後向其中添加三個檔案夾（*Development*、*Marketing* 和 *Private*）。
- 您有 Anne 和 Leo 兩個使用者。您希望 Anne 只能訪問 *Development* 檔案夾而 Leo 則只能訪問 *Marketing* 檔案夾，並且希望將 *Private* 檔案夾保持私人。在教程樣本中，通過建立存取控制（RAM）使用者（Anne 和 Leo）來管理存取權限，並授予他們必要的許可權。
- RAM 還支援建立使用者組並授予適用於組中所有使用者的組層級許可權。這有助於更好地系統管理權限。在本樣本中，Anne 和 Leo 都需要一些公用許可權。因此，您還要建立一個名為

Staff 的組，然後將 Anne 和 Leo 添加到該組中。首先，您需要給該組分配策略授予許可權。然後，將策略分配給特定使用者，添加特定使用者的許可權。

**说明:**

本教程樣本使用 `example-company` 作為儲存空間名、使用 Anne 和 Leo 作為 RAM 使用者名稱並使用 Staff 作為組名。由於阿里雲 OSS 要求儲存空間名全域唯一，所以您需要用自己的儲存空間名稱替換本教程中的儲存空間名。

樣本準備

本樣本使用阿里雲主帳號建立 RAM 使用者。最初，這些使用者沒有任何許可權。您將逐步授予這些使用者執行特定 OSS 操作的許可權。為了測試這些許可權，您需要使用每個使用者的 RAM 帳號登入到控制台。當您作為主帳號所有者逐步授予許可權並作為 RAM 使用者測試許可權時，您需要每次使用不同帳號進行登入和登出。您可以使用一個瀏覽器來執行此測試。如果您可以使用兩個不同的瀏覽器，則該測試過程用時將會縮短：一個瀏覽器用於使用主帳號串連到阿里雲控制台，另一個瀏覽器用於使用 RAM 帳號進行串連。

要使用您的主帳號登入到阿里雲控制台。RAM 使用者不能使用相同的連結登入。他們必須使用 RAM 使用者登入連結。作為主帳號所有者，您可以向 RAM 使用者提供此連結。

**说明:**

有關 RAM 的詳細資料，請參見[使用 RAM 使用者帳號登入](#)。

為 RAM 使用者提供登入連結：

1. 使用主帳號登入 [RAM 控制台](#)。
2. 在左側導覽列中，單擊概覽。
3. 在帳號管理地區，將 RAM 使用者登入地址的 URL 提供給 RAM 使用者，以便其使用 RAM 使用者名稱和密碼登入控制台。

步驟 1：建立儲存空間

在此步驟中，您可以使用主帳號登入到 OSS 控制台、建立儲存空間、將檔案夾（`Development # Marketing # Private`）添加到儲存空間中，並在每個檔案夾中上傳一個或兩個範例文件。

1. 使用主帳號登入 [OSS 控制台](#)。
2. 建立名為 `example-company` 的儲存空間。

有關詳細過程，請參見 OSS 控制台使用者指南中的[建立儲存空間](#)。

3. 將一個檔案上傳到儲存空間中。

本樣本假設您將檔案 `oss - dg . pdf` 上傳到儲存空間的根層級。您可以用不同的檔案名稱上傳自己的檔案。

有關詳細過程，請參見 OSS 控制台使用者指南中的[上傳檔案](#)。

4. 添加名為 `Development # Marketing` 和 `Private` 的三個檔案夾。

有關詳細過程，請參見 OSS 控制台使用者指南 中的[建立檔案夾](#)。

5. 將一個或兩個檔案上傳到每個檔案夾中。

本例假設您將具有以下對象鍵的對象上傳到儲存空間中：

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

步驟 2：建立 RAM 使用者和組

在此步驟中，您使用 RAM 控制台將兩個 RAM 使用者 Anne 和 Leo 添加到主帳號中。您還將建立一個名為 `Staff` 的組，然後將這兩個使用者添加到該組中。



说明：

在此步驟中，不要分配任何授予這些使用者權限的策略。在以下步驟中，您將逐步為其授予許可權。

有關建立 RAM 使用者的詳細過程，請參見 RAM 快速入門中的[建立 RAM 使用者](#)。請為每個 RAM 使用者建立登入密碼。

有關建立組的詳細過程，請參見 RAM 使用者指南中的[建立組](#)。

步驟 3：確認 RAM 使用者沒有任何許可權

如果您使用兩個瀏覽器，現在可以在另一個瀏覽器中使用其中一個 RAM 使用者帳號登入到控制台。

1. 開啟 RAM 使用者登入連結，並用 Anne 或 Leo 的帳號登入到 RAM 控制台。

2. 開啟 OSS 控制台。

您發現控制台中沒有任何儲存空間，這意味著 Anne 不具有對儲存空間 example-company 的任何許可權。

步驟 4: 授予組層級許可權

我們希望 Anne 和 Leo 都能執行以下操作：

- 列出主帳號所擁有的所有儲存空間。

為此，Anne 和 Leo 必須具有執行 `oss : ListBucket s` 操作的許可權。

- 列出 example-company 儲存空間中的根層級項、檔案夾和對象。

為此，Anne 和 Leo 必須具有對 example-company 儲存空間執行 `oss : ListObject s` 操作的許可權。

步驟 4.1. 授予列出所有儲存空間的許可權

在此步驟中，建立一個授予使用者最低許可權的策略。憑藉最低許可權，使用者可列出主帳號所擁有的所有儲存空間。您還將此策略分配給 Staff 組，以便授予獲得主帳號擁有的儲存空間列表的組許可權。

1. 使用主帳號登入 [RAM 控制台](#)。
2. 建立策略 AllowGroupToSeeBucketListInConsole。
 - a. 在左側瀏覽窗格中，單擊許可權管理 > 權限原則管理 > 建立權限原則。
 - b. 在策略名稱欄位中，輸入 AllowGroupToSeeBucketListInConsole。
 - c. 選擇配置模式為指令碼配置。
 - d. 在策略內容欄位中，複製並粘貼以下策略。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Effect ": " Allow ",
      " Action ": [
        " oss : ListBucket s ",
        " oss : GetBucketS tat ",
        " oss : GetBucketI nfo ",
        " oss : GetBucketA cl "
      ],
      " Resource ": [
        " acs : oss :*:~:*"
      ]
    }
  ]
}
```

```
}
```



说明:

- RAM 策略為 JSON 格式。各欄位定義如下：
 - Statement: 對象數組，每個對象使用名/值對的集合來描述許可權。
 - Effect: 決定是允許還是拒絕特定的許可權。
 - Action: 指定存取權限的類型。在本策略中，`oss : ListBucket s` 是預定義的 OSS 操作，可返回經過身分識別驗證的寄件者所擁有的所有儲存空間的列表。
- 推薦您使用 [RAM 策略編輯器](#) 快速產生 RAM 策略。更詳細的 RAM 策略介紹請參見 [如何構建 RAM Policy](#)。

3. 將 `AllowGroup ToSeeBucke tListInCon sole` 策略分配給 Staff 組。

有關分配策略的詳細過程，請參見 RAM 快速入門為 [RAM 使用者授權](#) 中的“為使用者所屬的使用者組授權”。

可以將策略分配給 RAM 控制台中的 RAM 使用者和組。在本例中，我們將策略分配給組，因為我們希望 Anne 和 Leo 都能夠列出這些儲存空間。

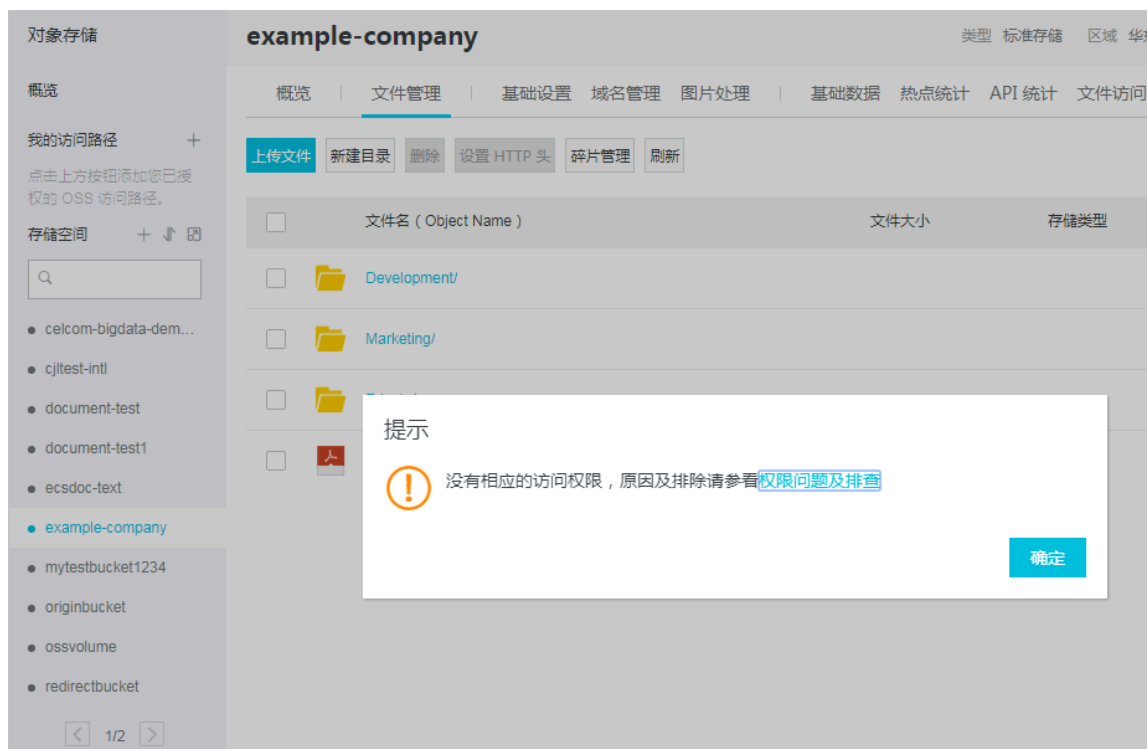
4. 測試許可權。

- 開啟 RAM 使用者登入連結，並用 Anne 或 Leo 的帳號登入到 RAM 控制台。
- 開啟 OSS 控制台。

控制台列出所有儲存空間。

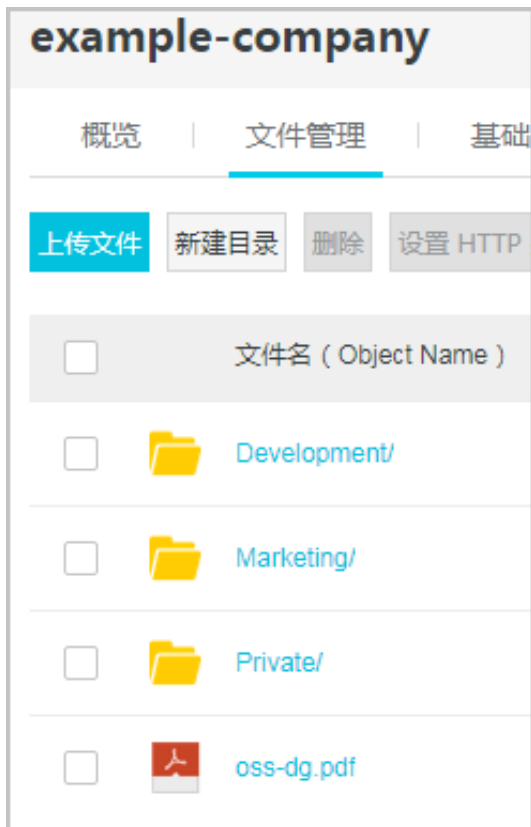
- 單擊 example-company 儲存空間，然後單擊檔案選項卡。

此時將顯示一個訊息框，表明您沒有相應的存取權限。



步驟 4.2. 授予列出儲存空間根級內容的許可權

在此步驟中，您授予許可權，允許所有使用者列出儲存空間 example-company 中的所有項目。當使用者在 OSS 控制台中單擊 example-company 時，能夠看到儲存空間中的根層級項。



1. 使用主帳號登入[RAM 控制台](#)。
2. 用以下策略取代分配給 Staff 組的現有策略 `AllowGroup ToSeeBuckets ListInContents`，該策略還允許 `oss : ListObjects` 操作。請用您的儲存空間名替換策略資源中的 `example-company`。

有關詳細過程，請參見 RAM 使用者指南中[授權策略](#)的“修改自訂授權策略”部分。注意，您最多可對 RAM 策略進行五次修改。如果超過了五次，則需要刪除該策略並建立一個新的策略，然後再次將新策略分配給 Staff 組。

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

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

**说明:**

- 要列出儲存空間內容，使用者需要調用 `oss : ListObject s` 操作的許可權。為了確保使用者僅看到根級內容，我們添加了一個條件：使用者必須在請求中指定一個空首碼，也就是說，他們不能單擊任何根級檔案夾。我們還通過要求使用者請求包含分隔字元參數和值 `/` 來添加需要檔案夾樣式訪問的條件。
- 當使用者登入到 OSS 控制台時，控制台檢查使用者的身份是否有訪問 OSS 服務的許可權。要在控制台中支援儲存空間操作，我們還需要添加 `oss : GetBucketA cl` 操作。

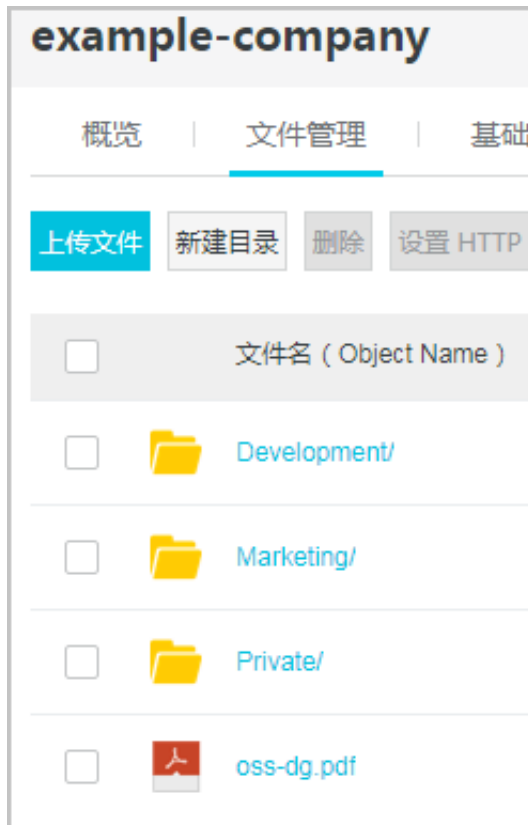
3. 測試更新的許可權。

- a. 開啟 RAM 使用者登入連結，並用 Anne 或 Leo 的帳號登入到 RAM 控制台。
- b. 開啟 OSS 控制台。

控制台列出所有儲存空間。

- c. 單擊 example-company 儲存空間，然後單擊檔案選項卡。

控制台列出所有根層級項。



- d. 單擊任何檔案夾或對象 `oss - dg . pdf`。

此時將顯示一個訊息框，表明您沒有相應的存取權限。

組策略摘要

添加組策略的最終結果是授予 RAM 使用者 Anne 和 Leo 以下最低許可權：

- 列出主帳號所擁有的所有儲存空間。
- 查看 example-company 儲存空間中的根層級項。

然而，他們可以進行的操作仍然有限。在以下部分中，我們將授予使用者以下特定許可權：

- 允許 Anne 在 `Development` 檔案夾中擷取和放入對象。
- 允許 Leo 在 `Finance` 檔案夾中擷取和放入對象。

對於使用者特定的許可權，您需要將策略分配給特定使用者，而非分配給組。以下部分授予 Anne 在 *Development* 檔案夾中操作的許可權。您可以重複這些步驟，授予 Leo 在 *Finance* 檔案夾中進行類似操作的許可權。

步驟 5：授予 RAM 使用者 Anne 特定許可權

在此步驟中，我們向 Anne 授予額外的許可權，使她可以看到 *Development* 檔案夾的內容，並將對象放入檔案夾中。

步驟 5.1. 授予 RAM 使用者 Anne 許可權以列出 *Development* 檔案夾內容

若要 Anne 能夠列出 *Development* 檔案夾內容，您必須為其分配策略。該策略必須能夠授予其對 `example-company` 儲存空間執行 `oss : ListObjects` 操作的許可權，還必須包括要求使用者在請求中指定首碼 `Development /` 的條件。

1. 使用主帳號登入 [RAM 控制台](#)。
2. 建立策略 `AllowListBucketsIfSpecificPrefixIsIncluded`，授予 RAM 使用者 Anne 許可權以列出 *Development* 檔案夾內容。
 - a. 在左側瀏覽窗格中，單擊許可權管理 > 權限原則管理 > 建立權限原則。
 - b. 在策略名稱欄位中，輸入 `AllowGroupToSeeBucketListInConsole`。
 - c. 選擇配置模式為指令碼配置。
 - d. 在策略內容欄位中，複製並粘貼以下策略。

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

3. 將策略分配給 RAM 使用者 Anne。

有關分配策略的詳細過程，請參見 RAM 快速入門中的 [為 RAM 使用者授權](#)。

4. 測試 Anne 的許可權。

- 開啟 RAM 使用者登入連結，並用 Anne 的帳號登入到 RAM 控制台。
- 開啟 OSS 控制台，控制台列出所有儲存空間。
- 單擊 `example-company` 儲存空間，然後單擊檔案選項卡，控制台列出所有根層級項。
- 單擊 `Development` / 檔案夾。控制台列出檔案夾中的對象。

步驟 5.2 授予 RAM 使用者 Anne 在 `Development` 檔案夾中擷取和放入對象的許可權。

若要 Anne 能夠在 `Development` 檔案夾中擷取和放入對象，您必須授予她調用 `oss : GetObject` 和 `oss : PutObject` 操作的許可權，包括使用者必須在請求中指定首碼 `Development /` 的條件。

- 使用主帳號登入 [RAM 控制台](#)。
- 用以下策略取代您在之前步驟中建立的策略 `AllowListBucketsIfSpecificPrefixIsIncluded`。

有關詳細過程，請參見 RAM 使用者指南中[授權策略](#)的“修改自訂授權策略”部分。注意，您最多可對 RAM 策略進行五次修改。如果超過了五次，則需要刪除該策略並建立一個新的策略，然後再次將新策略分配給 Staff 組。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Effect ": " Allow ",
      " Action ": [
        " oss : ListObject s "
      ],
      " Resource ": [
        " acs : oss :*: *: example - company "
      ],
      " Condition ": {
        " StringLike ": {
          " oss : Prefix ": [
            " Developmen t /*"
          ]
        }
      }
    },
    {
      " Effect ": " Allow ",
      " Action ": [
        " oss : GetObject ",
        " oss : PutObject ",
        " oss : GetObjectA ct "
      ],
      " Resource ": [
        " acs : oss :*: *: example - company / Developmen t /*"
      ],
      " Condition ": {}
    }
  ]
}
```

```
}
```

**说明:**

當使用者登入到 OSS 控制台時，控制台檢查使用者的身份是否有訪問 OSS 服務的許可權。要在控制台中支援儲存空間操作，我們還需要添加 `oss : GetObjectACL` 操作。

3. 測試更新的策略。

- a. 開啟 RAM 使用者登入連結，並用 Anne 的帳號登入到 RAM 控制台。
- b. 開啟 OSS 控制台。

控制台列出所有儲存空間。

- c. 在 OSS 控制台中，確認 Anne 現在可以在 `Development` 檔案夾中添加對象並下載對象。

步驟 5.3 顯式拒絕 RAM 使用者 Anne 訪問儲存空間中任何其他檔案夾的許可權

RAM 使用者 Anne 現在可以在 `example-company` 儲存空間中列出根級內容，並將對象放入 `Development` 檔案夾中。如果要嚴格限制存取權限，您可以顯式拒絕 Anne 對儲存空間中任何其他檔案夾的訪問。如果有授予 Anne 訪問儲存空間中任何其他檔案夾的其他策略，則此顯式策略將替代這些許可權。

您可以將以下語句添加到 RAM 使用者 Anne 的策略 `AllowListBucketsIfSpecificPrefixIsIncluded`，以要求 Anne 發送到 OSS 的所有請求包含首碼參數，該參數的值可以是 `Development/*` 或 `Null` 字元串。

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

按照之前的步驟更新您為 RAM 使用者 Anne 建立的策略 `AllowListBucketsIfSpecificPrefixIsIncluded`。複製並粘貼以下策略以替換現有策略。

```
{
  "Version": "1",
```

```

    " Statement ": [
      {
        " Effect ": " Allow ",
        " Action ": [
          " oss : ListObject s "
        ],
        " Resource ": [
          " acs : oss :*: *: example - company "
        ],
        " Condition ": {
          " StringLike ": {
            " oss : Prefix ": [
              " Developmen t /*"
            ]
          }
        }
      },
      {
        " Effect ": " Allow ",
        " Action ": [
          " oss : GetObject ",
          " oss : PutObject ",
          " oss : GetObjectA ct "
        ],
        " Resource ": [
          " acs : oss :*: *: example - company / Developmen t /*"
        ],
        " Condition ": {}
      },
      {
        " Effect ": " Deny ",
        " Action ": [
          " oss : ListObject s "
        ],
        " Resource ": [
          " acs : oss :*: *: example - company "
        ],
        " Condition ": {
          " StringNotL ike ": {
            " oss : Prefix ": [
              " Developmen t /*",
              ""
            ]
          }
        }
      }
    ]
  }
}

