ALIBABA CLOUD

Alibaba Cloud

対象存储
開発者ガイド

Document Version: 20201013

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.基本概念

OSS を使用する前に、次の基本概念を理解することを推奨します。

バケット

バケットは、OSS に保存されているオブジェクトのコンテナーです。 すべてのオブジェクトは、バケットに保存されます。 OSS のデータモデル構造は、階層型モデルではなくフラットモデルです。

- すべてのオブジェクト (ファイル) は、対応するバケットに直接関連付けられています。 したがって、OSS には、ファイルシステムのようなディレクトリとサブフォルダーの階層構造がありません。
- ユーザーは複数のバケットを所有できます。
- バケット名は OSS 全体で一意でなければなりません。また、バケットの作成後にバケット名を変更することはできません。
- バケットには、無制限の数のオブジェクトを保存できます。

バケットの命名規則は次のとおりです。

- バケット名は、小文字、数字、ハイフン (-) のみを使用できます。
- バケット名の先頭と末尾は、小文字または数字にする必要があります。
- バケット名は 3 バイト以上 63 バイト以下にする必要があります。

オブジェクト

オブジェクトはファイルとも呼ばれ、 OSS に保存される基本的なエンティティです。 オブジェクトは、メタデータ、データ、キーで構成されます。 キーは、バケット内の一意のオブジェクト名です。 メタデータによって、最終変更時刻やオブジェクトサイズなど、オブジェクトの属性が定義されます。 オブジェクトのカスタムメタデータを指定することもできます。

オブジェクトのライフサイクルは、アップロードで始まり、削除で終わります。 ライフサイクルの間、オブジェクトの内容を変更することはできません。 オブジェクトを変更する場合は、既存のオブジェクトと同じ名前の新しいオブジェクトをアップロードして置き換える必要があります。 したがって、ファイルシステムとは異なり、OSS ではユーザーがオブジェクトを直接変更することはできません。

OSS には、追加アップロード機能があります。これにより、オブジェクトの最後にデータを続けて追加できます。

オブジェクトの命名規則は次のとおりです。

- オブジェクト名には UTF-8 エンコードを使用する必要があります。
- オブジェクト名は 1 バイト以上 1023 バイト以下にする必要があります。
- オブジェクト名の先頭にバックスラッシュ (\) とフォワードスラッシュ (/) は使用できません。

> 説明　オブジェクト名は、大文字と小文字が区別されます。 特に明記しない限り、OSS ドキュメントに記載されているオブジェクトとファイルは、まとめてオブジェクトと呼ばれます。

リージョン

リージョンは、 OSS データセンターの物理的な場所を表します。 作成したバケットを保存する OSS のリージョンを選択できます。 レイテンシとコストを最小限に抑えることが可能なリージョン、あるいは特定の規制要件を満たすリージョンを選択します。 一般的に、ユーザーに近いリージョンほど、アクセス速度は速くなります。 詳細は、「OSS リージョンとエンドポイント」をご参照ください。

リージョンは、オブジェクトレベルではなくバケットレベルで設定します。 したがって、バケットに含まれるすべてのオブジェクトは同じリージョンに保存されます。 バケットの作成時にリージョンを指定します。バケット作成後にリージョンを変更することはできません。

エンドポイント

エンドポイントは、 OSS へのアクセスに使用されるドメイン名です。 OSS は HTTP RESTful API を介して外部サービスを提供します。 リージョンごとに異なるエンドポイントを使用します。 同じリージョンでも、イントラネットを経由するアクセスかインターネットを経由するアクセスかによって、使用するエンドポイントは異なります。 たとえば、杭州リージョンの場合、インターネットエンドポイントは oss-cn-hangzhou.aliyuncs.com、イントラネットエンドポイントは oss-cn-hangzhou-internal.aliyuncs.com です。 詳細は、「OSS リージョンとエンドポイント」をご参照ください。

AccessKey

AccessKey は、AccessKeyID と AccessKeySecret で構成されます。 この 2 つが 1 組で、アクセス ID が検証されます。 OSS では、 AccessKeyId と AccessKeySecret の対称暗号方式を使用して、リクエスト送信者の ID を検証します。 AccessKeyID は、ユーザーを識別するために使用されます。 AccessKeySecret は、ユーザーが署名を暗号化するため、およびその署名を OSS で検証するために使用されます。 AccessKeySecret の機密性を保つ必要があります。 OSS では、AccessKey は次の 3 つの方法で生成されます。

- バケット所有者が AccessKey を申請します。
- バケットの所有者が、 RAM を使用して第三者による AccessKey の申請を許可します。
- バケット所有者が、STS を使用して第三者による AccessKey の申請を許可します。

AccessKey の詳細は、「アクセス制御」をご参照ください。

強力な一貫性

OSS では、オブジェクト操作はアトミックです。つまり中間状態がなく、操作は成功か失敗のどちらかです。 破損したデータや部分的なデータを書き込むことはありません。

OSS でのオブジェクト操作は、非常に一貫性があります。 たとえば、ユーザーが アップロード (PUT) 成功のレスポンスを受信すると、そのオブジェクトはすぐに読み取り可能になり、オブジェクトのデータは既に冗長化して書き込まれています。 したがって、書き込み後の読み取りに対して、強力な一貫性が提供されます。 削除操作についても同じことが言えます。 オブジェクトを削除すると、そのオブジェクトは直ちに削除されます。

データ冗長メカニズム

OSS は、データ冗長ストレージメカニズムを使用して、同じエリア内の異なる施設の複数のデバイスに各オブジェクトの冗長データを保存し、ハードウェア障害が発生した場合にデータの信頼性と可用性を確保します。

- OSS でのオブジェクト操作は、非常に一貫性があります。 たとえば、ユーザーがアップロードまたはコピー成功のレスポンスを受信すると、そのオブジェクトはすぐに読み取り可能になり、冗長データは既に複数のデバイスに書き込まれています。
- 完全なデータ送信を保証するため、OSS はネットワークトラフィックパケットのチェックサムを計算することにより、クライアントとサーバー間でパケットが送信されるときにエラーが発生するかどうかをチェックします。
- OSS の冗長ストレージメカニズムは、2 つのストレージ施設が同時に損傷した場合でもデータの損失を回避できます。

- データが OSS に保存された後、OSS は冗長データが失われているかどうかチェックします。 冗長データが失われた場合、OSS は失われた冗長データを復旧し、データの信頼性と可用性を確保します。
- OSS は、検証を通じて定期的にデータの整合性をチェックし、ハードウェア障害などの要因で引き起こされたデータ損傷を発見します。 データが部分的に破損または失われた場合、OSS は冗長データを使用して破損したデータを再作成し、修復します。