```

步驟 6：授予 RAM 使用者 Leo 特定許可權

現在，您希望授予 Leo 訪問 *Marketing* 檔案夾的許可權。請遵循之前用於向 Anne 授予許可權的步驟，但應將 *Developmen t* 檔案夾替換為 *Marketing* 檔案夾。有關詳細過程，請參見步驟 5：授予 RAM 使用者 Anne 特定許可權。

步驟 7：確保 Private 檔案夾安全

在本例中，您僅擁有兩個使用者。您在組層級授予兩個使用者所有所需的最小許可權，只有當您真正需要單個使用者層級上的許可權時，才授予使用者層級許可權。此方法有助於最大限度地減少系

統管理權限的工作量。隨著使用者數量的增加，我們希望確保不意外地授予使用者對 *Private* 檔案夾的許可權。因此，我們需要添加一個顯式拒絕訪問 *Private* 檔案夾的策略。顯式拒絕策略會取代任何其他許可權。若要確保 *Private* 檔案夾保持私人，可以向組策略添加以下兩個拒絕語句：

- 添加以下語句以顯式拒絕對 *Private* 檔案夾(*example - company / Private /**) 中的資源執行任何操作。

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

- 您還要在請求指定了 *Private / prefix* 時拒絕執行 `ListObject s` 操作的許可權。在控制台中，如果 Anne 或 Leo 單擊 *Private* 檔案夾，則此策略將導致 OSS 返回錯誤響應。

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

- 用包含前述拒絕語句的更新策略取代 Staff 組策略 `AllowGroup ToSeeBucke tListInCon sole`。在應用程式更新策略後，組中的任何使用者都不能訪問您的儲存空間中的 *Private* 檔案夾。

1. 使用主帳號登入[RAM 控制台](#)。

2. 用以下策略取代分配給 Staff 組的現有策略 `AllowGroup ToSeeBucke tListInCon sole`。請用您的儲存空間名替換策略資源中的 `example-company`。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
```

```

    " Effect ": " Allow ",
    " Action ": [
      " oss : ListBucket s ",
      " oss : GetBucketS tat ",
      " oss : GetBucketI nfo ",
      " oss : GetBucketA cl "
    ],
    " Resource ": [
      " acs : oss :*:~:*"
    ],
    " Condition ": {}
  },
  {
    " Effect ": " Allow ",
    " Action ": [
      " oss : ListObject s "
    ],
    " Resource ": [
      " acs : oss :*:~:*: example - company "
    ],
    " Condition ": {
      " StringLike ": {
        " oss : Prefix ": [
          ""
        ],
        " oss : Delimiter ": [
          "/"
        ]
      }
    }
  },
  {
    " Effect ": " Deny ",
    " Action ": [
      " oss :*"
    ],
    " Resource ": [
      " acs : oss :*:~:*: example - company / Private /*"
    ],
    " Condition ": {}
  },
  {
    " Effect ": " Deny ",
    " Action ": [
      " oss : ListObject s "
    ],
    " Resource ": [
      " acs : oss :*:~:*"
    ],
    " Condition ": {
      " StringLike ": {
        " oss : Prefix ": [
          " Private /"
        ]
      }
    }
  }
]
}

```

清理

要進行清理，您需要在 RAM 控制台中刪除使用者 Anne 和 Leo。

有關詳細過程，請參見 RAM 使用者指南中[使用者](#)的“刪除 RAM 使用者”部分。

為了確保您不再因儲存而繼續被收取費用，您還需要刪除為本樣本建立的對象和儲存空間。

9.2.2 如何構建RAM Policy

RAM（Resource Access Management）是阿里雲提供的資源存取控制服務。RAM Policy是基於使用者的授權策略。通過設定RAM Policy，您可以集中管理您的使用者（比如員工、系統或應用程式），以及控制使用者可以訪問您名下哪些資源的許可權。比如能夠限制您的使用者只擁有對某一個 Bucket 的讀許可權。子帳號是從屬於主帳號的，並且這些帳號下不能擁有實際的任何資源，所有資源都屬於主帳號。



说明:

您可以使用[RAM策略編輯器](#)快速產生RAM Policy。

常見Policy樣本

· 完全授權的Policy

完全授權的Policy表示允許應用可以對OSS進行任何操作。



警告:

完全授權的Policy對行動裝置 App來說是不安全的授權，不推薦使用。

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

對OSS的操作	結果
列舉所有建立的Bucket	成功
上傳不帶首碼的Object, test.txt	成功
下載不帶首碼的Object, test.txt	成功
上傳帶首碼的Object, user1/test.txt	成功
下載帶首碼的Object, user1/test.txt	成功
列舉Object, test.txt	成功

對OSS的操作	結果
帶首碼的Object, user1/test.txt	成功

- 不限制首碼的唯讀不寫Policy

此Policy表示應用對Bucket `app - base - oss` 下所有的Object可列舉，可下載。

```
{
  "Statement": [
    {
      "Action": [
        "oss : GetObject ",
        "oss : ListObject s "
      ],
      "Effect": " Allow ",
      "Resource": ["acs : oss ::*: app - base - oss /*", "acs : oss ::*: app - base - oss "]
    }
  ],
  "Version": " 1 "
}
```

對OSS的操作	結果
列舉所有建立的Bucket	失敗
上傳不帶首碼的Object, test.txt	失敗
下載不帶首碼的Object, test.txt	成功
上傳帶首碼的Object, user1/test.txt	失敗
下載帶首碼的Object, user1/test.txt	成功
列舉不帶首碼的Object, test.txt	成功
列舉帶首碼的Object, user1/test.txt	成功

- 限制首碼的唯讀不寫Policy

此Policy表示應用對Bucket `app - base - oss` 下帶有首碼 `user1 /`的Object可列舉、可下載，但無法下載其他首碼的Object。採用此種Policy，如果不同的應用對應不同的首碼，就可以達到在同一個Bucket中空間隔離的效果。

```
{
  "Statement": [
    {
      "Action": [
        "oss : GetObject ",
        "oss : ListObject s "
      ],
      "Effect": " Allow ",
      "Resource": ["acs : oss ::*: app - base - oss / user1 / *", "acs : oss ::*: app - base - oss "]
    }
  ],
  "Version": " 1 "
}
```

```
}

```

對OSS的操作	結果
列舉所有建立的Bucket	失敗
上傳不帶首碼的Object, test.txt	失敗
下載不帶首碼的Object, test.txt	失敗
上傳帶首碼的Object, user1/test.txt	失敗
下載帶首碼的Object, user1/test.txt	成功
列舉Object, test.txt	成功
帶首碼的Object, user1/test.txt	成功

- 不限制首碼的唯寫不讀Policy

此Policy表示應用可以對Bucket `app - base - oss` 下所有的Object進行上傳。

```
{
  "Statement": [
    {
      "Action": [
        "oss:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app - base - oss /*", "acs:oss:*:*:app - base - oss "]
    }
  ],
  "Version": "1"
}
```

對OSS的操作	結果
列舉所有建立的Bucket	失敗
上傳不帶首碼的Object, test.txt	成功
下載不帶首碼的Object, test.txt	失敗
上傳帶首碼的Object, user1/test.txt	成功
下載帶首碼的Object, user1/test.txt	成功
列舉Object, test.txt	成功
帶首碼的Object, user1/test.txt	成功

- 限制首碼的唯寫不讀Policy

此Policy表示應用可以對Bucket `app - base - oss` 下帶有首碼 `user1 /`的Object進行上傳。但無法上傳其他首碼的Object。採用此種Policy，如果不同的應用對應不同的首碼，就可以達到在同一個Bucket中空間隔離的效果。

```
{
  "Statement": [
    {
      "Action": [
        "oss:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app - base - oss / user1 / *", "acs:oss:*:*:app - base - oss "]
    }
  ],
  "Version": "1"
}
```

對OSS的操作	結果
列舉所有建立的Bucket	失敗
上傳不帶首碼的Object, test.txt	失敗
下載不帶首碼的Object, test.txt	失敗
上傳帶首碼的Object, user1/test.txt	成功
下載帶首碼的Object, user1/test.txt	失敗
列舉Object, test.txt	失敗
帶首碼的Object, user1/test.txt	失敗

- 不限制首碼的讀寫Policy

此Policy表示應用可以對Bucket `app - base - oss` 下所有的Object進行列舉、下載、上傳和刪除。

```
{
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:PutObject",
        "oss:DeleteObject",
        "oss:ListParts",
        "oss:AbortMulti partUpload",
        "oss:ListObjects"
      ],
      "Effect": "Allow",
      "Resource": ["acs:oss:*:*:app - base - oss /*", "acs:oss:*:*:app - base - oss "]
    }
  ],
  "Version": "1"
}
```

```
}

```

對OSS的操作	結果
列舉所有建立的Bucket	失敗
上傳不帶首碼的Object, test.txt	成功
下載不帶首碼的Object, test.txt	成功
上傳帶首碼的Object, user1/test.txt	成功
下載帶首碼的Object, user1/test.txt	成功
列舉Object, test.txt	成功
帶首碼的Object, user1/test.txt	成功

· 限制首碼的讀寫Policy

此Policy表示應用可以對Bucket `app - base - oss` 下帶有首碼 `user1 /`的Object進行列舉、下載、上傳和刪除，但無法對其他首碼的Object進行讀寫。採用此種Policy，如果不同的應用對應不同的首碼，就可以達到在同一個Bucket中空間隔離的效果。

```
{
  "Statement": [
    {
      "Action": [
        "oss : GetObject ",
        "oss : PutObject ",
        "oss : DeleteObject ",
        "oss : ListParts ",
        "oss : AbortMulti partUpload ",
        "oss : ListObject s "
      ],
      "Effect": " Allow ",
      "Resource": ["acs : oss ::*: app - base - oss / user1 /
*"], "acs : oss ::*: app - base - oss "]
    }
  ],
  "Version": " 1 "
}
```

對OSS的操作	結果
列舉所有建立的Bucket	失敗
上傳不帶首碼的Object, test.txt	失敗
下載不帶首碼的Object, test.txt	失敗
上傳帶首碼的Object, user1/test.txt	成功
下載帶首碼的Object, user1/test.txt	成功
列舉Object, test.txt	成功
帶首碼的Object, user1/test.txt	成功

複雜Policy樣本

```

{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " oss : GetBucketAcl ",
        " oss : ListObject s "
      ],
      " Resource ": [
        " acs : oss :*: 1775305056 529849 : mybucket "
      ],
      " Effect ": " Allow ",
      " Condition ": {
        " StringEquals ": {
          " acs : UserAgent ": " java - sdk ",
          " oss : Prefix ": " foo "
        },
        " IpAddress ": {
          " acs : SourceIp ": " 192 . 168 . 0 . 1 "
        }
      }
    },
    {
      " Action ": [
        " oss : PutObject ",
        " oss : GetObject ",
        " oss : DeleteObject "
      ],
      " Resource ": [
        " acs : oss :*: 1775305056 529849 : mybucket / file
*"
      ],
      " Effect ": " Allow ",
      " Condition ": {
        " IpAddress ": {
          " acs : SourceIp ": " 192 . 168 . 0 . 1 "
        }
      }
    }
  ]
}

```

這是一個較為複雜的授權 Policy，使用者用這樣的一個 Policy 通過 RAM 或 STS 服務向其他使用者授權。Policy 當中有一個 Statement（一條 Policy 當中可以有多條 Statement）。Statement 裡面規定了相應的 Action、Resource、Effect 和 Condition。

這條 Policy 把使用者自己名下的 mybucket 和 mybucket / file *這些資源授權給相應的使用者，並且支援 GetBucketAcl、GetBucket、PutObject、GetObject 和 DeleteObject 這幾種操作。Condition 中的條件表示 UserAgent 為 java-sdk，源 IP 為 192.168.0.1 的時候鑒權才能通過，被授權的使用者才能訪問相關的資源。Prefix 這個 Condition 是在 GetBucket（ListObjects）的時候起作用的，關於這個欄位的解釋詳見 OSS 的 API 文檔。

Version

Version定義了Policy的版本，本文檔中 Version 設定為 1 。

Statement

通過Statement描述授權語義，其中可以根據業務情境包含多條語義，每條包含對 Action、Effect、Resource 和 Condition 的描述。每次請求系統會逐條依次匹配檢查，所有匹配成功的 Statement 會根據 Effect 的設定不同分為通過（Allow）、禁止（Deny），其中禁止（Deny）的優先。如果匹配成功的都為通過，該條請求即鑒權通過。如果匹配成功有一條禁止，或者沒有任何條目匹配成功，該條請求被禁止訪問。

Action

Action 分為三大類：

- Service 層級操作，對應的是 GetService 操作，用來列出所有屬於該使用者的 Bucket 列表。
- Bucket 層級操作，對應類似於 oss:PutBucketAcl、oss:GetBucketLocation之類的操作，操作的對象是 Bucket，它們的名稱和相應的介面名稱一一對應。
- Object 層級操作，分為 oss:GetObject、oss:PutObject、oss:DeleteObject和oss:AbortMultipartUpload，操作對象是 Object。

如想授權某一類的 Object 的操作，可以選擇這幾種的一種或幾種。另外，所有的 Action 前面都必須加上 `oss:`，如上面例子所示。Action 是一個列表，可以有多個 Action。具體的 Action 和 API 介面的對應關係如下：

- Service層級

API	Action
GetService (ListBuckets)	oss:ListBuckets

- Bucket 層級

API	Action
PutBucket	oss:PutBucket
GetBucket (ListObjects)	oss:ListObjects
PutBucketAcl	oss:PutBucketAcl
DeleteBucket	oss>DeleteBucket
GetBucketLocation	oss:GetBucketLocation
GetBucketAcl	oss:GetBucketAcl
GetBucketLogging	oss:GetBucketLogging

API	Action
PutBucketLogging	oss:PutBucketLogging
DeleteBucketLogging	oss>DeleteBucketLogging
GetBucketWebsite	oss:GetBucketWebsite
PutBucketWebsite	oss:PutBucketWebsite
DeleteBucketWebsite	oss>DeleteBucketWebsite
GetBucketReferer	oss:GetBucketReferer
PutBucketReferer	oss:PutBucketReferer
GetBucketLifecycle	oss:GetBucketLifecycle
PutBucketLifecycle	oss:PutBucketLifecycle
DeleteBucketLifecycle	oss>DeleteBucketLifecycle
ListMultipartUploads	oss:ListMultipartUploads
PutBucketCors	oss:PutBucketCors
GetBucketCors	oss:GetBucketCors
DeleteBucketCors	oss>DeleteBucketCors
PutBucketReplication	oss:PutBucketReplication
GetBucketReplication	oss:GetBucketReplication
DeleteBucketReplication	oss>DeleteBucketReplication
GetBucketReplicationLocation	oss:GetBucketReplicationLocation
GetBucketReplicationProgress	oss:GetBucketReplicationProgress

• Object 層級

API	Action
GetObject	oss:GetObject
HeadObject	oss:GetObject
PutObject	oss:PutObject
PostObject	oss:PutObject
InitiateMultipartUpload	oss:PutObject
UploadPart	oss:PutObject
CompleteMultipart	oss:PutObject
DeleteObject	oss>DeleteObject

API	Action
DeleteMultipleObjects	oss:DeleteObject
AbortMultipartUpload	oss:AbortMultipartUpload
ListParts	oss:ListParts
CopyObject	oss:GetObject,oss:PutObject
UploadPartCopy	oss:GetObject,oss:PutObject
AppendObject	oss:PutObject
GetObjectAcl	oss:GetObjectAcl
PutObjectAcl	oss:PutObjectAcl
RestoreObject	oss:RestoreObject

Resource

Resource 指代的是 OSS 上面的某個具體的資源或者某些資源（支援*通配），resource的規則是在acs : oss :{ region }:{ bucket_ownership }:{ bucket_name }/{ object_name }。對於所有 Bucket 層級的操作來說不需要最後的斜杠和{object_name}，即 acs : oss :{ region }:{ bucket_ownership }:{ bucket_name }。Resource 也是一個列表，可以有多個 Resource。其中的 region 欄位暫時不做支援，設定為*。

Effect

Effect 代表本條的Statement的授權的結果，分為 Allow 和 Deny，分別指代通過和禁止。多條 Statement 同時匹配成功時，禁止（Deny）的優先順序更高。

例如，期望禁止使用者對某一目錄進行刪除，但對於其他檔案有全部許可權：

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

```
}  
]  
}
```

Condition

Condition 代表 Policy 授權的一些條件，上面的樣本裡面可以設定對於 `acs:UserAgent` 的檢查、`acs:SourceIp` 的檢查，還有 `oss:Prefix` 項用來在 `GetBucket` 的時候對資源進行限制。

OSS 支援的 Condition 如下：

condition	功能	合法取值
<code>acs:SourceIp</code>	指定 ip 網段	普通的 ip，支援*通配
<code>acs:UserAgent</code>	指定 http useragent 頭	字串
<code>acs:CurrentTime</code>	指定合法的訪問時間	ISO8601格式
<code>acs:SecureTransport</code>	是否是 https 協議	“true” 或者 “false”
<code>oss:Prefix</code>	用作 ListObjects 時的 prefix	合法的object name

最佳實務

OSS提供了[RAM策略編輯器](#)協助您快速產生RAM Policy。您也可以使用圖形化管理工具 `ossbrowser` 的[簡化 Policy 授權](#)，一鍵完成對子帳號授予特定 Bucket 或特定目錄的許可權。

針對具體情境更多的授權策略配置樣本，可以參考[教程樣本：控制儲存空間和檔案夾的存取權限和 OSS 授權常見問題](#)。

9.3 基於Bucket Policy的許可權控制

Bucket Policy是基於資源的授權策略。相比於RAM Policy，Bucket Policy支援在控制台直接進行圖形化配置操作，並且Bucket擁有者直接可以進行訪問授權。

Bucket Policy常見的應用情境如下：

- 向其他帳號的RAM使用者授權訪問。

您可以授予其他帳號的RAM使用者訪問您的OSS資源的許可權。

- 向匿名使用者授予帶特定IP條件限制的存取權限。

某些情境下，您需要向匿名使用者授予帶IP限制的存取原則。例如，企業內部的機密文檔，只允許在企業內部訪問，不允許在其他地區訪問。由於企業內部人員較多，如果針對每個人配置RAM Policy，工作量非常大。此時，您可以基於Bucket Policy設定帶IP限制的存取原則，從而高效方便地進行授權。

Bucket Policy的配置方法和教程視頻請參見[使用Bucket Policy授權其他使用者訪問OSS資源](#)。

9.4 跨帳號授權

9.4.1 跨帳號授權概述

針對跨帳號授權訪問 OSS 資源的需求，OSS 提供了多種跨帳號授權訪問的方式。

OSS 資源預設都是私人的，若擁有者希望將資源共用給他人，可以通過跨帳號授權的方式，將 OSS 資源授權給他人訪問，下面提供幾種跨帳號授權訪問 OSS 資源的方式。

- **基於 Bucket Policy 實現跨帳號訪問OSS**：Bucket Policy 是基於資源的授權策略。相比於 RAM Policy，Bucket Policy 操作簡單，支援在控制台直接進行圖形化配置，並且 Bucket 擁有者直接可以進行訪問授權，無需具備 RAM 操作許可權。Bucket Policy 支援向其他帳號的 RAM 使用者、匿名使用者等授予帶特定 IP 條件限制的存取權限。
- **跨帳號授權OSS資源**：RAM 管理員可以配置 RAM 角色，並將夥伴的雲帳號 ID 添加為受信任雲帳號 ID。通過授予該 RAM 角色配置 OSS 存取權限的方式，可以將 OSS 資源共用給夥伴。

9.4.2 教程樣本：基於Bucket Policy實現跨帳號訪問OSS

阿里雲 OSS 的資源預設都是私人的，若您希望您的夥伴可以訪問您的 OSS 資源，可以通過 Bucket Policy 授予夥伴訪問 Bucket 的許可權。

案例：公司 A 希望其合作公司 B 可以訪問自己的 OSS 資源，但又不方便開放子帳號給 B 公司。此時，A 公司可以通過 Bucket Policy 授予夥伴訪問 Bucket 的許可權。B 公司帳號獲得授權之後，可以在控制台添加 A 公司 OSS 資源的訪問路徑進行訪問。

添加 Bucket Policy

- B 公司帳號：
 1. 登入 [RAM 存取控制台](#)，建立 RAM 子帳號，詳細配置方法請參見[建立 RAM 使用者](#)。
 2. 在 RAM 存取控制台，單擊使用者。
 3. 單擊剛剛建立的 RAM 使用者使用者名稱，查看並記錄 RAM 使用者的 U 識別碼。
- A 公司帳號
 1. 登入阿里雲 [OSS 控制台](#)。
 2. 在左側功能表列選擇您需要授權訪問的 Bucket。
 3. 單擊檔案管理 > 授權 > 新增授權。
 4. 在新增授權的對話方塊，填寫授權策略。其中，授權使用者選擇其他帳號，並填寫 B 公司子帳號的 U 識別碼。其他參數請參考 [Bucket Policy](#)。
 5. 單擊確定。

登入 RAM 子帳號並添加訪問路徑

Bucket Policy 添加完成之後，您還需要登入 B 公司子帳號，添加 A 公司的 Bucket 訪問路徑。配置步驟如下：

1. 通過 [RAM 帳號登入連結](#) 登入 B 公司子帳號。
2. 開啟 [OSS 控制台](#)。
3. 單擊左側功能表列我的訪問路徑後的加號 (+)，添加 A 公司授權訪問的 Bucket 路徑資訊。
 - 地區：下拉選擇 A 公司允許訪問的 Bucket 所在地區。
 - 訪問路徑：添加 A 公司允許訪問的 Bucket 訪問路徑，格式為「bucket/object-prefix」。
例如，僅允許訪問名為 *aliyun* 的 Bucket 下的 *abc* 檔案夾，則添加 *aliyun / abc*。

您也可以[建立子帳號的 AccessKey](#)，並通過 AccessKey 使用 [ossutil](#)、[ossbrowser](#) 等工具訪問被授權的 Bucket。

參考文檔

您也可以通過以下方式授權夥伴訪問您的 OSS 資源：

- [教程樣本：基於 RAM 角色實現跨帳號訪問 OSS](#)

10 安全管理

10.1 伺服器端加密編碼

本文介紹如何使用OSS的服務端加密功能，對持久化在OSS上的數據進行加密保護：

- [服務端加密選項](#)
- [OSS完全託管](#)
- [KMS託管主要金鑰](#)
- [服務端加密API使用](#)

OSS支援在伺服器端對上傳的數據進行加密編碼（Server-Side Encryption）：用戶上傳數據時，OSS對收到的用戶數據進行加密編碼，然後再將得到的加密資料持久化保存下來；用戶下載數據時，OSS自動對保存的加密資料進行解密並把未經處理資料返回給用戶，並在返回的HTTP請求Header中，聲明該數據進行了伺服器端加密。

OSS服務端加密主要應用場景

OSS通過服務端加密機制，提供待用資料保護。適合於對於檔案儲存體有高安全性或者合規性要求的應用場景。例如，深度學習樣本檔案的儲存、線上協作類文檔數據的儲存。針對不同的應用場景，OSS有如下幾種服務端加密方式：

- 直接使用KMS進行加密、解密

用戶直接調用KMS的API，使用指定的CMK進行加密、解密數據。這種方式適合於少量數據的加解密服務（且檔案大小小於4KB）。用戶數據會通過安全頻道傳輸至KMS服務端，對應的加解密結果通過安全頻道返回給客戶。

- 使用KMS託管的信封加密

用戶直接調用KMS的API，使用指定的CMK來產生明文數據密鑰、密文數據密鑰。這種場景適合於大量的數據加解密。用戶數據無需通過網路發送到KMS服務端進行加解密，這是一種低成本的加解密方式。

- 使用OSS完全託管加密

基於OSS完全託管的加密方式，是Object的一種屬性。用戶建立object時，只需要put object請求中攜帶x-oss-server-side-encryption的HTTP Header，並指定其值為AES256，即可以實現該Object的伺服器端加密儲存。該方式適合於批量數據的加解密。

服務端加密選項

OSS服務端加密有以下選項，用戶可根據需要（密鑰管理原則）進行選擇：

- 使用由OSS完全託管的服務端加密功能：資料加密金鑰的生成和管理，由OSS負責，並採用高強度、多因素的安全措施進行保護。資料加密的演算法採用使用行業標準的強密碼編譯演算法AES-256（即256位進階加密標準）。

服務端加密-OSS完全託管

OSS的伺服器端加密是Object的一個屬性。用戶建立一個Object的時候，只需要在Put Object的請求中攜帶 `x-oss-server-side-encryption` 的HTTP Header，並指定其值為 `AES256`，即可以實現該Object的伺服器端加密儲存。

以下操作支援在請求中攜帶這些HTTP Header：

- PutObject：簡單上傳
- CopyObject：複製Object
- InitiateMultipartUpload：分區上傳

服務端加密-KMS託管主要金鑰

當用戶建立Object時，在請求中攜帶 `x-oss-server-side-encryption` 的HTTP Header，並指定其值為 `KMS`，即可實現在服務端加密儲存該Object（密鑰管理使用KMS服務）。OSS通過KMS服務，採用信封加密機制、AES-256密碼編譯演算法等強安全方法，對用戶儲存在OSS的對象進行加密保護。

KMS是阿里雲提供的一款安全、易用的管理類服務。用戶無需花費大量成本來保護密鑰的保密性、完整性和可用性，藉助金鑰管理服務，用戶可以安全、便捷的使用密鑰，專注於開發加解密功能場景。用戶可以通過KMS控制台中查看和管理KMS密鑰。

以下操作支援在請求中攜帶這些HTTP Header

- PutObject：簡單上傳
- CopyObject：複製Object
- InitiateMultipartUpload：分區上傳



说明：

- 使用該模式，首次向某地域中的Bucket添加加密對象時，將自動為您該地域建立一個預設KMS密鑰（用作用戶主要金鑰CMK）。此密鑰將用於服務端加密（KMS模式）。
- 用戶可以在KMS控制台查看該密鑰資訊，請參見[KMS使用文檔](#)。

服務端加密API使用

操作請求

以下API支援要求攜帶 `x - oss - server - side - encryption` 頭：

- PutObject：簡單上傳
- CopyObject：複製Object
- InitiateMultipartUpload：分區上傳

HTTP Header格式如下：

名稱	描述	樣本
x-oss-server-side-encryption	指定服務端加密的模式 有效值：AES256、KMS	<code>x - oss - server - side - encryption :</code> KMS 表示使用服務端加密模式為KMS託管主要金鑰



说明：

- 其他OSS收到的請求中，如果出現 `x - oss - server - side - encryption` 頭，OSS會直接返回HTTP狀態碼400，並在消息體內註明錯誤碼為InvalidArgument。
- 如果您指定的 `x - oss - server - side - encryption` 頭不為有效值，OSS會直接返回HTTP狀態碼400，並在消息體內註明錯誤碼為 InvalidEncryptionAlgorithmError。

操作響應

通過伺服器端加密儲存的Object，以下API請求中OSS會返回 `x - oss - server - side - encryption` 頭：

- PutObject
- CopyObject
- InitiateMultipartUpload
- UploadPart
- CompleteMultipartUpload
- GetObject
- HeadObject

Meta資訊

通過伺服器端加密-KMS託管主要金鑰模式儲存的Object，其Meta資訊會增加以下欄位：

名稱	描述	樣本
x-oss-server-side-encryption	指定伺服器端加密的模式	x - oss - server - side - encryption : KMS
x-oss-server-side-encryption-key-id	該Object加密，所用的用戶KMS key的ID	x - oss - server - side - encryption - key - id : 72779642 - 7d88 - 4a0f - 8d1f - 1081a9cc7a fb

相關API

- API: [Append Object](#)
- API: [Put Object](#)
- API: [Copy Object](#)
- API: [Post Object](#)

11 靜態網站託管

11.1 配置靜態網站託管

OSS支援靜態網站託管。用戶可以通過[OSS 控制台](#)將自己的儲存空間配置成靜態網站託管模式。

配置生效後，假如這個Bucket在杭州，那麼這個靜態網站的訪問網域名稱為：

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

為了使用戶更方便地管理在OSS上託管的靜態網站，OSS支援兩種功能：

- 靜態頁面支援（Index Document Support）

靜態頁是指當用戶直接存取靜態網站根網域名稱時，OSS返回的預設靜態頁（相當於網站的index.html）。如果您為一個Bucket配置了靜態網站託管模式，就必須指定一個靜態頁。

- 錯誤頁面支援（Error Document Support）

錯誤頁面是指在用戶訪問該靜態網站時，如果遇到HTTP 4XX錯誤時（最典型的是404 “NOT FOUND” 錯誤），OSS返回給用戶的錯誤頁面。通過指定錯誤頁面，您可以為您的用戶提供恰當的出錯提示。

例如：使用者佈建索引頁面為index.html，錯誤頁面為error.html，Bucket為oss-sample，Endpoint為oss-cn-hangzhou.aliyuncs.com，那麼：

- 用戶訪問 `http :// oss - sample . oss - cn - hangzhou . aliyuncs . com /`和 `http :// oss - sample . oss - cn - hangzhou . aliyuncs . com / directory /`的時候，相當於訪問 `http :// oss - sample . oss - cn - hangzhou . aliyuncs . com / index . html`
- 用戶訪問 `http :// oss - sample . oss - cn - hangzhou . aliyuncs . com / object` 的時候，如果object不存在，OSS會返回 `http :// oss - sample . oss - cn - hangzhou . aliyuncs . com / error . html`

細節分析

- 所謂靜態網站是指所有的網頁都由靜態內容構成，包括客戶端執行的指令碼或直譯式程式，例如JavaScript。OSS不支援涉及到需要伺服器端處理的內容，例如PHP，JSP，APS.NET等。
- 如果您想使用自己的網域名稱來訪問基於Bucket的靜態網站，可以通過[綁定自訂網域名CNAME](#)來實現。
- 將一個Bucket設定成靜態網站託管模式時，必須指定索引頁面，錯誤頁面是可選的。

- 將一個Bucket設定成靜態網站託管模式時，指定的索引頁面和錯誤頁面必須是該Bucket內的一個Object。
- 在將一個Bucket設定成靜態網站託管模式後，對靜態網站根網域名稱的匿名訪問，OSS將返回索引頁面；對靜態網站根網域名稱的簽名訪問，OSS將返回Get Bucket結果。
- Bucket設定靜態網站託管模式後，對於靜態網站根網域名稱的訪問或者訪問不存在的Object會返回給用戶設定的Object，對此返回的流量和請求將會計費。

功能使用參考

- API: [PutBucketWebsite](#)
- 控制台: [靜態網站託管](#)

11.2 教程樣本：使用自訂網域名設定靜態網站託管

假設您要在阿里雲 OSS 上託管靜態網站。您註冊了網域名稱（例如，examplewebsite.com）並且想要從 OSS 內容響應對 `http://examplewebsite.com` 和 `http://www.examplewebsite.com` 的請求。不論您是已經有了要在 OSS 上託管的靜態網站，還是要從頭開始建立，都可以通過以下樣本了解如何在阿里雲 OSS 上託管網站。

準備工作

本教程涉及到以下服務：

- 網域名稱註冊

如果您沒有註冊網域名稱（如 `examplewebsite.com`），請選擇一個註冊商進行註冊。阿里雲也提供網域名稱註冊服務。詳情請參考 [阿里雲網域名稱服務 \(DNS\)](#)。

- 阿里雲 OSS

您可以使用阿里雲 OSS 建立 bucket，上傳樣本網頁，配置許可權允許所有人查看內容，然後配置 bucket 的網站託管功能。在本樣本中，因為要允許對 `http://examplewebsite.com` 和 `http://www.examplewebsite.com` 的請求，所以需要建立兩個 bucket。您可以僅在一個 bucket 中託管內容，再配置另一個 bucket 將請求重新導向到託管內容的 bucket。

- 雲解析

您可以將阿里雲解析配置為網域名稱解析服務 (DNS) 供應商。在本樣本中，您可以將網域名稱添加到雲解析並定義 CNAME 記錄，這樣能夠使用您的網域名稱而不是 OSS 分配的訪問網域名稱來訪問 OSS bucket。

- 在本樣本中，我們使用阿里雲解析。但您可以通過大多數註冊商定義指向 OSS bucket 的 CNAME 記錄。



说明:

本教程使用 `exampleweb site . com` 作為網域名稱。請使用您註冊的網域名稱替換此網域名稱。

步驟 1：註冊網域名稱

如果您已有註冊網域名稱，可以跳過該步驟。如果您從未託管過網站，請先註冊一個網域名稱，例如 `exampleweb site . com`。您可以使用阿里雲網域名稱服務 (DNS) 註冊一個網域名稱。

詳情請參考[購買網域名稱](#)。

步驟 2：建立和配置 bucket 並上傳數據

您可以建立兩個 bucket，以同時支援來自根網域名稱（如 `exampleweb site . com`）和子網域名稱（如 `http :// www . exampleweb site . com`）的請求。一個 bucket 用於儲存內容，另一個 bucket 用於將請求重新導向到儲存內容的 bucket。

步驟 2.1：建立兩個 bucket

在該步驟中，使用阿里雲帳戶登入 OSS 控制台，並建立以下兩個 bucket：

- originbucket：用於儲存內容
- redirectbucket：用於將請求重新導向到originbucket

1. 登入[OSS 控制台](#)。
2. 建立兩個 bucket（例如originbucket和redirectbucket）。將兩個 bucket 的讀寫權限設定為公共讀取，以便所有人都可以查看 bucket 的內容。

詳細流程，請參考[建立 bucket](#)。

3. 記下originbucket和redirectbucket的訪問網域名稱。在後續步驟中將會用到它們。您可以在bucket的概覽頁面上找到bucket的訪問網域名稱，如下圖所示。

The screenshot displays the 'originbucket' overview page in the OSS console. The left sidebar shows the 'Storage Spaces' list with 'originbucket' selected. The main content area shows the 'Overview' tab. The 'Basic Information' section includes a warning about data latency and five metrics: storage usage, network traffic, request counts, file count, and file fragments, all showing 0. The 'Access Domains' table lists three domains, with the first one, 'originbucket.oss-cn-hangzhou.aliyuncs.com', highlighted by a red box.

访问域名	EndPoint	访问域名	HTTPS
外网访问	oss-cn-hangzhou.aliyuncs.com	originbucket.oss-cn-hangzhou.aliyuncs.com	支持
ECS 的经典网络访问 (内网)	oss-cn-hangzhou-internal.aliyuncs.com	originbucket.oss-cn-hangzhou-internal.aliyuncs.com	支持
ECS 的 VPC 网络访问 (内网)	oss-cn-hangzhou-internal.aliyuncs.com	originbucket.oss-cn-hangzhou-internal.aliyuncs.com	支持

4. 將網站數據上傳到 originbucket。

您將在根域 bucket originbucket 外託管內容，並且將針對子域 bucket redirectbucket 的請求重新導向到根域 bucket originbucket。您可以在任一 bucket 中儲存內容。

本樣本將內容託管在originbucket bucket 中。內容可以是任何類型的檔案，例如文字檔、照片和視頻。如果您尚未建立網站，您在本樣本中僅需要兩個檔案。一個檔案用作網站首頁，另一個檔案用作網站的錯誤頁面。

例如，使用以下 HTML 建立一個名為 index.html 的檔案，並將其上傳到 bucket。在後續步驟中，將使用此檔案名作為網站的預設首頁。

```
< html >
  < head >
    < title > Hello  OSS !</ title >
    < meta charset =" utf - 8 ">
  </ head >
  < body >
    < p >開始阿里雲 OSS 託管</ p >
    < p >這是索引頁面</ p >
  </ body >
</ html >
```

您可以使用以下 HTML 建立另一個名為error.html 的檔案，並將其上傳到bucket。此檔案用作網站的 404 錯誤頁面。在後續步驟中，將使用此檔案名作為網站的預設404 頁面。

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