OSS とファイルシステムの比較

比較項目	OSS	ファイルシステム
データモデル	OSS は、キーと値のペア形式を使用した分散オブジェクトストレージサービスです。	ファイルシステムは、ファイルが保存されたディレクトリの階層ツリー構造です。
データ取得	オブジェクトは、一意のオブジェクト名 (キー) に基づいて取得されます。 test1/test.jpg などの名前を使用できますが、これはオブジェクト test.jpg が、test1 という名前のディレクトリに保存されていることを示す訳ではありません。 OSS の場合、test1/test.jpg と a.jpg との間に本質的な違いはありません。名前の異なるオブジェクトにアクセスする場合でも、同じ量のリソースが消費されます。	ファイルはディレクトリ内の位置に基づいて取得されます。
利点	大規模な同時アクセスをサポートしています。つまり、過剰なリソースを使用せずに、大量の非構造化データ (画像、ビデオ、ドキュメントなど) の保存と取得が可能です。	データのコピーや置き換えが不要なため、ディレクトリの名前変更、移動、削除などのフォルダー操作は非常に簡単です。
注意点	保存されているオブジェクトを直接変更することはできません。 オブジェクトを変更する場合は、既存のオブジェクトと同じ名前の新しいオブジェクトをアップロードして置き換える必要があります。	システムのパフォーマンスは、デバイスの容量に依存します。 ファイルシステム内に作成されるファイルやディレクトリが多いほど、多くのリソースが消費され、ユーザープロセスが長くなります。

そのため、OSS をファイルシステムにマッピングすることは推奨しません。 OSS を使用する際、大量の非構造化データ (画像、ビデオ、ドキュメントなど) を保存するための大量データ処理能力などの利点を最大限に活用することを推奨します。

OSS の概念とファイルシステムの概念の対応表は次のとおりです。

OSS	ファイルシステム
オブジェクト	ファイル

OSS	ファイルシステム
バケット	ホームディレクトリ
リージョン	該当なし
エンドポイント	該当なし
AccessKey	該当なし
該当なし	マルチレベルディレクトリ
GetService	ホームディレクトリリストの取得
GetBucket	ファイルリストの取得
PutObject	ファイルへの書き込み
AppendObject	既存のファイルへのデータの追加
GetObject	ファイルの読み取り
DeleteObject	オブジェクトの削除
該当なし	ファイルの内容の変更
CopyObject (同じターゲットとソース)	ファイル属性の変更
CopyObject	ファイルのコピー
該当なし	ファイル名の変更

2.エンドポイント

2.1. リージョンとエンドポイント

リージョンは、OSS のデータセンターがあるリージョンを示します。 エンドポイントは、ユーザーがインターネットを経由して OSS にアクセスする際に使用するドメイン名を示します。 このトピックでは、リージョンとエンドポイントの対応関係について説明します。

クラシックネットワークのリージョンと OSS エンドポイント

次の表に、クラシックネットワークの各リージョンの OSS インターネットエンドポイントとイントラネットエンドポイントを示します。

リージョン名	OSS リージョン	インターネットエンドポイント	インターネットエンドポイントプロトコル	ECS アクセス用のイントラネットエンドポイント	イントラネットエンドポイントプロトコル
中国 (杭州)	oss-cn-hangzhou	oss-cn-hangzhou.aliyuncs.com	HTTP と HTTPS	oss-cn-hangzhou-internal.aliyuncs.com	HTTP と HTTPS
中国 (上海)	oss-cn-shanghai	oss-cn-shanghai.aliyuncs.com	HTTP と HTTPS	oss-cn-shanghai-internal.aliyuncs.com	HTTP と HTTPS
中国 (青島)	oss-cn-qingdao	oss-cn-qingdao.aliyuncs.com	HTTP と HTTPS	oss-cn-qingdao-internal.aliyuncs.com	HTTP と HTTPS
中国 (北京)	oss-cn-beijing	oss-cn-beijing.aliyuncs.com	HTTP と HTTPS	oss-cn-beijing-internal.aliyuncs.com	HTTP と HTTPS
中国 (張家口)	oss-cn-zhangjiakou	oss-cn-zhangjiakou.aliyuncs.com	HTTP と HTTPS	oss-cn-zhangjiakou-internal.aliyuncs.com	HTTP と HTTPS
中国 (フフホト)	oss-cn-huhehaote	oss-cn-huhehaote.aliyuncs.com	HTTP と HTTPS	oss-cn-huhehaote-internal.aliyuncs.com	HTTP と HTTPS
中国 (深セン)	oss-cn-shenzhen	oss-cn-shenzhen.aliyuncs.com	HTTP と HTTPS	oss-cn-shenzhen-internal.aliyuncs.com	HTTP と HTTPS

リージョン名	OSS リージョン	インターネットエンドポイント	インターネットエンドポイントプロトコル	ECS アクセス用のイントラネットエンドポイント	イントラネットエンドポイントプロトコル
中国 (成都)	oss-cn-chengdu	oss-cn-chengdu.aliyuncs.com	HTTP と HTTPS	oss-cn-chengdu-internal.aliyuncs.com	HTTP と HTTPS
中国 (香港)	oss-cn-hongkong	oss-cn-hongkong.aliyuncs.com	HTTP と HTTPS	oss-cn-hongkong-internal.aliyuncs.com	HTTP と HTTPS
米国 (シリコンバレー)	oss-us-west-1	oss-us-west-1.aliyuncs.com	HTTP と HTTPS	oss-us-west-1-internal.aliyuncs.com	HTTP と HTTPS
米国 (バージニア)	oss-us-east-1	oss-us-east-1.aliyuncs.com	HTTP と HTTPS	oss-us-east-1-internal.aliyuncs.com	HTTP と HTTPS
シンガポール	oss-ap-southheast-1	oss-ap-southeast-1.aliyuncs.com	HTTP と HTTPS	oss-ap-southeast-1-internal.aliyuncs.com	HTTP と HTTPS
オーストラリア (シドニー)	oss-ap-southheast-2	oss-ap-southeast-2.aliyuncs.com	HTTP と HTTPS	oss-ap-southeast-2-internal.aliyuncs.com	HTTP と HTTPS
マレーシア (クアラルンプール)	oss-ap-southheast-3	oss-ap-southeast-3.aliyuncs.com	HTTP と HTTPS	oss-ap-southeast-3-internal.aliyuncs.com	HTTP と HTTPS
インドネシア (ジャカルタ)	oss-ap-southheast-5	oss-ap-southeast-5.aliyuncs.com	HTTP と HTTPS	oss-ap-southeast-5-internal.aliyuncs.com	HTTP と HTTPS
日本 (東京)	oss-ap-northeast-1	oss-ap-northeast-1.aliyuncs.com	HTTP と HTTPS	oss-ap-northeast-1-internal.aliyuncs.com	HTTP と HTTPS
インド (ムンバイ)	oss-ap-south-1	oss-ap-south-1.aliyuncs.com	HTTP と HTTPS	oss-ap-south-1-internal.aliyuncs.com	HTTP と HTTPS