```
< p >這是 404 錯誤頁面</ p >  
</ body >  
</ html >
```

步驟 2.2: 配置 bucket 的網站託管功能

在配置 bucket 的網站託管功能後，可以通過 OSS 分配的訪問網域名稱訪問該網站。

在此步驟中，將 originbucket 配置為網站。

1. 登入[OSS 控制台](#)。
2. 從 bucket 名稱列表中，選擇 originbucket。
3. 單擊基礎設定頁籤，找到靜態頁面區域。
4. 單擊設定，然後輸入以下資訊：
 - 預設首頁：索引頁面（相當於網站的 index.html）。只能使用已儲存在 bucket 中的 HTML 檔案。在本樣本中，輸入 index.html。
 - 預設 404 頁：當訪問錯誤路徑時，返回預設的 404 頁面。只能使用已儲存在 bucket 中的 HTML 和影像檔。在本樣本中，輸入 error.html。
5. 單擊保存。

步驟 2.3: 配置索引頁面的重新導向功能

配置 originbucket 的預設首頁和錯誤頁面之後，您還需要配置 redirectbucket 的預設首頁。

參考以下步驟配置索引頁面的重新導向功能：

1. 登入[OSS 控制台](#)。
2. 從 bucket 名稱列表中，選擇 redirectbucket。
3. 單擊基礎設定頁籤，找到靜態頁面區域。
4. 單擊設定，然後在預設首頁文字框中輸入 index.html。
5. 單擊保存。

步驟 3: 將網域名稱綁定到 OSS bucket

現在，您已有了根網域名稱 examplewebsite.com 和 OSS bucket originbucket，接下來您需要將網域名稱綁定到 OSS bucket，以便能夠使用您的網域名稱而不是 OSS 分配的網域名稱來訪問 OSS bucket。

在本樣本中，在將網域名稱 examplewebsite.com 綁定到 OSS bucket originbucket 之前，必須使用 OSS 分配的網域名稱 originbucket.oss-cn-beijing.aliyuncs.com 訪問 bucket originbucket。在綁定網域名稱 examplewebsite.com 後，可以使用此 examplewebsite.com 來訪問 OSS bucket。

同樣，還需要將子網域名稱 `www . exampleweb site . com` 綁定到OSS bucket `redirectbucket`，以便能夠使用 `www . exampleweb site . com` 而不是 OSS 分配的網域名稱 `originbucket.oss-cn-beijing.aliyuncs.com` 來訪問 OSS bucket。

參考以下步驟將根網域名稱 `exampleweb site . com` 綁定到OSS bucket `originbucket`：

1. 登入OSS 控制台。
2. 從 bucket 名稱列表中，選擇 `originbucket`。
3. 單擊網域名稱管理頁籤。
4. 單擊綁定用戶網域名稱。
5. 在用戶網域名稱文字框中，輸入根網域名稱`examplewebsite.com`。
6. 將 CNAME 記錄定義為 `originbucket`。
 - 如果已在阿里雲帳戶下解析網域名稱，則可以開啟自動添加 CNAME 記錄開關。然後單擊提交。
 - 如果未在阿里雲主帳戶下解析網域名稱，自動添加 CNAME 記錄開關處於禁用狀態。執行以下步驟，手動添加 CNAME 記錄，然後單擊提交。
 - a. 在雲解析中添加網域名稱。
 - 如果網域名稱是在阿里雲中註冊的，它會自動添加到雲解析列表中。您可以跳過該步驟。
 - b. 在雲解析控制台中，找到網域名稱。
 - c. 單擊網域名稱。
 - d. 單擊添加解析。
 - e. 在添加解析對話方塊中，從記錄類型下拉框中選擇 `CNAME`，然後在記錄值文字框中輸入 `bucket` 的 OSS 網域名稱。在本樣本中，輸入`originbucket.oss-cn-beijing.aliyuncs.com`。
 - f. 單擊確認。
7. 參考上述步驟，將子網域名稱 `www . exampleweb site . com` 綁定到 OSS bucket `redirectbucket`。

步驟 4：配置網站重新導向功能

您已配置 bucket 的網站託管功能並將自訂網域名綁定到 OSS bucket，接下來需要配置 `redirectbucket`，將 `http :// www . exampleweb site . com` 的所有請求重新導向到 `http :// exampleweb site . com`。

參考以下步驟配置網站的重新導向功能：

1. 登入[OSS 控制台](#)。
2. 從 bucket 列表中，選擇 `redirectbucket`。
3. 單擊基礎設定頁籤，找到鏡像回源區域。
4. 單擊設定，然後單擊建立規則。
5. 建立 404 重新導向規則，具體步驟如下：
 - a. 在回源類型區域，選擇重新導向。
 - b. 在回源條件區域，設定 HTTP 狀態碼為404。
 - c. 在回源地址區域，選擇添加前尾碼，輸入originbucket的網域名稱。在本樣本中，輸入 `examplewebsite.com`。
 - d. 單擊確定。

步驟 5：（可選）使用阿里雲 CDN 加快網站速度

您可以使用阿里雲 CDN 改善網站性能。CDN 讓您的網站檔案（如 HTML、映像和視頻）可供全球各地的資料中心（即，邊緣節點）使用。當訪問者從您的網站請求檔案時，CDN 自動將請求重新導向到最近邊緣節點上的檔案副本。因此下載速度要快於訪問者從較遠的資料中心請求內容。

CDN 在邊緣節點緩存內容的時間長度由您指定。如果訪問者請求的內容的緩存時間超過了到期日，CDN 會檢查原始伺服器，看看是否有新版本的內容可用。如果有新版本，CDN 將新版本複製到邊緣節點。您對原始內容所做的更改會在訪問者請求內容時複製到邊緣節點。但在到期日前，內容仍為之前的版本。我們建議開啟CDN 緩存自動刷新開關，以便您對原始內容所做的更改可以在 CDN 緩存中自動即時刷新。

參考以下步驟使用 CDN 加快originbucket速度：

1. 在 CDN 控制台中添加 CDN 加速網域名稱，並複製該網域名稱的 CNAME 地址。詳情請參考 [CDN 快速入門](#)中的步驟 2：添加加速網域名稱。
2. 在雲解析控制台中定義 CNAME 記錄。詳情請參考[配置CNAME流程](#)。
3. 開啟CDN 緩存自動刷新開關。
 - a. 登入[OSS 控制台](#)。
 - b. 從 bucket 列表中，選擇 `originbucket`。
 - c. 單擊網域名稱管理頁籤。
 - d. 開啟CDN 緩存自動刷新開關。
4. 參考上述步驟，使用 CDN 為 `redirectbucket` 加速。

步驟 6：測試網站

要驗證網站是否正常運行，請在瀏覽器中嘗試以下 URL：

URL	結果
<code>http://exampleweb site.com</code>	顯示originbucket中的索引文檔。
originbucket中不存在的檔案的 URL，例如 <code>http://exampleweb site.com / abc</code>	顯示originbucket中的錯誤頁面。
<code>http://www.exampleweb site. com</code>	將您的請求重新導向到 <code>http:// exampleweb site.com</code> 。
<code>http://www.exampleweb site. com / abc</code>	將您的請求重新導向到 <code>http:// exampleweb site.com / abc</code> 。

**说明:**

某些情況下，可能需要清除網頁瀏覽器的緩存才能看到預期行為。

清理

如果您建立靜態網站只是為了練習，別忘了刪除分配的阿里雲資源以免產生不必要的費用。刪除阿里雲資源後，網站不再可用。

參考以下步驟進行清理：

1. 禁用阿里雲 CDN 控制台中的網域名稱，然後將其刪除。
2. 刪除雲解析控制台中的 CNAME 記錄。
3. 刪除阿里雲 OSS 控制台中的 OSS 檔案和 bucket。

12 OSS沙箱

當您的 OSS Bucket 遭受攻擊或通過 Bucket 分享違法內容，OSS 會自動將 Bucket 切入沙箱。沙箱中的 Bucket 仍可以正常響應請求，但服務品質將被降級，您的應用可能會有明顯感知。

- 對於被攻擊的 Bucket，OSS 會將其切入沙箱。若您的 Bucket 遭受攻擊，您需要自行承擔因攻擊而產生的全額費用。
- 若您的使用者通過您的 Bucket 分享涉黃、涉政、涉恐等違法內容，也會導致您的 Bucket 被切入沙箱。情節嚴重者，將被追究法律責任。



说明:

當您的 Bucket 切入到沙箱中後，您將會收到簡訊和郵件通知。

針對攻擊的預防措施

為防止您的 Bucket 因攻擊原因被切入沙箱，您可使用高防 IP 來抵禦 DDoS 攻擊和 CC 攻擊。如果您的業務有可能遭受攻擊，可以按照如下兩種方案添加預防措施。

· 方案一：綁定網域名稱並配置高防 IP

1. OSS 綁定自訂網域名，配置步驟請參考[綁定自訂網域名](#)。
2. 根據業務情況，購買合適的[高防 IP](#)。
3. 將高防 IP 綁定到您已設定好的自訂網域名上。

防護網站：

注意：如果您加的是*.taobao.com这样的泛域名，请您再加一遍顶级域名taobao.com的规则；一级域名与二级域名需要分开配置

协议类型： ☒ HTTP ☐ HTTPS ☐ websocket ☐ websockets

源站IP/域名： ☐ 源站IP ☒ 源站域名

如果源站暴露，请参考[使用高防后源站IP暴露的解决方法](#)。

[下一步](#)

- 防護網站：填寫您綁定的 OSS 自訂網域名。
- 協議類型：按照您的實際訪問方式選擇。
- 來源站點IP/網域名稱：選擇來源站點網域名稱，並填寫您 OSS 的預設訪問網域名稱。

其他步驟及參數填寫，請參考文檔[設定高防 IP](#)。

· 方案二：配置 ECS 反向 Proxy並配置高防 IP

基於安全考慮，Bucket 預設網域名稱解析的 IP 是隨機變化的，若您期望使用固定 IP 方式訪問，推薦使用 ECS 搭建反向 Proxy的方式進行訪問 OSS。ECS 上的 EIP 可以綁定高防 IP 以抵禦 DDoS 攻擊和 CC 攻擊。具體可以按照如下方式進行配置：

1. 建立一個 CentOS 或 Ubuntu 的 ECS 執行個體，詳情請參考[建立 ECS 執行個體](#)。



注意：

若 Bucket 有很大的網路流量或訪問請求，請提高 ECS 硬體設定或者搭建 ECS 叢集。

2. 配置 ECS 反向 Proxy 方式訪問 OSS，詳細配置可參考[配置 OSS 反向 Proxy](#)。
3. 根據業務情況，購買合適的[高防 IP](#)。
4. 參考方案一的[步驟3](#)。其中，防護網站填寫您 ECS 綁定的網域名稱，來源站點 IP/網域名稱填寫 ECS 的外網 IP。

方案優劣勢分析

方案名稱	優勢	劣勢
方案一：綁定網域名稱並配置高防 IP	配置簡單：支援控制台圖形化設定。	應用情境有局限性：只能對未進入沙箱的 Bucket 提供防護。
方案 2：配置 ECS 反向 Proxy 並配置高防 IP	· 解決方案具有通用性：能夠為已進入沙箱和未進入沙箱的 Bucket 提供防護。 · 適合通過固定 IP 訪問 OSS 的情境。	· 配置複雜：需要使用者自定搭建 Nginx 反向 Proxy。 · 成本高：需要額外購買 ECS 搭建反向 Proxy。

針對違法內容的預防措施

為防止您的 Bucket 因分享涉黃、涉政、涉恐等違法內容被切入沙箱，建議您開通[Alibaba Content Security Service](#)服務。阿里雲 Alibaba Content Security Service 服務將定期針對您選中的 Bucket 進行檢測，有效降低涉黃、涉政、涉恐的風險。

Bucket 已進入沙箱如何處理

針對已經進入沙箱的 Bucket，阿里雲不提供遷出服務。若您的 Bucket 被切入沙箱，請按照以下操作執行。

- 因攻擊原因導致 Bucket 被切入沙箱
 - 針對已經進入沙箱的 Bucket，請按照[方案二](#)配置安全防護措施。



注意：

請在 Bucket 所在的 Region 搭建 ECS，並且 proxy_pass 填寫 Bucket 內網網域名稱地址。

- 若您的帳號下的 Bucket 曾多次遭受攻擊。那麼，您後續建立的 Bucket 預設也會進入沙箱。此時，針對建立 Bucket 的安全訪問措施如下：
 1. 購買高防 IP。
 2. 通過工單系統提交“建立 Bucket 預設不進入沙箱申請”。
 3. 申請通過後，參照[方案一](#)進行配置。
 - 因發布違法內容導致 Bucket 被切入沙箱
 - 針對已經入沙箱的 Bucket，請按照如下操作執行：
 1. 開通[Alibaba Content Security Service](#)服務，定期檢測您的 Bucket，保證不再發布違規內容。
 2. 請按照[方案二](#)配置 ECS 反向 Proxy，通過Proxy 伺服器進行訪問。
-  **注意：**
請在 Bucket 所在的 Region 搭建 ECS，並且 proxy_pass 填寫 Bucket 內網網域名稱地址。
- 若您多個 Bucket 同時發布違法內容，或單個 Bucket 多次發布違法內容。那麼，您後續建立的 Bucket 預設也會進入沙箱。此時，您需按如下操作執行：
 1. 購買Alibaba Content Security Service產品。
 2. 通過工單系統提交“建立 Bucket 預設不進入沙箱申請”。
 3. 申請通過後，配置Alibaba Content Security Service檢測，定期檢測您建立的 Bucket。

13 監控服務

13.1 監控服務概覽

OSS監控服務為您提供系統基本運行狀態、性能以及計量等方面的監控數據指標，並且提供自訂報警服務，幫助您跟蹤請求、分析使用方式、統計業務趨勢，及時發現以及診斷系統的相關問題。

OSS監控指標分類詳細，主要可以歸類為基礎服務指標、性能指標和計量指標，詳見[OSS監控指標參考](#)。

高即時性

高即時性能夠暴露可能隱藏的峰穀問題，顯示出實際的波動情況，有助於分析和評估業務場景。

OSS監控指標的即時性（除了計量指標）是按照分鐘粒度採集彙總的，輸出延時不超過1分鐘，即每分鐘內的用戶資訊都會彙總成一個值，並在一分鐘內輸出，代表這一分鐘的監控情況。

計量指標相關說明

為了保持和計費策略的統一，計量指標的收集和展現存在一定的特殊性，說明如下：

- 計量指標數據是按照小時粒度輸出的，即每個小時內的資源計量資訊都會彙總成一個值，代表這個小時總的計量情況。
- 計量指標數據會有近半個小時的延時輸出。
- 計量指標數據的數據時間是指該數據所統計時間區間的開始時間。
- 計量採集截止時間是當月最後一條計量數據所統計時間區間的結束時間，如果當月沒有產生任何一條計量監控數據，那麼計量資料擷取截止時間為當月1號0點。
- 計量指標數據的展示都是盡最大可能推送的，準確計量請參考費用中心—[使用記錄](#)

舉個例子，假設用戶只使用PutObject這個請求上傳數據，每分鐘平均10次。那麼在2018-05-10 08:00:00到2018-05-10 09:00:00這一個小時時間區間內，用戶的PUT類請求數的計量數據值為600次（10*60分鐘），數據時間為2018-05-10 08:00:00，這條數據將會在2018-05-10 09:30:00左右被輸出。如果這條數據是從2018-05-01 00:00:00開始到現在的最後一條計量監控數據，那麼當月的計量資料擷取截止時間就是2018-05-10 09:00:00。如果2018年5月該用戶沒有產生任何的計量數據，那麼計量採集截止時間為2018-05-01 00:00:00。

OSS報警服務

每個帳號最多能夠設定1000項報警規則。除計量指標和統計指標，其他的監控指標均可配置為報警規則加入報警監控，並且一個監控指標可以配置為多個不同的報警規則。

- 報警服務相關概念參考[報警服務概覽](#)。

- OSS報警服務使用指南詳見[使用報警服務](#)。
- OSS具體監控指標詳見[監控指標參考](#)。

監控數據保留原則

監控數據保留31天，過期自動清除，如果需要離線分析監控數據或者長期下載並保存曆史監控，需要使用工具或者編寫代碼來讀取Cloud Monitor資料存放區，請參考[OpenAPI訪問監控數據](#)。

控制台展示最近7天的數據，如果希望查詢7天以上的曆史數據，建議使用Cloud Monitor提供的SDK進行查詢，參考[OpenAPI訪問監控數據](#)。

OpenAPI訪問監控數據

OSS服務的相關監控指標數據可以通過Cloud Monitor提供的OpenAPI訪問，使用方法可以參考如下文檔：

- [Cloud MonitorAPI參考](#)
- [Cloud MonitorSDK參考](#)
- [訪問監控數據](#)

監控、診斷和故障排除

[監控診斷和故障排除](#)通過詳細介紹以下各個方面的內容幫助您更好的了解OSS服務的運行狀態並進行自主診斷和故障排除：

- [服務監控](#)

介紹如何使用監控服務持續監控OSS儲存服務的健全狀態和性能。

- [跟蹤診斷](#)

介紹如何使用OSS監控服務和logging記錄功能診斷問題；另外，還介紹如何關聯各種記錄檔中的相關資訊進行跟蹤診斷。

- [故障排除](#)

提供常見的問題場景和故障排除方法。

注意事項

OSS Bucket全域唯一，如果刪掉Bucket之後再建立同名的Bucket，那麼被刪掉的Bucket的監控以及報警規則會作用在新的同名Bucket上。

13.2 使用監控服務

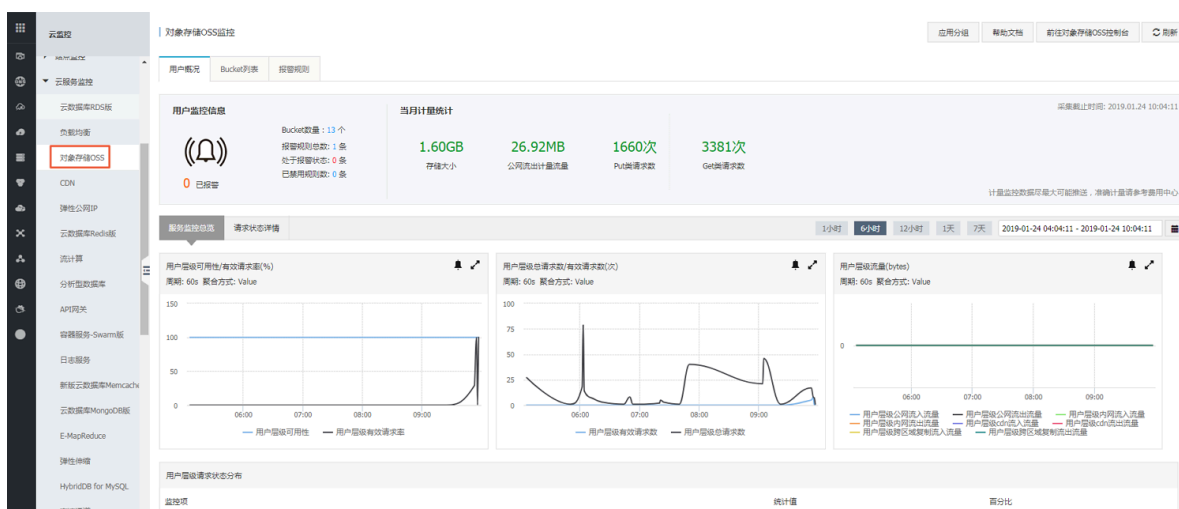
OSS監控服務入口

OSS監控服務處於Cloud Monitor控制台中。可以通過如下兩種方式進入。

- 登入OSS管理主控台，在OSS概覽頁右邊單擊設定報警規則，進入OSS監控服務。



- 直接進入Cloud Monitor控制台查看OSS監控服務。



OSS監控服務頁面

OSS監控服務首頁的主體由如下三部分組成。

- 用戶概況
- Bucket列表
- 報警規則



該頁面沒有自動刷新功能，可以單擊右上方的刷新按鈕自動更新數據資訊。

單擊前往OSS控制台可以直接進入OSS控制台介面。



用戶概況

用戶概況頁面從用戶層級監控用戶相關的資訊。主要包括用戶監控資訊、當月計量統計和用戶層級監控指標三大部分。

· 用戶監控資訊

該模組主要展示該帳號所擁有的Bucket總數以及相關的報警規則情況。



- ■ 單擊Bucket數量的數字，連結到Bucket列表Tab頁。
- 單擊報警規則總數的數字，連結到報警規則Tab頁。
- 單擊處於警示狀態的數字，連結到報警規則Tab頁，並且此時該頁展示的報警規則均處於警示狀態。
- 單擊已禁用規則數的數字，連結到報警規則Tab頁，並且此時該頁展示的報警規則均被禁用。
- 單擊警鈴表徵圖下面的數字，連結到報警規則Tab頁，並且此時該頁展示的報警規則均處於警示狀態。

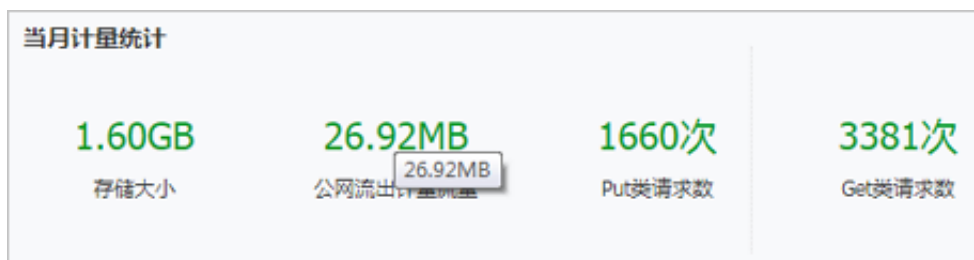
· 當月計量統計

當月計量統計展示了從當月1號0點開始，到採集截止時間為止，這段時間內所使用的OSS服務的計費相關的資源資訊，包括如下指標：

- 儲存大小
- 公網流出流量
- Put類請求數
- Get類請求數



各個計量框中展示的數據根據量級自動調整單位，滑鼠停留在數字上方會顯示精確的數值。



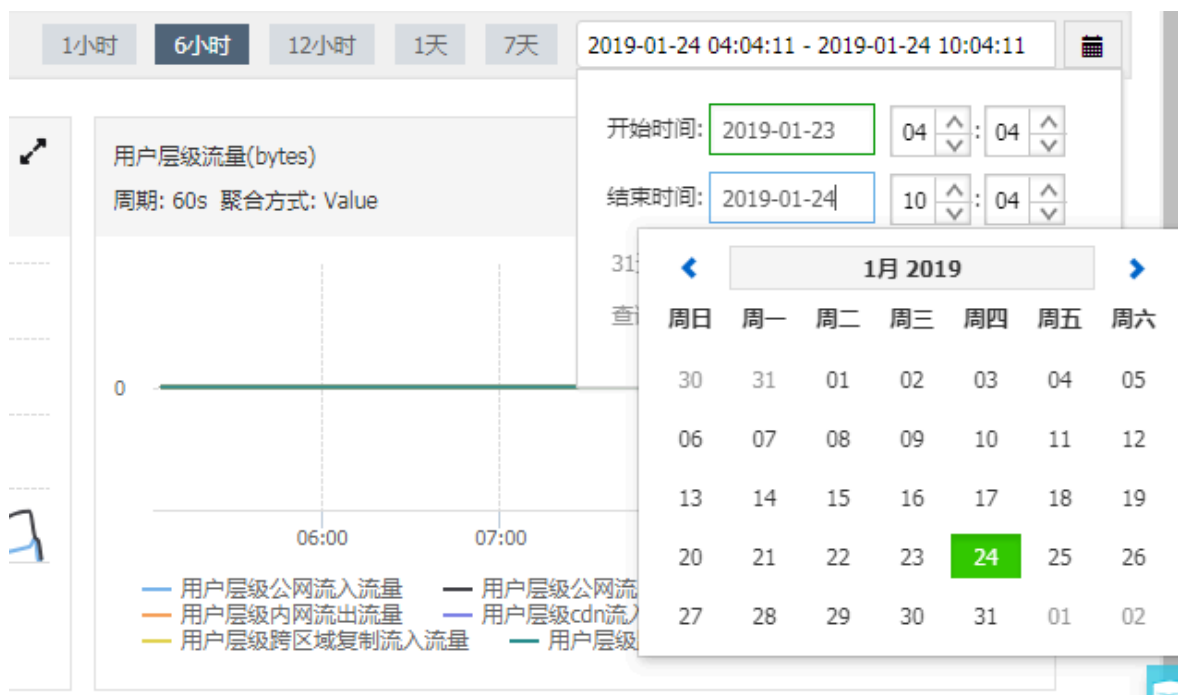
· 用戶層級監控指標

該模組主要展示具體的用戶層級的監控圖表，主要包括服務監控總覽和請求狀態詳情兩部分，下面會詳細介紹。



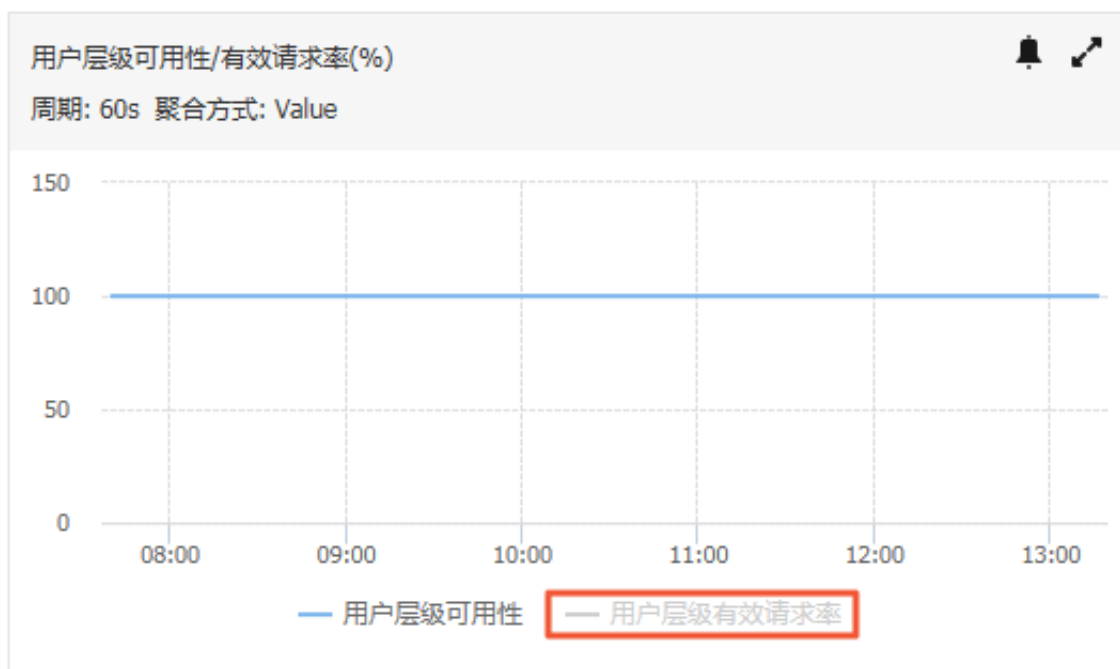
圖表展現提供了快速時間範圍選擇按鈕和自訂時間框。

- 快速時間範圍選擇按鈕提供1小時、6小時、12小時、1天和7天的時間範圍選擇，預設為1小時。
- 自訂時間框可以自訂起始時間和結束時間，精確到分鐘等級。注意，不支援查詢8天以前的數據。



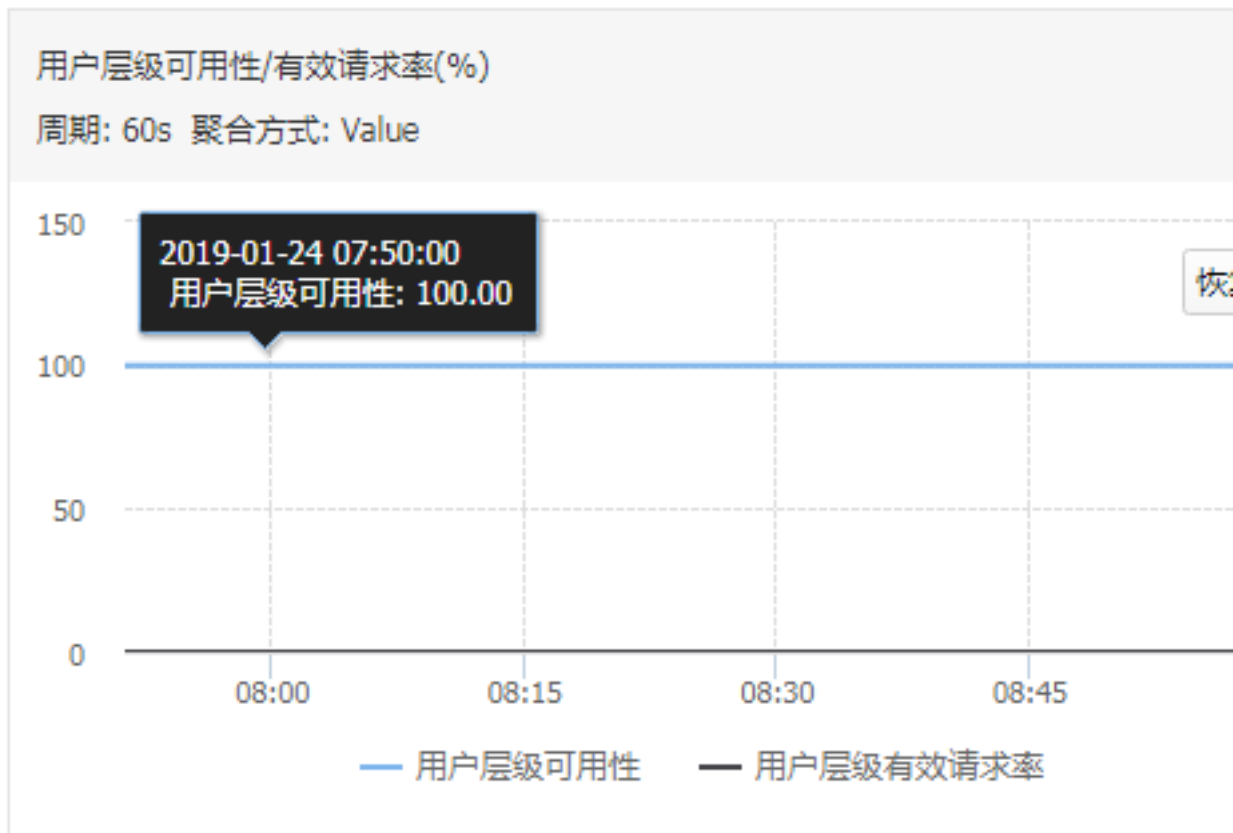
圖表展示還支援以下功能：

- 單擊相關圖例可以將該指標曲線隱去，如下圖：



- 單擊圖形右上表徵圖  可以將圖形放大展示。注意，表格不支援放大展示。
- 單擊圖形右上表徵圖  可以對該圖中展示的指標項設定相關報警規則。詳見[報警服務使用指南](#)。注意，表格和計量參考指標不支援報警設定。

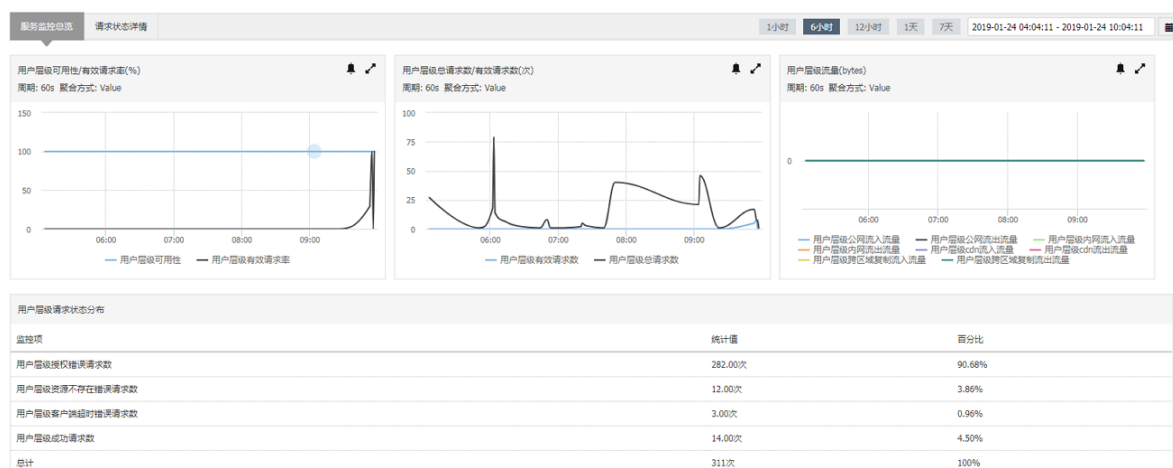
- 滑鼠按住圖形曲線區域拖放，可以進行時間範圍快速調整放大，單擊恢復縮放迴歸到拖放之前的時間範圍。



· 服務監控總覽

服務監控總覽頁面主要包括下面監控指標圖：

- 用戶層級可用性/有效請求率：包括可用性和有效請求率兩項指標。
- 用戶層級總請求數/有效請求數：包括總請求數和有效請求數兩項指標。
- 用戶層級流量：包括公網流出流量、公網流入流量、內網流出流量、內網流入流量、CDN流出流量、CDN流入流量、跨區域複製流出流量和跨區域複製流入流量八項指標。
- 用戶層級請求狀態分布：該表格中展示選定時間範圍內各個請求類型的個數以及佔比。



· 請求狀態詳情

請求狀態詳情是對請求狀態分布統計的一個具體監控，主要包括下面的監控指標圖：

- 用戶層級服務端錯誤請求數。
- 用戶層級服務端錯誤請求佔比。
- 用戶層級網路錯誤請求數。
- 用戶層級網路錯誤請求佔比。
- 用戶層級客戶端錯誤請求數：包括資源不存在錯誤請求數、授權錯誤請求數、客戶端逾時錯誤請求數和客戶端其他錯誤請求數四項指標。
- 用戶層級客戶端錯誤請求佔比：包括資源不存在錯誤請求佔比、授權錯誤請求佔比、客戶端逾時錯誤請求佔比和客戶端其他錯誤請求佔比四項指標。
- 用戶層級有效請求數：包括成功請求數和重新導向請求數兩項指標。
- 用戶層級有效請求佔比：包括成功請求佔比和重新導向請求佔比兩項指標。



Bucket列表

· Bucket列表資訊

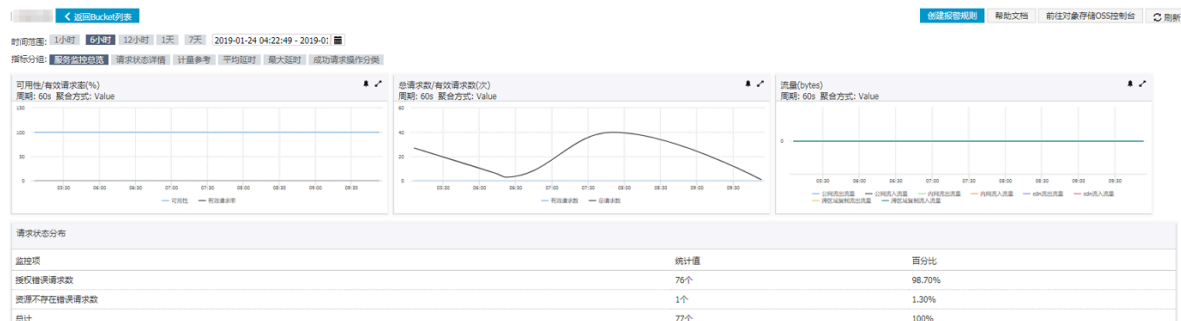
列表展現該帳號所擁有的Bucket的名稱、所屬地域、建立時間、當月計量數據統計資訊以及相關操作。

用戶概況 Bucket列表 报警規則								当月数据 (截止時間:2019.01.24 09:00:00)	
名称	地域	创建时间	存储大小	公网流出流量	Get类请求数	Put类请求数	操作		
 bucket-1	华北2 (北京)	2018-11-07 10:14:33	23.64MB	16.21MB	1547次	329次	监控图表 报警规则		
 bucket-2	华南1 (深圳)	2018-11-19 11:22:09	6.83MB	6.89MB	121次	33次	监控图表 报警规则		
 bucket-3	华南1 (深圳)	2018-11-08 17:19:05	1.33MB	0B	101次	30次	监控图表 报警规则		
 bucket-4	华东1 (杭州)	2018-11-08 14:14:43	2.90MB	624.27KB	483次	54次	监控图表 报警规则		
 bucket-5	华东1 (杭州)	2018-11-08 14:14:07	0B	0B	57次	2次	监控图表 报警规则		
 bucket-6	华北2 (北京)	2018-11-19 11:01:49	910.88KB	3.14MB	756次	76次	监控图表 报警规则		
 bucket-7	华北 5 (呼和浩特)	2019-01-18 14:18:22	0B	0B	7次	1次	监控图表 报警规则		
 bucket-8	英国 (伦敦)	2019-01-18 14:17:33	0B	0B	9次	1次	监控图表 报警规则		
 bucket-9	华北1 (青岛)	2019-01-18 14:17:51	0B	0B	15次	1次	监控图表 报警规则		
 bucket-10	华东2 (上海)	2019-01-18 14:18:40	0B	0B	10次	1次	监控图表 报警规则		
设置报警规则								共 13 条	

- 當月計量統計包括每個Bucket的儲存量、公網流出流量、Put類請求數和Get類請求數。
- 單擊監控圖表或者對應的Bucket名稱，能夠進入具體的Bucket監控視圖頁。
- 單擊報警規則，進入報警規則Tab頁，並且展現所有屬於該Bucket的報警規則。
- 通過上面的搜尋方塊能夠模糊比對快速找到具體的Bucket。
- 選中Bucket複選框，並單擊設定報警規則可以批量設定報警規則，詳見[報警服務使用指南](#)。

· Bucket層級監控視圖

單擊Bucket列表中具體的Bucket行中的監控圖表，就能進入對應的Bucket的監控視圖頁，如下圖：



Bucket監控視圖頁按指標分組進行展示監控圖，主要包含六個指標分組：

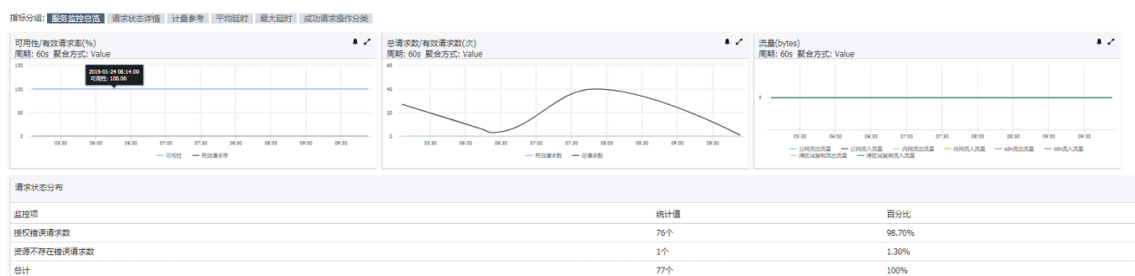
- ■ 服務監控總覽
- 請求狀態詳情
- 計量參考
- 平均延時
- 最大延時
- 成功請求操作分類

除了計量參考，所有的指標項都是分鐘等級彙總展示的。不同於用戶層級預設時間展現為最近1小時，Bucket層級的監控展示預設為6小時。單擊上方的返回Bucket列表能夠回到Bucket列表Tab頁。

- 服務監控總覽

該指標分組同用戶層級的服務監控總覽，只是從具體的Bucket進行監控，主要包括下面監控指標圖：

- 可用性/有效請求率：包括可用性和有效請求率兩項指標。
- 總請求數/有效請求數：包括總請求數和有效請求數兩項指標。
- 流量：包括公網流出流量、公網流入流量、內網流出流量、內網流入流量、cdn流出流量、cdn流入流量、跨區域複製流出流量和跨區域複製流入流量八項指標。
- 請求狀態分布：該表格中展示選定時間範圍內各個請求類型的個數以及佔比。



- 請求狀態詳情

該指標分組同用戶層級的請求狀態詳情，只是從具體的Bucket進行監控，主要包括下面監控指標圖：

■ 服務端錯誤請求數

■ 服務端錯誤請求佔比

■ 網路錯誤請求數

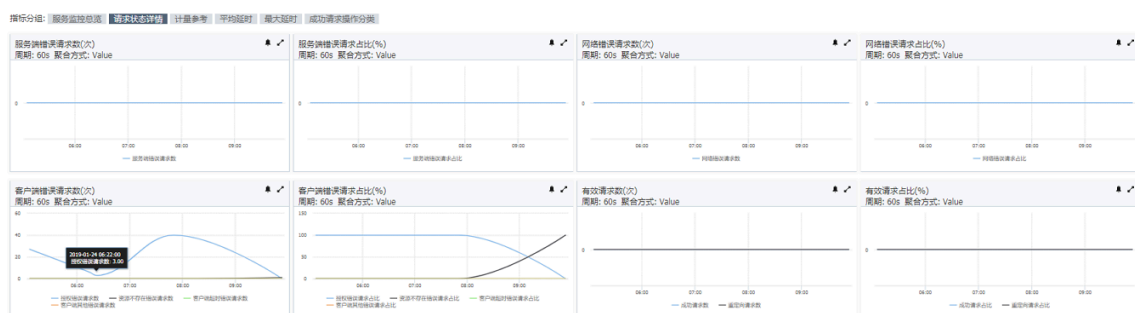
■ 網路錯誤請求佔比

■ 客戶端錯誤請求數：包括資源不存在錯誤請求數、授權錯誤請求數、客戶端逾時錯誤請求數和客戶端其他錯誤請求數四項指標。

■ 客戶端錯誤請求佔比：包括資源不存在錯誤請求佔比、授權錯誤請求佔比、客戶端逾時錯誤請求佔比和客戶端其他錯誤請求佔比四項指標。

■ 有效請求數：包括成功請求數和重新導向請求數兩項指標。

■ 有效請求佔比：包括成功請求佔比和重新導向請求佔比兩項指標。



- 計量參考

計量參考分組展示各個計量相關的指標資訊，以小時粒度收集展現，如下圖所示：



包含以下計量指標監控圖：

- 儲存大小
- 公網流出流量
- 計費請求數：包括Get類請求數和Put類請求數兩項指標項。

如果新建Bucket，需要到目前時間點的下一個整小時點才會採集到新數據，然後在半個小時內展示出來。

- 平均延時

該分組包含分API類型監控的各項平均延時指標，包含如下幾個指標圖：

- GetObject請求平均延時
- HeadObject請求平均延時
- PutObject請求平均延時
- PostObject請求平均延時
- AppendObject請求平均延時
- UploadPart請求平均延時
- UploadPartCopy請求平均延時

每個指標圖中都包含對應的平均E2E延時和平均伺服器延時，如下圖所示：



- 最大延時

該分組包含分API類型監控的各項最大延時指標，包含如下幾個指標圖：

- GetObject請求最大延時
- HeadObject請求最大延時
- PutObject請求最大延時
- PostObject請求最大延時
- AppendObject請求最大延時
- UploadPart請求最大延時
- UploadPartCopy請求最大延時

每個指標圖中都包含對應的最大E2E延時和最大伺服器延時，如下圖所示：



- 成功請求操作分類

該分組包含分API類型監控的各項成功請求數指標，包含如下幾個指標圖：

- GetObject成功請求
- HeadObject成功請求
- PutObject成功請求
- PostObject成功請求
- AppendObject成功請求
- UploadPart成功請求
- UploadPartCopy成功請求
- DeleteObject成功請求
- DeleteObjects成功請求

如下圖所示：



報警規則

報警規則Tab頁能夠展示和管理報警規則，如下圖所示：



報警規則頁的使用和相關說明請參見[使用報警服務](#)。

監控關注事項以及使用指導

監控關注點以及使用指南請參見[監視診斷和故障排除](#)。

13.3 使用報警服務

在介紹OSS監控服務控制台之前，請先閱讀Cloud Monitor提供的監控服務文檔，了解基本概念並進行報警連絡人和報警聯繫組的配置。

- [報警服務概覽](#)
- [報警連絡人和聯繫組](#)

因為OSS的報警規則是根據OSS監控項設定的，所以類似於OSS監控項的維度分類，將其分成兩個報警維度：用戶層級和Bucket層級。

報警規則頁

報警規則頁是OSS監控報警相關的規則管理頁面，您可以查看、修改、啟用、禁用和刪除對應的報警規則，而且能夠查看該對應報警規則的曆史報警情況。



- 單擊對應報警規則的修改，就可以對該報警規則進行修改。
- 單擊對應報警規則的刪除，就可以刪除該報警規則。選中多條報警規則，然後單擊表格最下方的刪除按鈕，可以大量刪除報警規則。
- 如果報警規則處於已啟用狀態，單擊該報警規則的禁用，可以禁用該報警規則，報警規則失效，用戶不能再收到對應的警示資訊。選中多條報警規則，然後單擊表格最下方的禁用按鈕，可以批量禁用報警規則。
- 如果報警規則處於已禁用狀態，單擊該報警規則的啟用，可以啟用該報警規則，報警規則重新生效，能檢測並發出對應的警示資訊。選中多條報警規則，然後單擊表格最下方的啟用按鈕，可以批量啟用報警規則。
- 單擊對應報警規則的報警曆史，可以查看該報警規則曆史發生的所有的警示情況。

appendObject请求
返回
禁用 删除

1 小时 2 小时 4 小时 6 小时 12 小时 1天 3天 7天
2019-02-14 15:39:00 - 2019-02-15 15:39:00

产品类型	故障资源	发生时间	持续时间	规则名称	通知方式	状态	通知对象	报警回调	操作
恭喜您，最近没有报警历史									

這裡解釋一下報警曆史的相關概念：

- 報警曆史指的是該報警規則的狀態變化曆史，比如從正常變成警示狀態，是一個狀態變化；從警示變成正常也是狀態變化。另外，還有一個特殊的狀態變化：通道沉默。
- 當通知對象為通道沉默時，表示該報警規則引發警示之後的24小時內一直滿足報警觸發狀態(即一直在警示，沒有恢復到正常狀態)。這個時候系統是不會再向用戶發送警示資訊進行通知的，一直持續到24小時之後才會有新的報警資訊發送到通知對象。
- 報警曆史資訊能夠保存一個月，即一個月之前的警示資訊會被自動清理。查詢時一次最多隻能查詢3天的數據，但不支援查詢31天前的數據。

另外，單擊具體報警規則的通知對象旁邊的查看，可以顯示該通知對象（報警聯繫組）的成員以及每個成員接收警示資訊的方式（SMS、郵箱或者旺旺），如下圖所示：

報警联系人
×

lianxiren

联系人	短信	邮箱	旺旺	钉钉机器人
陈				

云账号报警联系人

联系人	短信	邮箱	旺旺	钉钉机器人
陈				

确定

查看報警規則

快速定位報警規則

根據報警規則頁中的下面控制項資訊能夠快速定位到被搜索的報警規則：

- 報警維度下拉框：全部和Bucket層級。當選項為全部時，顯示所有用戶層級和Bucket層級的報警規則。

The screenshot shows the '报警规则' (Alarm Rules) tab selected. Below the tabs, there is a dropdown menu labeled '报警维度' (Alarm Dimension) with the option '全部' (All) selected. To the right of this dropdown is a red asterisk and another dropdown menu labeled '请选择' (Please select).

- Bucket下拉框：當報警維度下拉框為 Bucket 層級時，這裡可以羅列該帳號下所有的Bucket。選擇對應的Bucket，可以展示屬於該Bucket的所有報警規則。

The screenshot shows the '报警规则' (Alarm Rules) tab selected. Below the tabs, there is a dropdown menu labeled '报警维度' (Alarm Dimension) with the option 'Bucket层级' (Bucket Level) selected. To the right of this dropdown is a red asterisk and another dropdown menu labeled '请选择' (Please select). The '请选择' dropdown is highlighted with a red border.

- 監控項下拉框：羅列所有的OSS的監控項，包括用戶層級和Bucket層級的監控項。當選項為監控項時，顯示用戶層級或者Bucket層級所有監控項的報警規則。
 - 報警狀態下拉框：報警狀態、警示和正常。當為報警狀態時顯示所有狀態的報警規則。
 - 啟用狀態下拉框：啟用狀態、已禁用和已啟用。當為啟用狀態時，顯示所有狀態的報警規則。
- 查看全部報警規則

單擊報警規則Tab頁進入報警規則也預設顯示全部報警規則。將報警維度下拉框設定為全部，即能顯示全部報警規則。配合監控項、報警狀態和啟用狀態這三個下拉框能夠更精確定位全部維度下的報警規則。

查看特定Bucket的報警規則

如果需要查看某個特定的Bucket的報警規則，需要設定報警維度下拉框為Bucket層級，然後在Bucket下拉框中選擇對應的Bucket名稱。通過Bucket列表中的對應Bucket的報警規則進入的報警頁，就能顯示該Bucket的所有的報警規則。配合監控項、報警狀態和啟用狀態這三個，即能顯示該報警維度下更精確的報警規則。

查看與具體監控項相關的報警規則

設定監控項下拉框為具體的某項監控項，就能顯示對應監控項的所有報警規則。

- 查看指定報警狀態的報警規則

設定報警狀態下拉框為指定報警狀態，如警示，就能展示所有處於該警示狀態的報警規則。

- 查看指定啟用狀態的報警規則

設定啟動狀態下拉框為指定啟用狀態，如已禁用，就能顯示已經被禁用的所有的報警規則。

添加報警規則

在Bucket列表Tab頁選中指定的Bucket之後單擊設定報警規則按鈕，或者在用戶概況Tab頁亦或某個Bucket監控視圖頁中的監控圖形上的  都可以開啟批量設定報警規則視窗，進行報警規則的

設定。

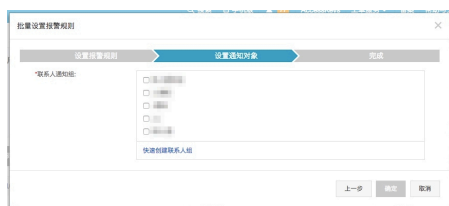
下面示範一下報警規則設定的過程（注意，下面出現的名詞概念請詳見Cloud Monitor[報警服務概覽](#)。）

1. 首先進入如下設定報警規則步驟：



- 報警維度: 提示用戶目前設定的報規則屬於哪個維度監控。如果是Bucket層級，則會提示目前為哪個具體的Bucket設定報警規則，圖例中顯示的是為用戶層級。
- 監控項: 該多選框顯示屬於該報警維度所有的監控項選項。可以通過快速搜尋框迅速定位：
- 統計周期: 間隔多久進行一次數據統計，預設為5min。
- 連續次數: 指連續幾個統計周期監控項的值持續超過閾值後觸發報警。
- 統計方法: 指定該監控項中的具體的統計指標，OSS監控服務的統計方法均為監控值。
- 添加報警規則按鈕: 單擊按鈕可以設定更多監控項的報警規則。
- 刪除: 單擊對應報警規則後的刪除，可以刪除該報警規則。

2. 設定完成之後，單擊下一步，進入設定通知對象步驟，如下圖：



如果事先已經按照**報警連絡人**和**聯繫組**設定好報警聯繫組，那麼就會像圖例中展示出已經存在的報警連絡人群組。如果沒有設定過報警連絡人群組，則單擊**快建立連絡人群組**，按照提示建立即可。

3. 單擊確定，完成報警規則的設定。



· Bucket列表添加報警規則

在Bucket列表Tab頁中可以為多個Bucket大量新增相同的報警規則。選中需要配置報警規則的Bucket，單擊設定報警規則按鈕，進入報警規則設定頁面。



说明：

批量設定時報警維度是確定的，監控項都是Bucket層級的。

· 具體監控圖表中添加報警規則

在用戶概況Tab頁或者具體Bucket的監控視圖頁中，單擊具體的監控圖右上方的可以為該監控圖用的監控指標項設定報警規則。



说明：

通過單擊監控圖的報警表徵圖進入報警規則頁面時，報警維度都是確定的，而且只能設定該圖中的監控項的報警規則。

注意事項

目前屬於某個Bucket的報警規則存在性並沒有與該Bucket的存在性強關聯，即如果刪除了某個Bucket，屬於這個Bucket的報警規則依賴存在。建議用戶在刪除Bucket之前先刪除對應的報警規則。

13.4 訪問監控數據

本章內容為OpenAPI（或者Cloud Monitor提供的SDK）查詢OSS監控服務指標數據的使用提供相關參數依據。

關於Project

OSS監控服務指標項的數據都使用相同的Project名稱：acs_oss。

使用Java SDK設定程式碼範例如下：