リージョン名	OSS リージョン	インターネットエンドポイント	インターネットエンドポイントプロトコル	ECS アクセス用のイントラネットエンドポイント	イントラネットエンドポイントプロトコル
ドイツ (フランクフルト)	oss-eu-central-1	oss-eu-central-1.aliyuncs.com	HTTP と HTTPS	oss-eu-central-1-internal.aliyuncs.com	HTTP と HTTPS
イギリス (ロンドン)	oss-eu-west-1	oss-eu-west-1.aliyuncs.com	HTTP と HTTPS	oss-eu-west-1-internal.aliyuncs.com	HTTP と HTTPS
UAE (ドバイ)	oss-me-east-1	oss-me-east-1.aliyuncs.com	HTTP と HTTPS	oss-me-east-1-internal.aliyuncs.com	HTTP と HTTPS

説明

- リンクを共有したり、カスタムドメイン名 (CNAME) をバインドする場合は、`bucket name` + `endpoint` 形式で、第 3 レベルドメイン名を使用することを推奨します。 たとえば、中国 (上海) にあるバケット oss-sample の第 3 レベルドメイン名は、oss-sample.oss-cn-shanghai.aliyuncs.com です。
- SDK を使用する場合は、初期化パラメーターとして、`http://` または `http://` + `endpoint` の形式を使用してください。 たとえば、中国 (上海) の場合、エンドポイントの初期化パラメーターとして、`http://oss-cn-shanghai.aliyuncs.com` または `https://oss-cn-shanghai.aliyuncs.com` を使用することを推奨します。 初期化パラメーターとして、第 3 レベルドメイン名 `http://bucket.oss-cn-shanghai.aliyuncs.com` は使用しません。
- デフォルトでは、インターネットアドレス oss.aliyuncs.com は 中国 (杭州) のインターネットエンドポイントを指し、イントラネットアドレス oss-internal.aliyuncs.com は、中国 (杭州) のイントラネットエンドポイントを指します。

VPC ネットワークのリージョンと OSS エンドポイント

VPC ネットワーク内の ECS インスタンスの場合、次のエンドポイントを使用して OSS にアクセスできます。

リージョン名	OSS リージョン	VPC ネットワークのエンドポイント	プロトコル
中国 (杭州)	oss-cn-hangzhou	oss-cn-hangzhou-internal.aliyuncs.com	HTTP と HTTPS
中国 (上海)	oss-cn-shanghai	oss-cn-shanghai-internal.aliyuncs.com	HTTP と HTTPS

リージョン名	OSS リージョン	VPC ネットワークのエンドポイント	プロトコル
中国 (青島)	oss-cn-qingdao	oss-cn-qingdao-internal.aliyuncs.com	HTTP と HTTPS
中国 (北京)	oss-cn-beijing	oss-cn-beijing-internal.aliyuncs.com	HTTP と HTTPS
中国 (張家口)	oss-cn-zhangjiakou	oss-cn-zhangjiakou-internal.aliyuncs.com	HTTP と HTTPS
中国 (フフホト)	oss-cn-huhehaote	oss-cn-huhehaote-internal.aliyuncs.com	HTTP と HTTPS
中国 (深セン)	oss-cn-shenzhen	oss-cn-shenzhen-internal.aliyuncs.com	HTTP と HTTPS
中国 (成都)	oss-cn-chengdu	oss-cn-chengdu-internal.aliyuncs.com	HTTP と HTTPS
中国 (香港)	oss-cn-hongkong	oss-cn-hongkong-internal.aliyuncs.com	HTTP と HTTPS
米国 (シリコンバレー)	oss-us-west-1	oss-us-west-1-internal.aliyuncs.com	HTTP と HTTPS
米国 (バージニア)	oss-us-east-1	oss-us-east-1-internal.aliyuncs.com	HTTP と HTTPS
シンガポール	oss-ap-southheast-1	oss-ap-southeast-1-internal.aliyuncs.com	HTTP および HTTPS
オーストラリア (シドニー)	oss-ap-southheast-2	oss-ap-southeast-2-internal.aliyuncs.com	HTTP と HTTPS
マレーシア (クアラルンプール)	oss-ap-southheast-3	oss-ap-southeast-3-internal.aliyuncs.com	HTTP と HTTPS
インドネシア (ジャカルタ)	oss-ap-southheast-5	oss-ap-southeast-5-internal.aliyuncs.com	HTTP と HTTPS
日本 (東京)	oss-ap-northeast-1	oss-ap-northeast-1-internal.aliyuncs.com	HTTP と HTTPS
インド (ムンバイ)	oss-ap-south-1	oss-ap-south-1-internal.aliyuncs.com	HTTP と HTTPS
ドイツ (フランクフルト)	oss-eu-central-1	oss-eu-central-1-internal.aliyuncs.com	HTTP と HTTPS
イギリス (ロンドン)	oss-eu-west-1	oss-eu-west-1-internal.aliyuncs.com	HTTP と HTTPS

リージョン名	OSS リージョン	VPC ネットワークのエンドポイント	プロトコル
UAE (ドバイ)	oss-me-east-1	oss-me-east-1-internal.aliyuncs.com	HTTP と HTTPS

エンドポイントの使用

- OSS エンドポイントの構成ルール、およびインターネットとイントラネットを経由して OSS にアクセスする方法については、「エンドポイント」をご参照ください。
- ECS ユーザーとして OSS イントラネットエンドポイントを使用する場合は、「ECS から OSS への接続方法」をご参照ください。