```
QueryMetricRequest request = new QueryMetricRequest ();  
request . setProject (" acs_oss ");
```

關於StartTime和EndTime

Cloud Monitor的時間參數取值範圍採用左開右閉的形式，即(StartTime, EndTime]，即StartTime的數據不會被查詢到，而EndTime的數據會被查詢到。

另外，Cloud Monitor數據保留時間為31天，設定的StartTime和EndTime的時間間距不能大於31天，並且不能夠查詢31天以前的數據。

其他時間參數資訊詳見Cloud MonitorAPI參考。

使用Java SDK設定程式碼範例如下：

```
request . setStartTime (" 2016 - 05 - 15    08 : 00 : 00 ");  
request . setEndTime  (" 2015 - 05 - 15    09 : 00 : 00 ");
```

關於Dimensions

OSS監控服務根據用戶使用場景，將監控項分為用戶和Bucket兩個不同的層級進行監控。針對不同的層級監控數據的訪問，Dimensions不同：

- 用戶層級數據不需要設定Dimensions。

- Bucket層級數據的Dimensions設定為：

```
{" BucketName ": " your_bucket_name "}
```

其中，your_bucket_name為屬於該用戶的某個需要查詢的Bucket名稱。

注意，Dimensions是一個JSON字元串，OSS監控指標的Dimensions只有一對Key-Value。

使用Java SDK設定程式碼範例如下：

```
request . setDimensions ("{" BucketName ":" your_bucket_name "}" );
```

關於Period

OSS監控指標除了計量指標，其他所有的指標的彙總粒度均為60s。計量指標的彙總粒度為3600s。

使用Java SDK設定程式碼範例如下：

```
request . setPeriod (" 60 " );
```

關於Metric

[OSS監控指標參考](#)中詳細介紹的各項指標項對應的Metric名稱和層級如下表所示：

Metric	對應指標項名稱	單位	層級
UserAvailability	用戶層級可用性	%	用戶層級
UserRequestValidRate	用戶層級有效請求率	%	用戶層級
UserTotalRequestCount	用戶層級總請求數	次數	用戶層級
UserValidRequestCount	用戶層級有效請求數	次數	用戶層級
UserInternetSend	用戶層級公網流出流量	位元組	用戶層級
UserInternetRecv	用戶層級公網流入流量	位元組	用戶層級
UserIntranetSend	用戶層級內網流出流量	位元組	用戶層級
UserIntranetRecv	用戶層級內網流入流量	位元組	用戶層級
UserCdnSend	用戶層級cdn流出流量	位元組	用戶層級
UserCdnRecv	用戶層級cdn流入流量	位元組	用戶層級
UserSyncSend	用戶層級跨區域複製流出流量	位元組	用戶層級
UserSyncRecv	用戶層級跨區域複製流入流量	位元組	用戶層級

Metric	對應指標項名稱	單位	層級
UserServerErrorCount	用戶層級服務端錯誤請求總數	次數	用戶層級
UserServerErrorRate	用戶層級服務端錯誤請求佔比	%	用戶層級
UserNetworkErrorCount	用戶層級網路錯誤請求總數	次數	用戶層級
UserNetworkErrorRate	用戶層級網路錯誤請求佔比	%	用戶層級
UserAuthorizationErrorCount	用戶層級客戶端授權錯誤請求總數	次數	用戶層級
UserAuthorizationErrorRate	用戶層級客戶端授權錯誤請求佔比	%	用戶層級
UserResourceNotFoundErrorCount	用戶層級客戶端資源不存在錯誤請求總數	次數	用戶層級
UserResourceNotFoundErrorRate	用戶層級客戶端資源不存在錯誤請求佔比	%	用戶層級
UserClientTimeoutErrorCount	用戶層級客戶端逾時錯誤請求總數	次數	用戶層級
UserClientOtherErrorRate	用戶層級客戶端逾時錯誤請求佔比	%	用戶層級
UserClientOtherErrorCount	用戶層級客戶端其他錯誤請求總數	次數	用戶層級
UserClientOtherErrorRate	用戶層級客戶端其他錯誤請求佔比	%	用戶層級
UserSuccessCount	用戶層級成功請求總數	次數	用戶層級
UserSuccessRate	用戶層級成功請求佔比	%	用戶層級
UserRedirectCount	用戶層級重新導向請求總數	次數	用戶層級
UserRedirectRate	用戶層級重新導向請求佔比	%	用戶層級
Availability	可用性	%	Bucket層級
RequestValidRate	有效請求率	%	Bucket層級
TotalRequestCount	總請求數	次數	Bucket層級

Metric	對應指標項名稱	單位	層級
ValidRequestCount	有效請求數	次數	Bucket層級
InternetSend	公網流出流量	位元組	Bucket層級
InternetRecv	公網流入流量	位元組	Bucket層級
IntranetSend	內網流出流量	位元組	Bucket層級
IntranetRecv	內網流入流量	位元組	Bucket層級
CdnSend	cdn流出流量	位元組	Bucket層級
CdnRecv	cdn流入流量	位元組	Bucket層級
SyncSend	跨區域複製流出流量	位元組	Bucket層級
SyncRecv	跨區域複製流入流量	位元組	Bucket層級
ServerErrorCount	服務端錯誤請求總數	次數	Bucket層級
ServerErrorRate	服務端錯誤請求佔比	%	Bucket層級
NetworkErrorCount	網路錯誤請求總數	次數	Bucket層級
NetworkErrorRate	網路錯誤請求佔比	%	Bucket層級
AuthorizationErrorCount	客戶端授權錯誤請求總數	次數	Bucket層級
AuthorizationErrorRate	客戶端授權錯誤請求佔比	%	Bucket層級
ResourceNotFoundErrorCount	客戶端資源不存在錯誤請求總數	次數	Bucket層級
ResourceNotFoundErrorRate	客戶端資源不存在錯誤請求佔比	%	Bucket層級
ClientTimeoutErrorCount	客戶端逾時錯誤請求總數	次數	Bucket層級
ClientOtherErrorRate	客戶端逾時錯誤請求佔比	%	Bucket層級
ClientOtherErrorCount	客戶端其他錯誤請求總數	次數	Bucket層級
ClientOtherErrorRate	客戶端其他錯誤請求佔比	%	Bucket層級
SuccessCount	成功請求總數	次數	Bucket層級
SuccessRate	成功請求佔比	%	Bucket層級
RedirectCount	重新導向請求總數	次數	Bucket層級

Metric	對應指標項名稱	單位	層級
RedirectRate	重新導向請求佔比	%	Bucket層級
GetObjectE2eLatency	GetObject請求平均E2E延時	毫秒	Bucket層級
GetObjectServerLatency	GetObject請求平均伺服器延時	毫秒	Bucket層級
MaxGetObjectE2eLatency	GetObject請求最大E2E延時	毫秒	Bucket層級
MaxGetObjectServerLatency	GetObject請求最大伺服器延時	毫秒	Bucket層級
HeadObjectE2eLatency	HeadObject請求平均E2E延時	毫秒	Bucket層級
HeadObjectServerLatency	HeadObject請求平均伺服器延時	毫秒	Bucket層級
MaxHeadObjectE2eLatency	HeadObject請求最大E2E延時	毫秒	Bucket層級
MaxHeadObjectServerLatency	HeadObject請求最大伺服器延時	毫秒	Bucket層級
PutObjectE2eLatency	PutObject請求平均E2E延時	毫秒	Bucket層級
PutObjectServerLatency	PutObject請求平均伺服器延時	毫秒	Bucket層級
MaxPutObjectE2eLatency	PutObject請求最大E2E延時	毫秒	Bucket層級
MaxPutObjectServerLatency	PutObject請求最大伺服器延時	毫秒	Bucket層級
PostObjectE2eLatency	PostObject請求平均E2E延時	毫秒	Bucket層級
PostObjectServerLatency	PostObject請求平均伺服器延時	毫秒	Bucket層級
MaxPostObjectE2eLatency	PostObject請求最大E2E延時	毫秒	Bucket層級
MaxPostObjectServerLatency	PostObject請求最大伺服器延時	毫秒	Bucket層級
AppendObjectE2eLatency	AppendObject請求平均E2E延時	毫秒	Bucket層級

Metric	對應指標項名稱	單位	層級
AppendObjectServerLatency	AppendObject請求平均伺服器延時	毫秒	Bucket層級
MaxAppendObjectE2eLatency	AppendObject請求最大E2E延時	毫秒	Bucket層級
MaxAppendObjectServerLatency	AppendObject請求最大伺服器延時	毫秒	Bucket層級
UploadPartE2eLatency	UploadPart請求平均E2E延時	毫秒	Bucket層級
UploadPartServerLatency	UploadPart請求平均伺服器延時	毫秒	Bucket層級
MaxUploadPartE2eLatency	UploadPart請求最大E2E延時	毫秒	Bucket層級
MaxUploadPartServerLatency	UploadPart請求最大伺服器延時	毫秒	Bucket層級
UploadPartCopyE2eLatency	UploadPartCopy請求平均E2E延時	毫秒	Bucket層級
UploadPartCopyServerLatency	UploadPartCopy請求平均伺服器延時	毫秒	Bucket層級
MaxUploadPartCopyE2eLatency	UploadPartCopy請求最大E2E延時	毫秒	Bucket層級
MaxUploadPartCopyServerLatency	UploadPartCopy請求最大伺服器延時	毫秒	Bucket層級
GetObjectCount	GetObject成功請求數	次數	Bucket層級
HeadObjectCount	HeadObject成功請求數	次數	Bucket層級
PutObjectCount	PutObject成功請求數	次數	Bucket層級
PostObjectCount	PostObject成功請求數	次數	Bucket層級
AppendObjectCount	AppendObject成功請求數	次數	Bucket層級
UploadPartCount	UploadPart成功請求數	次數	Bucket層級
UploadPartCopyCount	UploadPartCopy成功請求數	次數	Bucket層級

Metric	對應指標項名稱	單位	層級
DeleteObjectCount	DeleteObject成功請求數	次數	Bucket層級
DeleteObjectsCount	DeleteObjects成功請求數	次數	Bucket層級

計量類指標的Metric如下表所示，注意彙總粒度為3600s。

Metric	對應指標項名稱	單位	層級
MeteringStorageUtilization	儲存大小	位元組	若設定Dimensions，則為Bucket層級；如果不設定，那麼返回的指標為整個用戶層級數據。
MeteringGetRequest	Get類請求數	次數	若設定Dimensions，則為Bucket層級；如果不設定，那麼返回的指標為整個用戶層級數據。
MeteringPutRequest	Put類請求數	次數	若設定Dimensions，則為Bucket層級；如果不設定，那麼返回的指標為整個用戶層級數據。
MeteringInternetTX	公網流出計量流量	位元組	若設定Dimensions，則為Bucket層級；如果不設定，那麼返回的指標為整個用戶層級數據。
MeteringCdnTX	cdn流出計量流量	位元組	若設定Dimensions，則為Bucket層級；如果不設定，那麼返回的指標為整個用戶層級數據。
MeteringSyncRX	跨區域複製流入計量流量	位元組	若設定Dimensions，則為Bucket層級；如果不設定，那麼返回的指標為整個用戶層級數據。

使用Java SDK設定程式碼範例如下：

```
request.setMetric("UserAvailability");
```

13.5 監控指標參考

根據用戶使用場景，將OSS的指標分為用戶層級和Bucket（儲存空間）層級兩個層級維度進行監控。

另外，為了更好地觀察監控數據以及匹配計費策略，除了一般監控項的時間序指標外，OSS對現有的監控指標項進行統計分析，提供了一段時間內的統計指標，如請求狀態分布統計和當月計量統計。

除了計量指標和統計指標，所有的指標（時間序指標）都是分鐘等級的數據匯總（如求和、求最大值或者求均值等等）。而計量指標是按小時的數據進行匯總的時間序指標。

用戶層級指標

用戶層級指標是指從用戶的賬戶等級對OSS系統使用的總體情況進行監控的指標資訊，是對該賬戶下的所有的Bucket相關監控數據的匯總。其中包括當月計量統計、服務監控總覽和請求狀態詳情三個方面。

- 服務監控總覽

服務監控總覽指標屬於基礎服務指標，具體指標項詳見下表。注意，下面所有的指標都是在用戶層級監控的。

服務監控總覽指標名稱	單位	描述
可用性	%	儲存服務的系統可用性衡量指標。通過公式 $1 - \frac{\text{服務端錯誤請求（返回狀態碼為5xx）}}{\text{總請求數}}$ 佔總請求的百分比而得到
有效請求率	%	有效請求佔總請求數的百分比，有效請求的介紹見下面說明
總請求數	次數	被OSS服務端接收並處理的請求總數
有效請求數	次數	返回狀態碼為2xx和3xx的請求總數
公網流出流量	位元組	通過互連網網路的下行流量

服務監控總覽指標名稱	單位	描述
公網流入流量	位元組	通過互連網網路的上行流量
內網流出流量	位元組	通過服務系統內部網路的下行流量
內網流入流量	位元組	通過服務系統內部網路的上行流量
cdn流出流量	位元組	開通cdn加速服務之後，通過cdn產生的下行流量，即回源流量
cdn流入流量	位元組	開通cdn加速服務之後，通過cdn產生的上行流量
跨區域複製流出流量	位元組	開通跨區域複製功能之後，數據複製過程產生的下行流量
跨區域複製流入流量	位元組	開通跨區域複製功能之後，數據複製過程產生的上行流量

除了以上具體的監控指標，還提供一段時間內的請求狀態分布統計，主要是根據返回的狀態碼或者OSS錯誤碼進行分類的請求的統計資訊（被觀察時間段內的請求次數總和以及佔比），相關的監控指標項資訊詳見以下請求狀態詳情的介紹。

· 請求狀態詳情

請求狀態詳情指標是指根據請求返回狀態碼或者OSS錯誤碼進行分類的請求的監控資訊，屬於基礎服務指標，具體指標項詳見下表。注意，下面所有的指標都是在用戶層級監控的。

請求狀態詳情指標名稱	單位	描述
服務端錯誤請求總數	次數	返回狀態碼為5xx的系統級錯誤請求總數
服務端錯誤請求佔比	%	服務端錯誤請求總數佔總請求數的百分比
網路錯誤請求總數	次數	HTTP狀態碼為499的請求總數，
網路錯誤請求佔比	%	網路錯誤請求數佔總請求數的百分比
客戶端授權錯誤請求總數	次數	返回狀態碼403的請求總數
客戶端授權錯誤請求佔比	%	授權錯誤請求數佔總請求數的百分比

請求狀態詳情指標名稱	單位	描述
客戶端資源不存在錯誤請求總數	次數	返回狀態碼為404的請求總數
客戶端資源不存在錯誤請求佔比	%	資源不存在錯誤請求數佔總請求數百分比
客戶端逾時錯誤請求總數	次數	返回狀態碼為408或者返回的OSS錯誤碼為RequestTimeout的請求總數
客戶端逾時錯誤請求佔比	%	網路錯誤請求總數佔總請求數的百分比
客戶端其他錯誤請求總數	次數	除了以上提到的客戶端錯誤請求之外的其他返回狀態碼為4xx的請求總數
客戶端其他錯誤請求佔比	%	客戶端其他錯誤請求數佔總請求數的百分比
成功請求總數	次數	返回狀態碼為2xx的請求總數
成功請求佔比	%	成功請求數佔總請求數的百分比
重新導向請求總數	次數	返回狀態碼為3xx的請求總數
重新導向請求佔比	%	重新導向請求數佔總請求數的百分比

· 當月計量統計

當月計量統計指標是指從當月的1號0點開始，到當月計量採集截止時間為止，這段時間內計量指標的統計數據。

目前統計的計量指標如下：

當月計量統計指標名稱	單位	描述
儲存大小	位元組	在計量採集截止時間前屬於該用戶的所有Bucket佔用的儲存總大小
公網流出流量	位元組	從本月1號0點開始累積到計量採集截止時間為止，用戶所使用的公網流出流量的總和。

當月計量統計指標名稱	單位	描述
Put類請求數	次數	從本月1號0點開始累積到計量採集截止時間為止，用戶所使用的所有Put類請求的總和。
Get類請求數	次數	從本月1號0點開始累積到計量採集截止時間為止，用戶所使用的所有Get類請求的總和。

Bucket層級指標

Bucket層級指標是指對具體的儲存空間的OSS操作情況進行監控的指標資訊，具有更強的業務場景，所以除了類似從用戶層面可以監控的服務監控總覽和請求狀態詳情這些基礎服務指標項和當月計量統計之外，還有計量參考、延時和成功請求操作分類等計量指標和性能指標。

- 服務監控總覽

監控項指標含義同用戶層級的服務監控總覽，從具體的Bucket進行監控。

- 請求狀態詳情

監控項指標含義同用戶層級的請求狀態詳情，從具體的Bucket進行監控。

- 當月計量統計

統計方式同用戶層級的當月計量統計，從具體的Bucket資源使用方式進行統計。

當月計量統計指標名稱	單位	描述
儲存大小	位元組	在計量採集截止時間前該Bucket佔用的儲存大小
公網流出流量	位元組	從本月1號0點開始累積到計量採集截止時間為止，該Bucket的公網流出流量的總和。
Put類請求數	次數	從本月1號0點開始累積到計量採集截止時間為止，該Bucket的所有Put類請求的總和。
Get類請求數	次數	從本月1號0點開始累積到計量採集截止時間為止，該Bucket的所有Get類請求的總和。

· 計量參考

計量指標的時間序監控，具體如下：

當月計量統計指標名稱	單位	描述
儲存大小	位元組	該Bucket每小時使用的平均儲存大小
公網流出流量	位元組	該Bucket每小時的公網流出流量的總和
Put類請求數	次數	該Bucket每小時的Put類請求的總和
Get類請求數	次數	該Bucket每小時的Get類請求的總和

· 延時

請求延時是系統性能的直觀反映。監控服務提供了分鐘等級的平均延時和最大延時兩類指標，分別反映系統平均響應能力和系統抖動情況。並且根據OSS API請求操作類型進行分類，更細粒度地反應系統應對不同操作的性能狀況。目前只對關於Bucket的操作並且涉及數據操作（不包含對meta操作）的API進行監控。

另外，延時監控指標分別從E2E和伺服器兩條不同的鏈路進行收集，便於分析性能熱點以及環境問題，其中：

- E2E延時是指向OSS系統發出的成功請求的端到端延隔時間，包括在OSS系統中讀取請求、發送響應以及接受響應確認所需的處理時間。
- 伺服器延時是指OSS系統成功處理請求所使用的延隔時間，不包括E2E延時中的網路延隔時間。

注意，性能相關指標都是對成功請求（返回狀態碼為2xx）進行的監控。

具體的監控指標項如下表：

延時指標名稱	單位	描述
GetObject請求平均E2E延時	毫秒	請求API為GetObject的成功請求的平均端到端延時
GetObject請求平均伺服器延時	毫秒	請求API為GetObject的成功請求的平均伺服器延時
GetObject請求最大E2E延時	毫秒	請求API為GetObject的成功請求的最大端到端延時
GetObject請求最大伺服器延時	毫秒	請求API為GetObject的成功請求的最大伺服器延時

延時指標名稱	單位	描述
HeadObject請求平均E2E延時	毫秒	請求API為HeadObject的成功請求的平均端到端延時
HeadObject請求平均伺服器延時	毫秒	請求API為HeadObject的成功請求的平均伺服器延時
HeadObject請求最大E2E延時	毫秒	請求API為HeadObject的成功請求的最大端到端延時
HeadObject請求最大伺服器延時	毫秒	請求API為HeadObject的成功請求的最大伺服器延時
PutObject請求平均E2E延時	毫秒	請求API為PutObject的成功請求的平均端到端延時
PutObject請求平均伺服器延時	毫秒	請求API為PutObject的成功請求的平均伺服器延時
PutObject請求最大E2E延時	毫秒	請求API為PutObject的成功請求的最大端到端延時
PutObject請求最大伺服器延時	毫秒	請求API為PutObject的成功請求的最大伺服器延時
PostObject請求平均E2E延時	毫秒	請求API為PostObject的成功請求的平均端到端延時
PostObject請求平均伺服器延時	毫秒	請求API為PostObject的成功請求的平均伺服器延時
PostObject請求最大E2E延時	毫秒	請求API為PostObject的成功請求的最大端到端延時
PostObject請求最大伺服器延時	毫秒	請求API為PostObject的成功請求的最大伺服器延時
AppendObject請求平均E2E延時	毫秒	請求API為AppendObject的成功請求的平均端到端延時
AppendObject請求平均伺服器延時	毫秒	請求API為AppendObject的成功請求的平均伺服器延時
AppendObject請求最大E2E延時	毫秒	請求API為AppendObject的成功請求的最大端到端延時
AppendObject請求最大伺服器延時	毫秒	請求API為AppendObject的成功請求的最大伺服器延時
UploadPart請求平均E2E延時	毫秒	請求API為UploadPart的成功請求的平均端到端延時

延時指標名稱	單位	描述
UploadPart請求平均伺服器延時	毫秒	請求API為UploadPart的成功請求的平均伺服器延時
UploadPart請求最大E2E延時	毫秒	請求API為UploadPart的成功請求的最大端到端延時
UploadPart請求最大伺服器延時	毫秒	請求API為UploadPart的成功請求的最大伺服器延時
UploadPartCopy請求平均E2E延時	毫秒	請求API為UploadPartCopy的成功請求的平均端到端延時
UploadPartCopy請求平均伺服器延時	毫秒	請求API為UploadPartCopy的成功請求的平均伺服器延時
UploadPartCopy請求最大E2E延時	毫秒	請求API為UploadPartCopy的成功請求的最大端到端延時
UploadPartCopy請求最大伺服器延時	毫秒	請求API為UploadPartCopy的成功請求的最大伺服器延時

· 成功請求操作分類

配合延時監控，成功請求的監控一定程度上反應了系統處理訪問請求的能力。目前只監控關於Bucket的操作中涉及數據操作的API。詳細的指標項如下：

成功請求操作分類指標名稱	單位	描述
GetObject成功請求數	次數	請求API為GetObject的成功請求數
HeadObject成功請求數	次數	請求API為HeadObject的成功請求數
PutObject成功請求數	次數	請求API為PutObject的成功請求數
PostObject成功請求數	次數	請求API為PostObject的成功請求數
AppendObject成功請求數	次數	請求API為AppendObject的成功請求數
UploadPart成功請求數	次數	請求API為UploadPart的成功請求數
UploadPartCopy成功請求數	次數	請求API為UploadPartCopy的成功請求數
DeleteObject成功請求數	次數	請求API為DeleteObject的成功請求數

成功請求操作分類指標名稱	單位	描述
DeleteObjects成功請求數	次數	請求API為DeleteObjects的成功請求數

13.6 監控、診斷和故障排除

相對於傳統應用程式，開發雲端應用雖然降低了用戶在基礎設施搭建、運維等方面的成本，但卻增大了監控、診斷和故障排查的難度。OSS儲存服務為您提供了豐富的監控和日誌資訊，幫助您深刻洞察程式行為，及時發現並快速定位問題。

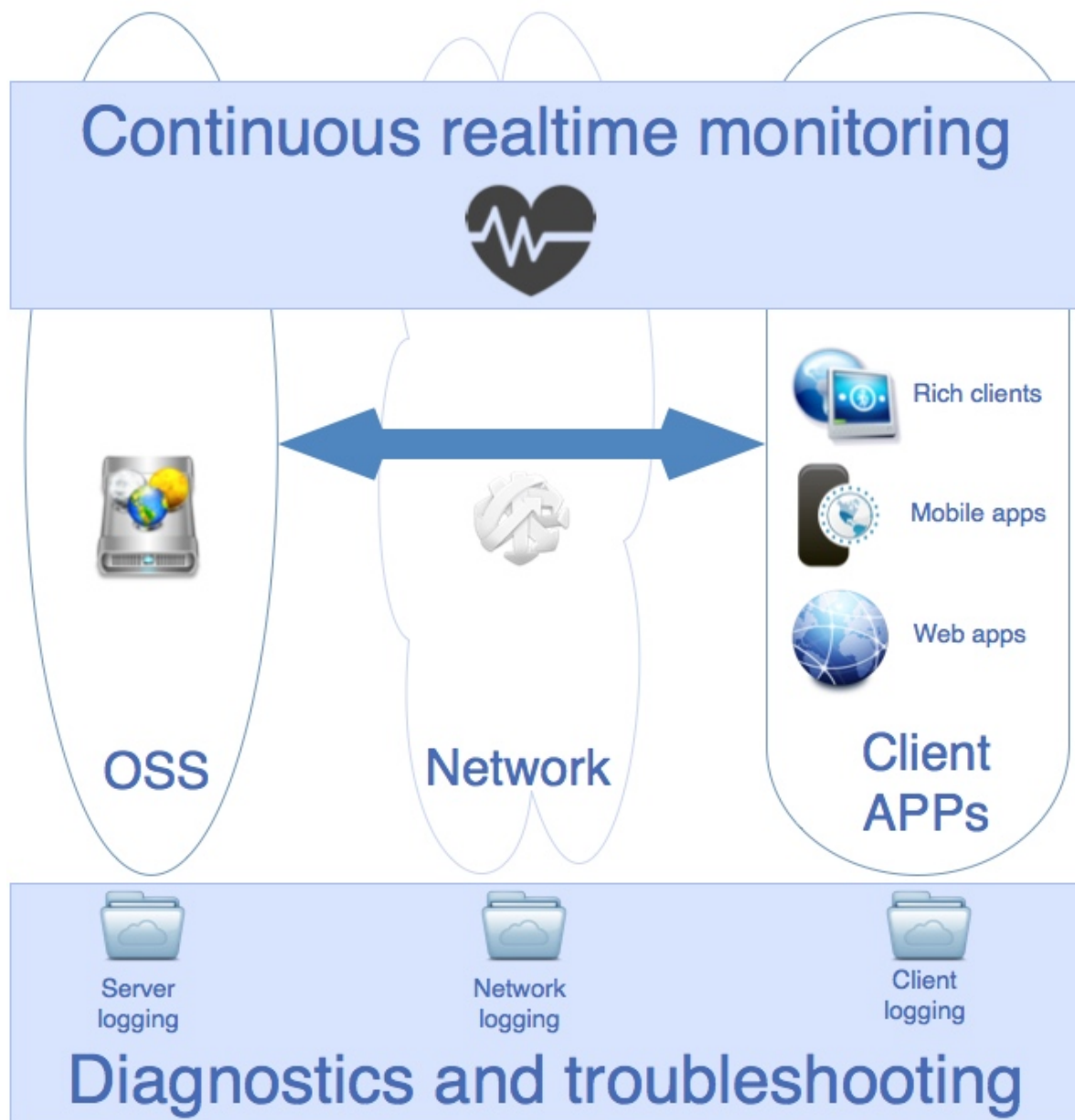
概述

本文主要描述如何使用OSS監控服務、日誌記錄功能以及其他第三方工具來監控、診斷和排查應用業務使用OSS儲存服務時遇到的相關問題，幫助您達到如下目標：

- 即時監控OSS儲存服務的健全狀態和性能，並及時報警通知。
- 獲取有效方法和工具來定位問題。
- 根據相關問題的處理和操作指南，快速解決與OSS相關的問題。

本文包括如下內容：

- [服務監控](#)，介紹如何使用OSS監控服務持續監控OSS儲存服務的健全狀態和性能。
- [跟蹤診斷](#)，介紹如何使用OSS監控服務和logging記錄功能診斷問題；另外，還介紹如何關聯各種記錄檔中的相關資訊進行跟蹤診斷。
- [故障排除](#)，提供常見的問題場景和故障排除方法。

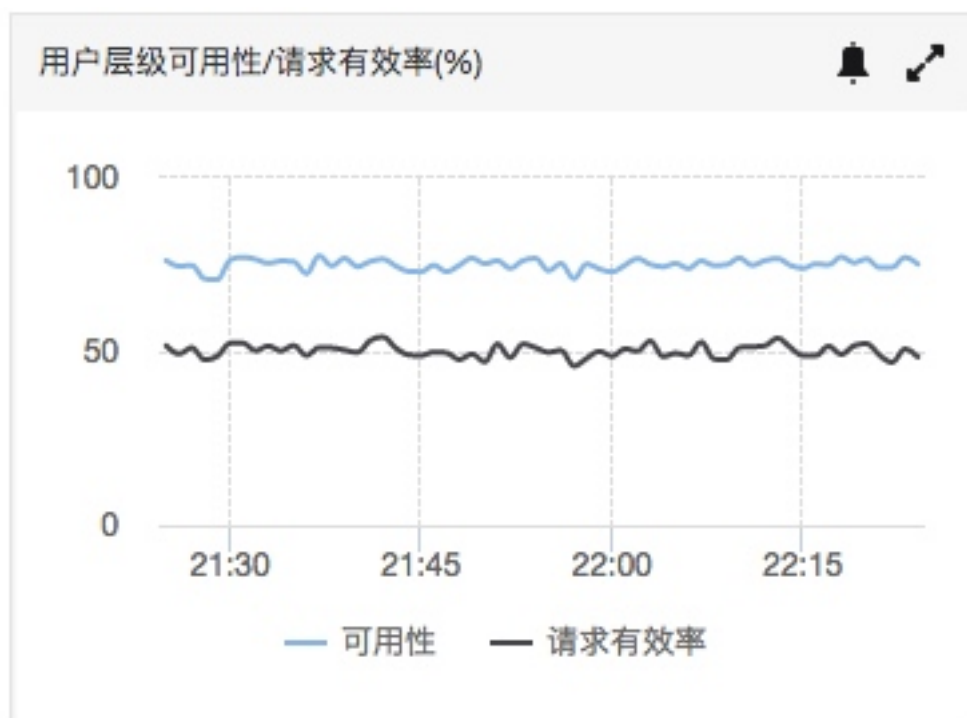


服務監控

監視總體健全狀態

- 可用性和有效請求率

可用性和有效請求率是有關系統穩定性和用戶是否正確使用系統的最重要指標，指標小於100%說明某些請求失敗。



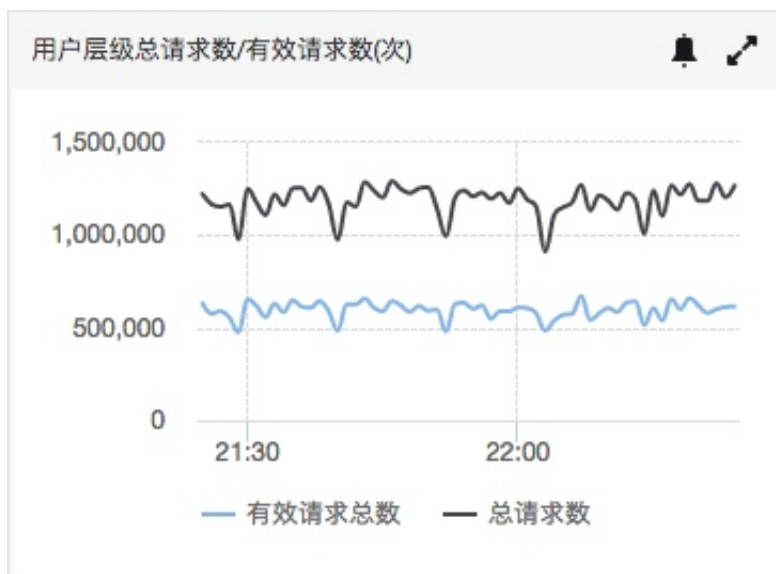
可能因為一些系統優化因素出現暫時性的低於100%，比如為了負載平衡而出現分區遷移，此時OSS的SDK能夠提供相關的重試機制無縫處理這類間歇性的失敗情況，使得業務端無感知。

另外，對於有效請求率低於100%的情況，用戶需要根據自己的使用方式進行分析，可以通過請求分布統計或者請求狀態詳情確定錯誤請求的具體類型，[跟蹤診斷](#)確定原因，並[故障排除](#)。當然，對於一些業務場景，出現有效請求率低於100%是符合預期的，比如，用戶需要先check訪問的object是否存在，然後根據object的存在性採取一定的操作，如果object不存在，檢測object存在性的讀取請求必然會收到404錯誤（資源不存在錯誤），那麼必然會導致該指標項低於100%。

對於系統可用性指標要求較高的業務，可以設定其報警規則，當低於預期閾值時，進行報警。

- 總請求數和有效請求數

該指標從總訪問量角度來反應系統運行狀態。當有效請求數不等於總請求數時表明某些請求失敗。



用戶可以關注總請求數或者有效請求數的波動狀況，特別是突然上升或者下降的情況，需要進行跟進調查，可以通過設定報警規則進行及時通知，對於存在周期性狀況的業務，可以設定為環比報警規則（環比報警方式即將上線），詳見[報警服務使用指南](#)。

- 請求狀態分布統計

當可用性或者有效請求率低於100%（或者有效請求數不等於總請求數時），可以通過查看請求狀態分布統計快速確定請求的錯誤類型，該統計監控指標的詳細介紹參見[OSS監控指標參考手冊](#)。

用户层级请求状态分布		
监控项	统计值	百分比
服务端错误请求数	1246960次	5.56%
网络错误请求数	1232826次	5.49%
授权错误请求数	1246098次	5.55%
资源不存在错误请求数	1252403次	5.58%
客户端超时错误请求数	2485623次	11.08%
客户端其他错误请求数	12472186次	55.64%
成功请求数	1246815次	5.56%
重定向请求数	1232157次	5.49%
总计	22415068次	100%

監視請求狀態詳情

請求狀態詳情在請求狀態分布統計的基礎上進一步細化請求監控狀態，可以更加深入、具體地監視某類請求。

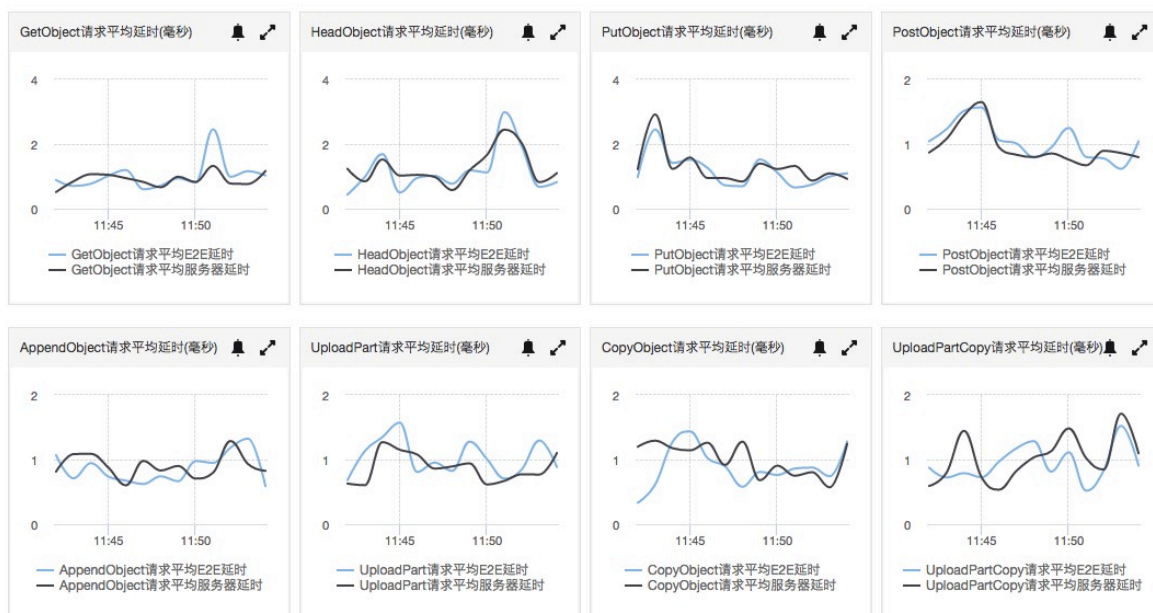


監視性能

監控服務提供了以下幾個監控項用來監控性能相關的指標。

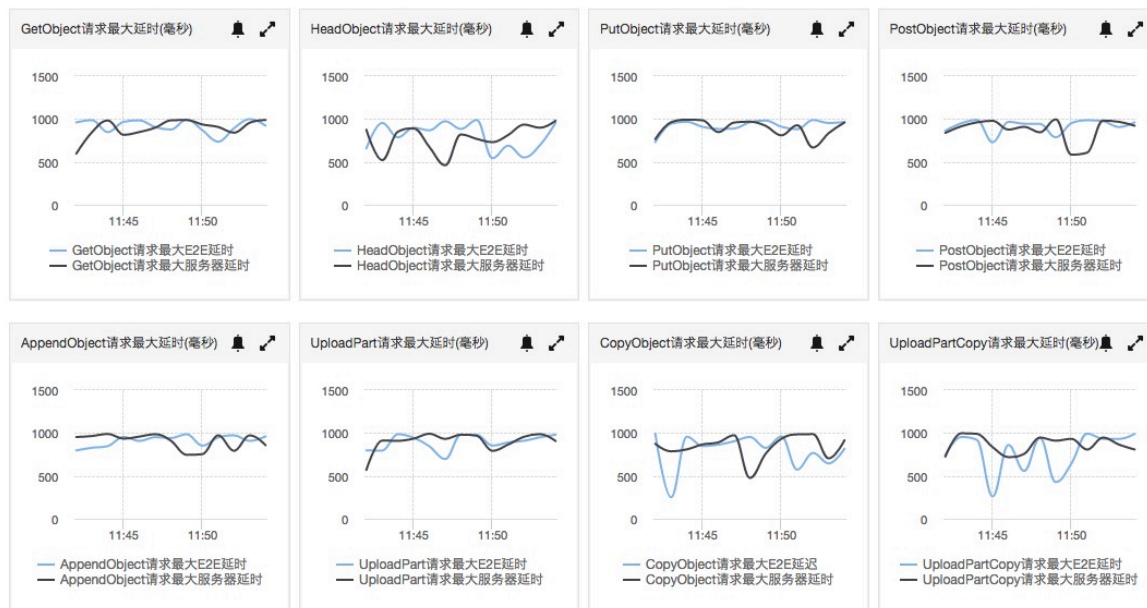
- 平均延時，包括E2E平均延時和伺服器平均延時

指标分组: 服务监控总览 请求状态详情 计量参考 **平均延时** 最大延时 成功请求操作分类



- 最大延時，包括E2E最大延時和伺服器最大延時

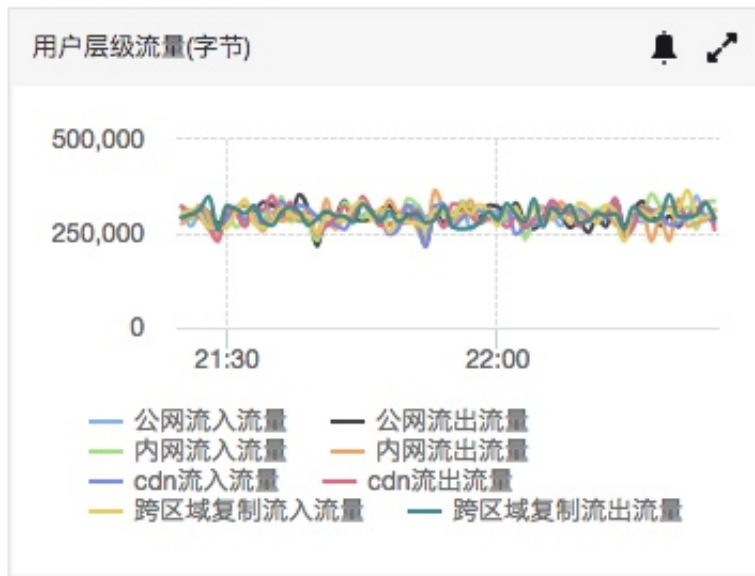
指标分组: 服务监控总览 请求状态详情 计量参考 平均延时 最大延时 成功请求操作分类



- 成功请求操作分类



· 流量



以上各個監控項（除流量）都分別從API操作類型進行分類監控：

- GetObject
- HeadObject
- PutObject
- PostObject
- AppendObject
- UploadPart
- UploadPartCopy

延時指標顯示API操作類型處理請求所需的平均和最大時間。其中E2E延時是端到端延遲指標，除了包括處理請求所需的時間外，還包括讀取請求和發送響應所需的時間以及請求在網路上傳輸的延時；而伺服器延時只是請求在伺服器端被處理的時間，不包括與客戶端通訊的網路延時。所以當出現E2E延時突然升高的情況下，如果伺服器延時並沒有很大的變化，那麼可以判定是網路的不穩定因素造成的性能問題，排除OSS系統內部故障。

成功請求操作分類除了以上提到的API之外，還提供以下兩個API操作類型請求數量的監控：

- DeleteObject
- DeleteObjects

流量指標從用戶或者具體的Bucket層級的總體情況進行監控，關注公網、內網、cdn回源以及跨域複製等場景中的網路資源佔用狀況。

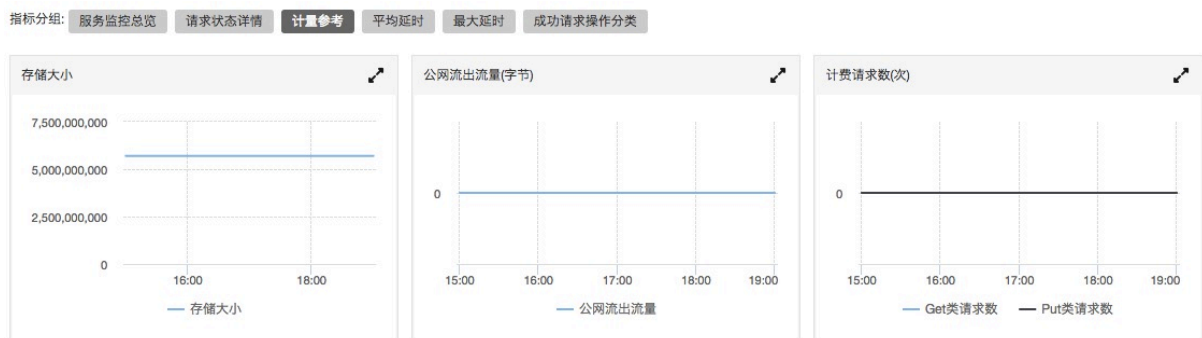
對於性能類指標項，需要重點關注其突發的異常變化，例如，平均延時突然出現尖峰，或者長時間處於高出正常請求延時的基準上方等等。用戶可以通過對性能指標設定對應的報警規則，對於

低於或者超過閾值時及時通知到相關人員。對於有業務周期性的業務，可以設定環比報警規則（環比一周、環比一天或者環比一個小時，環比報警方式即將上線）。

監視計量

撰寫本文檔時，OSS監控服務只支援監控儲存區大小、公網流出流量、Put類請求數和Get類請求數（不包括跨區域複製流出流量和cdn流出流量），而且不支援對計量數據的報警設定和OpenAPI的讀取。

OSS監控服務對Bucket層級的計量監控數據進行小時等級的粒度採集，可以在具體的Bucket監控視圖中查看其持續的監控趨勢圖。用戶可以根據監控視圖分析其業務的儲存服務資源使用趨勢並且進行成本的預估等。如下所示：



OSS監控服務還提供了用戶層級和Bucket層級兩個不同維度當月消費的資源數統計。即從當月1號開始到目前的賬戶或者某個Bucket所消耗的OSS資源總量，每小時更新，幫助用戶即時了解本月使用資源的情況並計算消費。

OSS的消費計算請參考[計量項和計費項](#)。



说明:

監控服務中提供的計量數據是盡最大可能推送的，可能會與實際消費賬單中產生差異，請以費用中心的數據為準。

跟蹤診斷

問題診斷

- 診斷性能

對於應用程式的性能判斷多帶有主觀因素，用戶需要根據具體的業務場景確定滿足業務需求的基準線，來判定性能問題。另外，從客戶端發起的請求，能引起性能問題的因素貫穿整個請求鏈路。比如OSS儲存服務load過大、客戶端tcp配置問題或網路基礎結構中存在流量瓶頸等。

因此診斷性能問題首先需要設定合理的基準線，然後通過監控服務提供的性能指標確定性能問題可能的根源位置，然後根據日誌查到詳細的資訊以便進一步診斷並且排除故障。

在下文[故障排除](#)中例舉了常見的性能問題和排查措施，可以參考。

- 診斷錯誤

用戶端應用程式會在請求發生錯誤時接收到服務端返回的相關錯誤資訊，監控服務也會記錄並顯示各種錯誤類型請求的計數和佔比。用戶也可以通過檢查伺服器端日誌、客戶端日誌和部落格來獲取相關單個請求的詳細資料。通常，響應中返回的HTTP狀態碼和OSS錯誤碼以及OSS錯誤資訊都指示請求失敗的原因。

相關錯誤響應資訊請參考 [OSS錯誤響應](#)。

- 使用Logging功能

OSS儲存服務為用戶的請求提供服務端日誌記錄功能，能幫助用戶記錄端到端的詳細請求日誌跟蹤。

有關如何開啟並使用logging功能，請參考[設定日誌](#)。

有關Log Service的命名規則以及記錄格式等詳細資料，請參見[設定訪問日誌記錄](#)。

- 使用部落格記錄工具

在許多情況下，通過Logging功能記錄的儲存日誌和用戶端應用程式的日誌數據已足以診斷問題，但在某些情況下，可能需要更詳細的資訊，這時需要使用部落格記錄工具。

捕獲客戶端和伺服器之間的流量，可以更詳細地獲取客戶端和伺服器之間交換的數據以及底層網路狀況的詳細資料，幫助問題的調查。例如，在某些情況下，用戶請求可能會報告一個錯誤，而伺服器端日誌中卻看不到任何該請求的訪問情況，這時就可以使用OSS的Log Service功能記錄的日誌來調查該問題的原因是否出在客戶端上，或者使用網路監視工具來調查網路問題。

最常用的部落格分析工具之一是[Wireshark](#)。該免費的協議分析器運行在數據包等級，能夠查看各種網路通訊協定的詳細數據包資訊，從而可以排查丟失的數據包和串連問題。

更詳細的WireShark操作請參見[WireShark用戶指南](#)。

E2E跟蹤診斷

請求從客戶端應用進程發起，通過網路環境，進入OSS服務端被處理，然後響應從服務端回到網路環境，被客戶端接收。整個過程，是一個端到端的跟蹤過程。關聯用戶端應用程式日誌、網路追蹤記錄檔和服務端日誌，有助於排查問題的詳細資料根源，發現潛在的問題。

在OSS儲存服務中，提供了RequestID作為關聯各處日誌資訊的標識符。另外，通過日誌的時間戳記，不僅可以迅速尋找和定位日誌範圍，還能夠了解在請求發生時間點範圍內，客戶端應用、網路或者服務系統發生的其他事件，有利於問題的分析和調查。

· RequestID

OSS服務始會為接收的每個請求分配唯一的伺服器請求ID，即RequestID。在不同的日誌中，RequestID位於不同的欄位中：

- 在OSS logging功能記錄的服務端日誌記錄中，RequestID出現在“Request ID”列中。
- 在網路跟蹤（如WireShark捕獲的資料流）中，RequestID作為x-oss-request-id標題值出現在響應消息中。
- 在客戶端應用中，需要客戶端code實現的時候將請求的RequestID自行列印到客戶端日誌中。在撰寫本文的時候，最新版本的JAVA SDK已經支援列印正常請求的RequestID資訊，可以通過各個API介面返回的Result結果的getRequestId這個方法獲取。OSS的各個版本SDK都支援列印出異常請求的RequestID，可以通過調用OSSException的getRequestId方法獲取。

· 時間戳記

使用時間戳來尋找相關日誌項，需要注意客戶端和伺服器之間可能存在的時間偏差。在客戶端上基於時間戳記搜索logging功能記錄的伺服器端日誌條目時，應加/減15分鐘。

故障排除

性能相關常見問題

- 平均E2E延時高，而平均服務端延時低

前面介紹了平均E2E延時與平均伺服器延時的區別。所以產生高E2E延時、低伺服器延時可能的原因有兩個：

- 用戶端應用程式響應慢。
- 網路原因導致。

用戶端應用程式響應速度慢的可能原因包括：

- 可用串連數或可用線程數有限。
 - 對於可用串連數問題，可以使用相關命令確定系統是否存在大量TIME_WAIT狀態的串連。如果是，可以通過調整核心參數解決。
 - 對於可用線程數有限，可以先查看客戶端CPU、記憶體、網路等資源是否已經存在瓶頸，如果沒有，適當調大並發線程數。
 - 如果還不能解決問題，那麼就需要通過優化客戶端代碼，比如，使用非同步訪問方式等。也可以使用性能分析功能分析用戶端應用程式熱點，然後具體優化。
- CPU、記憶體或網路頻寬等資源不足。
 - 對於這類問題，需要先使用相關系統的資源監控查看客戶端具體的資源瓶頸在哪裡，然後通過優化代碼使其對資源的使用更為合理，或者擴容客戶端資源（使用更多的核心或者記憶體）。

網路延遲的可能原因是：

通常，因網路導致的端到端高延遲是由暫時狀況導致的。可以使用Wireshark調查臨時和持久網路問題，例如數據包丟失問題。

- 平均E2E延時低，平均服務端延時低，但客戶端請求延時高

客戶端出現請求延時高的情況，最可能的原因是請求還未達到服務端就出現了延時。所以應該調查來自客戶端的請求為什麼未到達伺服器。

對於客戶端延遲發送請求，可能的客戶端的原因有兩個：

- 可用串連數或可用線程數有限，在上面章節已經描述過解決方案。
- 客戶端請求出現多次重試，如果遇到這種情況，需要根據重試資訊具體調查重試的原因再解決。可以通過下面方式確定客戶端是否出現重試：
 - 檢查客戶端日誌。詳細日誌記錄會指示重試已發生過。以OSS的JAVA SDK為例，可以搜索如下日誌提示，warn或者info的等級。如果存在該日誌，說明可能出現了重試。