2.2. エンドポイント

ドメイン名の構成ルール

GetService API に対するリクエストを除き、OSS に対するネットワーク要求では、ドメイン名はバケット名が指定された第 3 レベルのドメイン名です。

ドメイン名はバケット名とエンドポイントから構成され、BucketName.Endpoint の形式になります。 エンドポイントはアクセスドメイン名です。 OSS では、HTTP RESTful API を介して外部サービスを提供します。 ドメイン名はリージョンごとに異なります。 また、リージョンにはインターネットエンドポイントとイントラネットエンドポイントがあります。 たとえば、中国 (杭州) リージョンのインターネットエンドポイントは oss-cn-hangzhou.aliyuncs.com で、イントラネットエンドポイントは oss-cn-hangzhou-internal.aliyuncs.com です。 詳細は、「リージョンとエンドポイント (Regions and endpoints)」をご参照ください。

インターネットを経由した OSS へのアクセス

インターネットを経由して、いつでも OSS にアクセスできます。 インターネットでは、インバウンドトラフィック (書き込み) は無料で、アウトバウンドトラフィック (読み取り) は課金対象です。 アウトバウンドトラフィック料金の詳細は、「OSS の価格 (OSS Pricing)」をご参照ください。

次のいずれかの方法で、インターネットを経由して OSS にアクセスできます。

- 方法1：URL を使用して OSS リソースにアクセスします。 URL の構成は、次のとおりです。

```
<Schema>://<Bucket>.<Internet Endpoint>/<Object>, where:
 Schema is HTTP or HTTPS.
 Bucket is your OSS storage space.
 Endpoint is the access domain name for the region of a bucket. Enter the Internet endpoint here.
 Object is a file uploaded to the OSS.
```

たとえば、中国 (杭州) リージョンで、"myfile/aaa.txt" という名前のオブジェクトがバケット "abc" に格納されているとします。 オブジェクトのインターネットアクセスアドレスは次のとおりです。

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

次のように、HTML 形式でオブジェクトの URL を直接使用できます。

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

- 方法2：OSS SDK を使用して、インターネットアクセスドメイン名を構成します。

  異なるリージョンのバケットを操作するときは、異なるエンドポイントを設定する必要があります。

  たとえば、中国 (杭州) リージョンでバケットを設定する前に、クラスのインスタンス化の際にエンドポイントを設定する必要があります。

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

イントラネットを介した OSS へのアクセス

イントラネットとは、Alibaba Cloud プロダクト間の内部通信ネットワークを指します。 たとえば、ECS を介して OSS にアクセスします。 イントラネットでは、インバウンドトラフィックとアウトバウンドトラフィックは無料です。 ECS インスタンスと OSS バケットが同じリージョンにある場合は、イントラネットを使用して OSS にアクセスすることを推奨します。

次のいずれかの方法で、イントラネットを経由して OSS にアクセスできます。

- 方法1：URL を使用して OSS リソースにアクセスします。 URL の構成は、次のとおりです。

```
<Schema>://<Bucket>. <IntranetEndpoint>/<Object>, where:
 Schema is HTTP or HTTPS.
 Bucket is your OSS storage space.
 Endpoint is the access domain name for the region of a bucket. Enter the intranet endpoint here.
 Object is a file uploaded to the OSS.
```

  たとえば、中国 (杭州) リージョンで、"myfile/aaa.txt" という名前のオブジェクトがバケット "abc" に格納されているとします。 オブジェクトのイントラネットアクセスアドレスは次のとおりです。

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

- 方法2：ECS 上の OSS SDK を使用して、イントラネットアクセスドメイン名を構成します。

  たとえば、ECS 上の Java SDK の場合、 次のようにイントラネットエンドポイントを設定します。

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

  > 説明　イントラネットを経由して OSS にアクセスする場合は、ECS インスタンスと OSSバケットが同じリージョンにある必要があります。

  たとえば、中国 (北京) の ECS インスタンスを購入し、2 つの OSS バケットがあるとします。

  - 1 つのバケットは "beijingres" で、そのリージョンは中国 (北京) です。 ECS インスタンスでイントラネットアドレス `beijingres.oss-cn-beijing-internal.aliyuncs.com` を使用すると、"beijingres" リソースにアクセスできます。このとき生成されるトラフィックは無料です。

- もう 1 つのバケットは "qingdaores" で、リージョンは中国 (青島) です。 ECS インスタンスでイントラネットアドレス `qingdaores.oss-cn-qingdao-internal.aliyuncs.com` を使用しても "qingdaores" リソースにはアクセスできません。 "qingdaores" リソースにアクセスするには、インターネットアドレス `qingdaores.oss-cn-qingdao.aliyuncs.com` を使用します。このときに生成されるアウトバウンドトラフィックは、課金対象です。

3.ストレージクラス

3.1. 概要

OSS は、標準、低頻度アクセス (IA)、アーカイブの 3 つのストレージクラスを提供し、ホットデータからコールドデータまで、さまざまなデータストレージシナリオに対応します。

標準

OSS 標準ストレージは、高信頼性、高可用性、高性能のオブジェクトストレージサービスを提供し、頻繁なデータアクセスをサポートします。 OSS の高スループットかつ低レイテンシのレスポンス性能により、ホットスポットデータへのアクセスを効果的にサポートできます。 標準ストレージは、ソーシャルネットワーキングや共有で使用する画像の保存に最適です。また、オーディオやビデオアプリケーション、大規模な Web サイト、ビッグデータ分析で使用するデータの保存にも適しています。

標準ストレージクラスには、次の特徴があります。

- 99.999999999%のデータ信頼性を実現する設計です。
- 99.995%のサービス可用性を実現する設計です。
- 高スループットかつ低レイテンシのアクセスパフォーマンスを実現します。
- HTTPS ベースの送信に対応しています。
- 画像処理に対応しています。

IA

OSS IA ストレージは、保存期間が長く、アクセス頻度の低い (月に 1 回または 2 回) データの保存に適しています。 ストレージ単価が標準ストレージよりも低いため、各種モバイルアプリ、スマートデバイスデータ、およびエンタープライズデータの長期的なバックアップに適しています。 また、リアルタイムのデータアクセスも可能です。 IA ストレージクラスのオブジェクトには、最小保存期間があります。 保存期間が 30 日未満のオブジェクトを削除した場合、料金が発生します。 IA ストレージクラスのオブジェクトには、最小課金サイズがあります。 64 KB 未満のオブジェクトは、64 KB として課金されます。 また、データの取得には料金が発生します。

IA ストレージクラスには、次の特徴があります。

- 99.999999999%のデータ信頼性を実現する設計です。
- 99.995%のサービス可用性を実現する設計です。
- リアルタイムアクセスが可能です。
- HTTPS ベースの送信に対応しています。
- 画像処理に対応しています。
- 最小保存期間と最小課金サイズが定められています。