```
[ Server ] Unable to execute HTTP request :  
或者
```

```
[ Client ] Unable to execute HTTP request :
```

- 如果客戶端的記錄層級為debug，以OSS的JAVA SDK為例，可以搜索如下日誌，如果存在，那麼肯定出現過重試。

```
Retrying on
```

如果客戶端沒有問題，則應調查潛在的網路問題，例如數據包丟失。可以使用工具（如Wireshark）調查網路問題。

- 平均服務端延時高

對於下載或者上傳出現服務端高延時的情況，可能的原因有2個：

- 大量客戶端頻繁訪問同一個小Object。

這種情況，可以通過查看logging功能記錄的服務端日誌資訊來確定是否在一段時間內，某個或某組小Object被頻繁訪問。

對於下載場景，建議用戶為該Bucket開通CDN服務，利用CDN來提升性能，並且可以節約流量費用；對於上傳場景，用戶可以考慮在不影響業務需求的情況下，收回Object或Bucket的寫存取權限。

- 系統內部因素。

對於系統內部問題或者不能通過優化方式解決的問題，請提供客戶端日誌或者Logging功能記錄的日誌資訊中的RequestID，聯繫系統工作人員協助解決。

服務端錯誤問題

對於服務端錯誤的增加，可以分為兩個場景考慮：

- 暫時性的增加

對於這一類問題，用戶需要調整用戶端程式中的重試策略，採用合理的退讓機制，例如指數退避。這樣不僅可以有效避免因為優化或者升級等系統操作（如為了系統負載平衡進行分區遷移等）暫時導致的服務不可用問題，還可以避開業務峰值的壓力。

- 永久性的增加

如果服務端錯誤持續在一個較高的水平，那麼請提供客戶端日誌或者Logging功能記錄的RequestID，聯繫後台工作人員協助調查。

網路錯誤問題

網路錯誤是指服務端正在處理請求時，串連在非伺服器端斷開而來不及返回HTTP要求標頭的情況。此時系統會記錄該請求的HTTP狀態碼為499。以下幾種情況會導致伺服器記錄請求的狀態碼變為499：

- 伺服器在收到讀寫請求處理之前，會檢查串連是否可用，不可用則為499。
- 伺服器正在處理請求時，客戶端提前關閉了串連，此時請求被記錄為499。

總之，在請求過程中，客戶端主動關閉請求或者客戶端網路斷掉都會產生網路錯誤。對於客戶端主動關閉請求的情況，需要調查客戶端中的代碼，了解客戶端斷開與儲存服務串連的原因和時間。對於客戶端網路斷掉的情況，用戶可以使用工具（如Wireshark）調查網路連接問題。

客戶端錯誤問題

- 客戶端授權錯誤請求增加

當監控中的客戶端授權錯誤請求數增加，或者用戶端程式接收到大量的403請求錯誤，那麼最常見的可能原因有如下幾個：

- 用戶訪問的Bucket網域名稱不正確。
 - 如果用戶直接用第三層網域名或者次層網域訪問，那麼可能的原因就是用戶的Bucket並不屬於該網域名稱所指示的region內，比如，用戶建立的Bucket的地域為杭州，但是訪問的網域名稱卻為Bucket.oss-cn-shanghai.aliyuncs.com。這時需要確認Bucket的所屬區域，然後更正網域名稱資訊。
 - 如果用戶開啟了cdn加速服務，那麼可能的原因是cdn綁定的回源網域名稱錯了，請檢查cdn回源網域名稱是否為用戶Bucket的第三層網域名。
- 如果用戶使用javascript客戶端遇到403錯誤，可能的原因就是CORS（跨域資源共用）的設定問題，因為Web瀏覽器實施了“同源策略”的安全限制。用戶需要先檢查所屬Bucket的CORS設定是否正確，並進行相應的更正。設定CORS參考[跨域資源共用](#)。
- 存取控制問題，可以分為下面四種：
 - 用戶使用主AK訪問，那麼用戶需要檢查是否AK設定出錯，使用了無效AK。
 - 用戶使用RAM子帳號訪問，那麼用戶需要確定RAM子帳號是否使用了正確的子AK，或者對應子帳號的相關操作是否已經授權。
 - 用戶使用STS臨時Token訪問，那麼用戶需要確認一下這個臨時Token是否已經過期。如果過期，需要重新申請。
 - 如果Bucket或者Object設定了存取控制，這個時候需要查看用戶所訪問的Bucket或者Object是否支援相關的操作。
- 授權第三方下載，即用戶使用簽名URL進行OSS資源訪問，如果之前訪問正常而突然遇到403錯誤，最大可能的原因是URL已經過期。
- RAM子帳號使用OSS周邊工具的情況也會出現403錯誤。這類周邊工具如ossftp、ossbrowser、OSS控制台客戶端等，在填寫相關的AK資訊登入時就拋出錯誤，此時如

果您的AK是正確填寫的，那麼您需要查看使用的AK是否為子帳號AK，該子帳號是否有GetService等操作的授權等。

- 客戶端資源不存在錯誤請求增加

客戶端收到404錯誤說明用戶試圖訪問不存在的資源資訊。當看到監控服務上資源不存在錯誤請求增加，那麼最大可能是以下問題導致的：

- 用戶的業務使用方式，比如，用戶需要先檢查Object是否存在來進行下一步動作，這時他會調用doesObjectExist（以JAVA SDK為例）方法，如果Object不存在，在客戶端則收到false值，但是這時在伺服器端實際上會產生一個404的請求資訊。所以，這種業務場景下，出現404是正常的業務行為。
- 客戶端或其他進程以前刪除了該對象。這種情況可以通過搜索logging功能記錄的服務端日誌資訊中的相關對象操作即可確認。
- 網路故障引起丟包重試。舉個例子，客戶端發起一個刪除操作刪除某個Object，此時請求達到服務端，執行刪除成功，但是響應在網路環境中丟包，然後客戶端發起重試，第二次的刪除操作可能會遇到404錯誤。這種由於網路問題引起的404錯誤可以通過客戶端日誌和服務端日誌確定：

- 查看客戶端應用日誌是否出現重試請求。

- 查看服務端日誌是否對該Object有兩次刪除操作，前一次的刪除操作HTTP Status為2xx。

- 有效請求率低且客戶端其他錯誤請求數高

有效請求率為請求返回的HTTP狀態碼為2xx/3xx的請求數佔總請求的比例。狀態碼為4XX和5XX範圍內的請求將計為失敗並降低該比例。客戶端其他錯誤請求是指除服務端錯誤請求（5xx）、網路錯誤請求（499）、客戶端授權錯誤請求（403）、客戶端資源不存在錯誤請求（404）和客戶端逾時錯誤請求（408或者OSS錯誤碼為RequestTimeout的400請求）這些錯誤請求之外的請求。

可以通過查看Logging功能記錄的服務端日誌確定這些錯誤的具體類型，可以在[OSS錯誤響應](#)上找到儲存服務返回的常見錯誤碼的列表，然後檢查客戶端代碼中尋找具體原因進行解決。

儲存容量異常增加

儲存容量異常增加，如果不是上傳類請求量增多，一般常見的原因應該是清理操作出現了問題，可以根據下面兩個方面進行調查：

- 客戶端應用使用特定的進程定期清理來釋放空間。針對這種請求的調查步驟是：
 1. 查看有效請求率指標是否下降，因為失敗的刪除請求會導致清理操作沒能按預期完成。
 2. 定位請求有效率降低的具體原因，查看具體是什麼錯誤類型的請求導致。然後還可以結合具體的客戶端日誌定位更詳細的錯誤資訊（例如，用於釋放空間的STS臨時Token已過期）。
- 客戶端通過設定Lifecycle來清理空間：針對這種請求，需要通過控制台或者API介面查看目前Bucket的Lifecycle是否為之前設定的預期值。如果不是，可以直接更正目前配置；進一步的調查可以通過Logging功能記錄的服務端日誌記錄查詢以前修改的具體資訊。如果Lifecycle是正常的，但是卻沒有生效，請聯繫OSS系統管理員協助調查。

其他儲存服務問題

如果前面的故障排除章節未包括您遇到的儲存服務問題，則可以採用以下方法來診斷和排查您的問題：

1. 查看OSS監控服務，了解與預期的基準行為相比是否存在任何更改。監控視圖可能能夠確定此問題是暫時的還是永久性的，並可確定此問題影響哪些儲存操作。
2. 使用監控資訊來幫助您搜索Logging功能記錄的服務端日誌數據，獲取相關時間點發生的任何錯誤資訊。此資訊可能會幫助您排查和解決該問題。
3. 如果伺服器端日誌中的資訊不足以成功排查此問題，則可以使用客戶端日誌來調查用戶端應用程式，或者配合使用網路工具（Wireshark等）調查您的網路問題。

14 雲端資料處理

圖片服務(IMG)

請參見[圖片處理指南](#)。

ApsaraVideo for Media Processing

ApsaraVideo for Media Processing是為多媒體數據提供的轉碼計算服務。它以經濟、易用、彈性和高可擴充的音視頻轉換方法，幫助您將儲存於OSS的音視頻轉碼成適合在PC、TV以及移動終端上播放的格式。

ApsaraVideo for Media Processing基於阿里雲雲端運算服務構建，它改變了以往進行轉碼時需要購買、搭建、管理轉碼軟硬體的高昂投入以及配置優化、轉碼參數適配等複雜性問題。同時，藉助雲端運算服務的Auto Scaling特性，可以按需提供轉碼能力，從而最大限度的滿足業務轉碼需求、避免資源浪費。

ApsaraVideo for Media Processing功能包括Web管理主控台、服務API和軟體開發套件。您可以通過它們使用和管理ApsaraVideo for Media Processing服務，也可以將ApsaraVideo for Media Processing功能整合到您自己的應用和服務中。

ApsaraVideo for Media Processing功能包括：

- 轉碼
- 管道
- 截圖
- 媒體資訊
- 浮水印
- 預置模版
- 自訂模版
- 剪輯輸出
- 解析度按比例縮放
- M3U8輸出自訂切片時長
- 音視頻抽取
- 視頻畫面旋轉
- 視頻轉GIF

更多功能及詳細介紹請參見[ApsaraVideo for Media Processing文檔](#)。