アーカイブ

OSS アーカイブストレージは、3 つのストレージクラスの中で最も低価格です。 医療用画像、科学資料、ビデオ映像など、長期間 (半年以上を推奨) のアーカイブデータの保存に適しています。 保存期間中、データにアクセスする頻度はかなり低いです。 データを凍結状態から読み取り可能な状態に復元するのに約 1 分かかります。 アーカイブストレージクラスのオブジェクトには、最小保存期間があります。 保存期間が 60 日未満のオブジェクトを削除した場合、料金が発生します。 アーカイブストレージクラスのオブジェクトには、最小課金サイズがあります。 64 KB 未満のオブジェクトは、64 KB として課金されます。 また、データの取得には料金が発生します。

OSS アーカイブストレージには、次の特徴があります。

- 99.999999999%のデータ信頼性を実現する設計です。
- 99.99%のサービス可用性を実現する設計です。
- 保存されたデータを凍結状態から読み取り可能な状態に復元するのに約 1 分かかります。
- HTTPS ベースの送信に対応しています。
- 画像処理に対応していますが、最初にデータを復元する必要があります。
- 最小保存期間と最小課金サイズが定められています。

ストレージクラスの比較

項目	標準	IA	アーカイブ
データ信頼性	99.999999999%	99.999999999%	99.999999999%
サービス可用性	99.995%	99.995%	99.99% (データ復元後)
オブジェクトの最小課金サイズ	オブジェクトの実際のサイズ	64 KB	64 KB
最小保存期間	N/A	30 日	60 日
データ取得料金	データ取得料金なし	取得するデータのサイズに応じて課金。 単位: GB	復元するデータのサイズに応じて課金。 単位: GB
データアクセス	リアルタイムアクセス (数ミリ秒のレイテンシあり)	リアルタイムアクセス (数ミリ秒のレイテンシあり)	データが凍結状態から読み取り可能な状態に復元されるまで 1 分
画像処理	対応	対応	対応 (データが凍結状態から読み取り可能な状態に復元された後)

> **説明**
>
> データ取得料金は、基盤となる分散ストレージシステムから読み取られるデータサイズに基づいて課金されます。 インターネットで送信されるデータは、アウトバウンドトラフィックの一部として課金されます。

対応する API

API	標準	IA	アーカイブ
バケットの作成、削除、照会			
PutBucket	対応	対応	対応
GetBucket	対応	対応	対応
DeleteBucket	対応	対応	対応

API	標準	IA	アーカイブ
バケットの ACL			
PutBucketAcl	対応	対応	対応
GetBucketAcl	対応	対応	対応
バケットのログ			
PutBucketLogging	対応	対応	対応
GetBucketLogging	対応	対応	対応
バケットの静的 Web サイトホスティング			
PutBucketWebsite	対応	対応	非対応
GetBucketWebsite	対応	対応	非対応
バケットのホットリンク保護			
PutBucketReferer	対応	対応	対応
GetBucketReferer	対応	対応	対応
バケットのライフサイクル			
PutBucketLifecycle	対応	対応	データ削除のみ対応
GetBucketLifecycle	対応	対応	対応
DeleteBucketLifecycle	対応	対応	対応
クロスリージョンレプリケーション			
PutBucketReplication	対応	対応	対応
CORS (Cross-Origin Resource Sharing)			
PutBucketcors	対応	対応	対応
GetBucketcors	対応	対応	対応
DeleteBucketcors	対応	対応	対応
オブジェクト操作			
PutObject	対応	対応	対応
PutObjectACL	対応	対応	対応

API	標準	IA	アーカイブ
GetObject	対応	対応	対応 (データが凍結状態から読み取り可能な状態に復元された後)
GetObjectACL	対応	対応	対応
GetObjectMeta	対応	対応	対応
HeadObject	対応	対応	対応
CopyObject	対応	対応	対応
OptionObject	対応	対応	対応
DeleteObject	対応	対応	対応
DeleteMultipleObjects	対応	対応	対応
PostObject	対応	対応	対応
PutSymlink	対応	対応	対応
GetSymlink	対応	対応	対応
RestoreObject	非対応	非対応	対応
マルチパート操作			
InitiateMultipartUpload	対応	対応	対応
UploadPart	対応	対応	対応
UploadPartCopy	対応	対応	対応
CompleteMultipartUpload	対応	対応	対応
AbortMultipartUpload	対応	対応	対応
ListMultipartUpload	対応	対応	対応
ListParts	対応	対応	対応
画像処理	対応	対応	対応

3.2. ストレージクラス間の変換

このトピックでは、オブジェクトのストレージクラスを標準、低頻度アクセス (IA)、およびアーカイブ間で変換する方法について説明します。

ライフサイクルオブジェクトの切り替え

OSS は、Standard、Infrequent Access、および Archiveの 3 つのストレージクラスをサポートします。

オブジェクト切り替えのメカニズムは、中国全リージョンの OSS ライフサイクル管理機能で使用可能になっています。 自動変換では、以下のストレージクラスがサポートされています。

- Standard から Infrequent Access
- Standard から Archive
- Infrequent Access から Archive

例

次のように、1 つのバケットに特定のプレフィックスを持つオブジェクトのライフサイクルポリシーを設定できます。

- 保存後 30 日間経つと、Infrequent Access クラスに変換されます。
- 保存後 180 日間経つと、Archive クラスに変換されます。
- 保存後 360 日間経つと、自動的に削除されます。

コンソールで前述のライフサイクルポリシーの設定を完了します。 詳しくは、「Set lifecycle」をご参照ください。

> **説明** 次の 3 つのパラメーターが設定されているとします。

指定日数後に、Transition to IA After、Transition to Archive After、およびDelete All Objects After Specified Daysを行う場合、各パラメーターに設定される日数は次の基準を満たす必要があります。

Infrequent Access に変換するまでの日数 < Archive に変換するまでの日数 < 削除されるまでの指定した日数

注記

オブジェクトタイプの変換後、変換後のストレージクラスの単価に基づいて、ストレージのコストが計算されます。

Infrequent Access および Archive のストレージタイプに関する注記

- 最小請求金額:

  128 KB より小さいオブジェクトは 128 KB として課金されます。

- 最小保存期間:

  保存したデータは少なくとも 30 日間は保存する必要があります。 30 日以内に保存されているファイルを削除した場合、料金が発生します。

- Archive タイプの復元時間:

  Archive タイプのオブジェクトがデータを読み取り可能な状態に復元するのに 1 分かかります。 ビジネスシナリオでリアルタイムの読み取りが必要な場合は、ファイルを Archive クラスではなく Infrequent Access ストレージクラスに変換することを推奨します。 Infrequent Access ストレージクラスにしないと、ファイルを Archive クラスに変換した後、データをリアルタイムに読み取ることはできません。

- データアクセスの料金

Infrequent Access と Archive の両方のクラスで、アウトバウンドトラフィックに対する個々の料金として、データアクセス料金をお支払いいただく必要があります。 オブジェクトごとの平均アクセス頻度が月に 1 回を超える場合は、データを Infrequent Access または Archive クラスに変換することを推奨しません。

他の方法によるストレージクラスの変換

Archive タイプから Standard クラスまたは Infrequent Access クラスへの変換、または Infrequent Access クラスから Standard クラスへの変換の場合、オブジェクトを読み取り、それを対応するストレージクラスのバケットに書き換えることができます。 オブジェクトの既定のストレージクラスは、バケットによって決定されます。

たとえば、Standard タイプのバケット内の Infrequent Access オブジェクトを Standard オブジェクトに変換する場合は、そのオブジェクトを読み書きすることができます。 バケットのタイプに基づくので、新しく書き込まれたオブジェクトは Standard ストレージクラスになります。

Archive クラスに変換されたオブジェクトは、復元操作を実行し、読み取り可能な状態に復元した後でのみ、読み取りが行えます。

詳しくは、 「アーカイブバケットの作成と使用」をご参照ください。

4.バケット

4.1. pay-by-requester モードの有効化

OSS でバケットの pay-by-requester モードを有効化すると、バケット所有者ではなく、リクエスターがリクエストとトラフィックの料金を支払います。 バケット所有者は、データのストレージ料金を支払います。 この機能を有効にすると、自分ですべてのリクエストとトラフィックの料金を支払わなくてもデータを共有できます。

> 説明 pay-by-requester モードは、中国 (深セン) でのみ利用可能です。

例

- 郵便番号簿、参照データ、地理空間情報、Web クロールデータなどの大規模データセットを共有します。 たとえば、ある研究所は公開データセットを提供して顧客とデータを共有し、リクエストとトラフィックの料金を顧客に支払ってもらいたいと考えています。 設定手順は次のとおりです。
  i. バケットの pay-by-requester モードを有効にします。 手順の詳細は、「バケットの pay-by-requester モードの有効化」をご参照ください。
  ii. バケットポリシーを設定して、顧客の Alibaba Cloud アカウントの RAM ユーザーがバケットにアクセスすることを許可します。 設定の詳細は、「バケットポリシーを使用した他のユーザーに対する OSS リソースへのアクセス許可」をご参照ください。
- 顧客やパートナーにデータを配信します。 たとえば、ある企業は生産データをパートナーに配信する必要があり、データのダウンロードによって生成されるリクエストとトラフィックの料金をパートナーに支払ってもらいたいと考えています。

  設定手順は次のとおりです。
  i. バケットの pay-by-requester モードを有効にします。
  ii. バケットの ACLを非公開に設定します。
  iii. バケットポリシーを設定して、パートナーの Alibaba Cloud アカウントの RAM ユーザーがバケットにアクセスすることを許可します。 設定の詳細は、「チュートリアル: バケットポリシーを追加して、他の Alibaba Cloud アカウントで RAM ユーザーに権限付与」をご参照ください。

> 注意 顧客またはパートナーの Alibaba Cloud アカウントの RAM ユーザーにバケットへのアクセスを許可する必要がありますが、企業側の RAM ユーザーの AccessKey ペアを提供しないでください。 顧客またはパートナーが、企業側の RAM ユーザーを使用してバケットにアクセスすると、リクエスターは企業側なので、リクエストとトラフィックの料金を支払う必要があります。

リクエスト方法

- 匿名アクセスは許可されていません。

  バケットの pay-by-requester モードを有効にしている場合、匿名ユーザーはバケットにアクセスできません。 リクエスターは OSS で識別されるように認証情報を提供し、リクエストとトラフィックの料金がバケット所有者ではなくリクエスターに課金されるようにしなければなりません。

  リクエスターが Alibaba Cloud アカウントの RAM ロールを使用してデータをリクエストした場合、この Alibaba Cloud アカウントにリクエストとトラフィックの料金が課金されます。
- リクエスターは、リクエストヘッダーに x-oss-request-payer フィールドを含める必要があります。

バケットの pay-by-requester モードを有効にした場合、リクエスターは、POST、GET、または HEAD リクエストのリクエストヘッダーに x-oss-request-payer:requester フィールドを含める必要があります。 このフィールドは、リクエスターがリクエストとデータのダウンロードに料金が発生することを理解していることを示します。 このフィールドを含めない場合、リクエストは認証に合格できません。

バケット所有者自身のバケットにアクセスするとき、リクエストヘッダーに x-oss-request-payer フィールドを含める必要はありません。 この場合、バケット所有者は、リクエストとトラフィックの料金を支払うリクエスターです。

料金分析

pay-by-requester モードでは、リクエスターがリクエストとトラフィックの料金を支払い、バケット所有者はデータのストレージ料金を支払います。 ただし、次の場合、リクエストは失敗し (HTTP ステータスコード 403 が返されます)、リクエスト料金はバケット所有者に課金されます。

- リクエスターは、POST、GET、または HEAD リクエストのリクエストヘッダーに x-oss-request-payer フィールドを含めていない。または、このフィールドを RESTful API を介したリクエストのパラメーターとして使用していない。
- リクエスターは認証されなかった。
- リクエスターは匿名。

参照

- コンソール: バケットの pay-by-requester モードの有効化
- ossutil: pay-by-requester モードでのバケットへのアクセス

4.2. カスタムドメイン名の割り当て

オブジェクトが OSS バケットにアップロードされると、そのオブジェクトの URL が自動的に生成されます。 この URL を使用して、バケット内のオブジェクトにアクセスできます。 カスタムドメイン名を使用して、アップロードされたオブジェクトにアクセスするには、オブジェクトが格納されているバケットにカスタムドメイン名を割り当て、そのバケットのインターネットドメイン名が示される CNAME レコードを追加する必要があります。

> 注意 中華人民共和国のインターネットの管理規則の要件に従って、カスタムドメイン名を割り当てる必要があるすべてのユーザーは、事前にドメイン名を工業情報化部に申請しなければなりません。 ドメイン名が登録されていない場合は、Alibaba Cloud が提供する ICP サービスを通じて登録できます。

基本概念

カスタムドメイン名をバケットに割り当てる際の主な概念は、以下のとおりです。

- ユーザードメイン名 (カスタムドメイン名または自己ホストドメイン名): ドメインネームプロバイダから購入したドメイン名を示します。
- OSS ドメイン名 (バケットドメイン名) : OSS によってバケットに割り当てられるドメイン名を示します。 このドメイン名を使って、バケット内のリソースにアクセスできます。 ユーザードメイン名を使用して OSS バケットにアクセスするには、ユーザードメイン名をバケットの OSS ドメイン名に割り当てる必要があります。つまり、Alibaba Cloud のドメインネームシステム (DNS) に CNAME レコードを追加します。
- Alibaba Cloud CDN domain name: Alibaba Cloud Content Distribution Network (CDN) でユーザードメイン名に割り当てられる CDN 高速化ドメイン名を示します。 CDN 高速化サービスを使用して

バケット内のリソースにアクセスするには、ユーザードメイン名を CDN 高速化ドメイン名に割り当てる必要があります。つまり、Alibaba Cloud DNS に CNAME レコードを追加する必要があります。

- Auto CDN cache update：バケット内のオブジェクトを変更しても、CDN ノード上のオブジェクトキャッシュの有効期限が切れていない場合、ユーザーは変更前のオブジェクトにのみアクセスできます。 この場合は、CDN ノードのオブジェクトキャッシュを手動で更新する必要があります。 操作を簡便化するため、OSS には自動 CDN キャッシュ更新機能が備わっています。 この機能を有効にすると、バケット内のオブジェクトが変更されると、自動的に CDN ノードに対して更新が実行されます。 詳細は、「自動 CDN キャッシュ更新の有効化 (Enable auto CDN cache update)」をご参照ください。

アプリケーションシナリオ

たとえば、ドメイン名が `img.abc.com` の Web サイトがあります。この Web サイトには、URL が `http://img.abc.com/logo.png` の画像が含まれます。 管理を容易にするために、コードを変更せず (つまり画像の URL を変更せずに)、画像に対するすべてのアクセスリクエストを OSS にリダイレクトすることを検討しています。 このような場合に、カスタムドメイン名をバケットに割り当てます。 カスタムドメイン名をバケットに割り当てるには、次の手順を実行します。

1. *abc-img* という名前のバケットを作成し、画像をバケットにアップロードします。
2. OSS コンソールからカスタムドメイン名 `img.abc.com` をバケット *abc-img* に割り当てます。
3. カスタムドメイン名 `img.abc.com` をバケット *abc-img* に割り当てると、OSS ではドメイン名がバケットにマッピングされます。
4. DNS サーバーに CNAME ルールを追加して、カスタムドメイン名 `img.abc.com` を `abc-img.oss-cn-hangzhou.aliyuncs.com` (バケット *abc-img* の OSS ドメイン名) にマッピングします。
5. `http://img.abc.com/logo.png` へのアクセスリクエストを受信すると、OSS では `img.abc.com` と *abc-img* のマッピング関係に基づいて、リクエストがバケット *abc-img* にリダイレクトされます。 つまり、`http://img.abc.com/logo.png` という URL で画像にアクセスしたユーザーは、`http://abc-img.oss-cn-hangzhou.aliyuncs.com/logo .png` という URL にリダイレクトされます。

次の表は、カスタムドメイン名の割り当て前後のアクセスプロセスを示します。

	カスタムドメイン名の割り当て前	カスタムドメイン名の割り当て後
アクセスプロセス	1. ユーザーが `http://img.abc.com/logo.png` へのアクセスリクエストを送信します。 2. DNS ではリクエストからサーバーの IP アドレスが取得されます。 3. ユーザーがサーバー上の画像 "logo.png" にアクセスします。	1. ユーザーが `http://img.abc.com/logo.png` へのアクセスリクエストを送信します。 2. DNS では、リクエストから URL `abc-img.oss-cn-hangzhou.aliyuncs.com` を解決します。 3. ユーザーは OSS バケット *abc-img* 内の画像 *logo.png* にアクセスします。

参照情報

- カスタムドメイン名の割り当て (Attach a custom domain name)
- CDN 高速化ドメイン名の割り当て (Attach a CDN acceleration domain name)

- 認証情報ホスティング (Certificate hosting)

4.3. 転送アクセラレーション

Alibaba Cloud OSS は、転送アクセラレーションを提供してデータ転送速度を向上させ、企業がより多くの Alibaba Cloud リージョンにビジネスをデプロイし、ユーザーエクスペリエンスを向上できるようにします。 また、インターネット経由での OSS オブジェクトのアップロードとダウンロードを高速化するために、転送アクセラレーションを備えています。 転送アクセラレーションは、スマートスケジューリング、プロトコルスタックチューニング、最適なルート選択、および転送アルゴリズムの最適化と、OSS サーバー側の設定を組み合わせて、エンドツーエンドのアクセラレーションソリューションを提供します。

さらに、世界中に分散しているデータセンターを使用して、転送アクセラレーションを実行しています。 データ転送要求が送信されると、最適なネットワークパスとプロトコルを使用して、バケットが存在するデータセンターにルーティングされます。

> 説明
>
> - 転送アクセラレーションを使用すると、トラフィックに基づいて追加料金が請求されます。 詳細については、「転送アクセラレーション料金」をご参照ください。
> - この機能は、UAE (ドバイ) を除くすべてのリージョンで利用できます。

利用イメージ

OSS 転送アクセラレーションは、アクセスを高速化し、ユーザーエクスペリエンスを向上させるために使用されます。

- リモートデータ転送の高速化

  たとえば、世界中にユーザーがいるフォーラムやオンラインの共同生産性ツールは、OSS にデータを保存できます。 ただし、アップロードとダウンロードの速度はリージョンによって異なるため、アクセスの一貫性は確保されません。 この場合、OSS 転送アクセラレーションを使用して、さまざまなリージョンのユーザーが、自分のリージョンに基づいて最適なネットワーク経由でデータを転送できるようにすることができます。 これにより、データ転送が高速化され、さまざまなリージョンのユーザーのアクセスエクスペリエンスが向上します。

- GB または TB のラージオブジェクトのアップロードとダウンロードの高速化

  インターネット経由でリモートリージョンのラージオブジェクトをアップロードまたはダウンロードするときに、転送アクセラレーションを有効にしてデータ転送を高速化できます。 転送アクセラレーションは、最適なインターネットルート選択とプロトコルスタックチューニングを使用して、リモートリージョンでのデータ転送タイムアウトを大幅に削減します。 マルチパートアップロードを使用すると、アップロードが失敗した位置から、再開可能なアップロードを実行できます。 マルチパートアップロードと転送アクセラレーションを連携して、リモートリージョンのラージオブジェクトをアップロードおよびダウンロードできます。

- 非静的および非ホットスポットデータのダウンロードを高速化

  たとえば、ユーザーエクスペリエンスは、写真管理アプリ、ゲーム、E コマースアプリ、企業ポータル Web サイト、金融アプリなど、高いデータダウンロード速度を必要とするアプリにおいて、製品競争力と顧客維持を推進する原動力となります。 ソーシャルネットワーキングアプリに関するコメントを取得するには、高いダウンロード速度も必要です。 OSS の転送アクセラレーションは、アップロードとダウンロードを高速化するために設計されたサービスです。 転送アクセラレーションを有効にして、帯域幅の使用率を最大化し、データ転送を高速化できます。

手順

バケットで転送アクセラレーションが有効になっている場合、2 つのパブリックエンドポイントがバケットで使用可能になります。

- アクセラレーションエンドポイント：oss-accelerate.aliyuncs.com。 このエンドポイントを使用して、アクセスとデータ転送を高速化できます。 たとえば、test という名前のバケットが、米国 (シリコンバレー) リージョンにあるとします。 *123.jpg* という名前のオブジェクトが、バケットのルートディレクトリに保存されます。 ブラウザ経由で OSS リソースにアクセスする場合、アクセラレーションエンドポイント http://test.oss-accelerate.aliyuncs.com/123.jpg を使用してアクセスを高速化できます。
- 通常のエンドポイント：形式は oss-region.aliyuncs.com です。 このエンドポイントは、高速化が不要な場合に使用できます。 通常のエンドポイントの詳細については、「リージョンとエンドポイント」をご参照ください。

> 注意 転送アクセラレーションを使用すると、追加料金が発生します。 コストを最小限に抑えるには、この機能を次のように使用することを推奨します。
>
> - オブジェクトのアップロードやダウンロードなど、アクセス速度の向上が必要な場合は、アクセラレーションエンドポイントを使用できます。
> - バケットの設定やオブジェクトの削除など、アクセス速度が重要でない場合は、通常のエンドポイントを使用できます。
> - OSS 内部エンドポイントを使用して、同じリージョンの ECS インスタンスから OSS バケットにアクセスすることを推奨します。 たとえば、バケットと ECS インスタンスが中国 (上海) リージョンにあるとします。 oss-cn-shanghai-internal.aliyuncs.com を使用して、ECS インスタンスから OSS にアクセスすることを推奨します。

ossutil は、実際のアクセラレーション効果をテストする例として使用されています。 結果は以下の図に示す通りです。

実行モード

転送アクセラレーションの設定 の指示に従って、OSS 転送アクセラレーションを有効にします。 転送アクセラレーションが有効になっている場合、次のいずれかの方法を使用して、OSS に保存されているデータにアクセスするために転送アクセラレーションを実行できます。

- ブラウザ

  ブラウザを使用して OSS に保存されているデータにアクセスする場合、オブジェクト URL のエンドポイントをアクセラレーションエンドポイントに置き換えます。 たとえば、https://test.oss-cn-shenzhen.aliyuncs.com/myphoto.jpg をhttps://test.oss-accelerate.aliyuncs.com/myphoto.jpg に置き換えます。 オブジェクト ACL がプライベートの場合、署名情報をオブジェクト URL に追加する必要があります。

  > 説明 転送アクセラレーションが有効になっており、カスタムドメイン名がバインドされているバケットへのアクセスを高速化するには、カスタムドメイン名をアクセラレーションエンドポイントに送信するように CNAME を設定してください。 詳細については、「高速化エンドポイントのバインド」をご参照ください。

- ossutil

ossutil を使用して OSS に保存されているデータにアクセスする場合、設定ファイルのエンドポイントを、アクセラレーションエンドポイントに置き換えます。 または、 `-e oss-accelerate.aliyuncs.com` を各コマンドに追加して、操作を実行します。 ossutil のエンドポイントを設定する方法の詳細については、「 ossutil」をご参照ください。

- ossbrowser

  ossbrowser を使用して OSS に保存されているデータにアクセスする場合、エンドポイントをカスタマイズに設定し、アクセラレーションエンドポイントを指定します。 ossbrowser のエンドポイントを設定する方法の詳細については、「ossbrowser」をご参照ください。

- SDK

  OSS にアクセスするために異なる言語を使用する SDK を使用する場合、エンドポイントをアクセラレーションエンドポイントに設定します。 次のコードは、OSS Java SDK を使用して簡単なアップロードを実行する方法の例を示しています。

```
// This example uses the endpoint of the China (Hangzhou) region. Specify the actual endpoint bas
ed on your requirements.
String endpoint = "http://oss-accelerate.aliyuncs.com";
// Security risks may arise if you use the AccessKey pair of an Alibaba Cloud account to log on to OS
S, because the account has permissions on all API operations. We recommend that you log on to the
OSS console as a RAM user to call API operations or perform routine operations and maintenance. T
o create a RAM user, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// Upload your local file. <yourLocalFile> consists of the local file path and a filename with extensio
n, for example, /users/local/myfile.txt.
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new File("<yourLocalFile>"));

// Shut down the OSSClient instance.
ossClient.shutdown();
```

  SDK のサンプルについての詳細は、「概要」をご参照ください。

注意

- 転送アクセラレーションを有効または無効にすると、設定が有効になるまで約 30 分かかります。
- 転送アクセラレーションを有効にしたら、アクセラレーションエンドポイントを使用してアクセスを高速化する必要があります。
- ツールのエンドポイントがアクセラレーションエンドポイントに設定されている場合、転送アクセラレーションが有効になっているバケットでのみ操作を実行できます。
- 転送アクセラレーションが有効になっている場合、他のエンドポイントは引き続き通常通り機能します。 エンドポイントはいつでも切り替えることができます。

- 転送アクセラレーションは、インターネット経由のアクセスの高速化に適用されます。
- データ転送のセキュリティを確保するために、転送アクセラレーションは転送プロトコルを HTTPS に選択的にアップグレードします。 したがって、アクセスに HTTPS を使用する場合、実際のアクセラレーションエンドポイントベースの URL は HTTPS を使用してください。 アクセスに HTTP を使用する場合、実際のアクセラレーションエンドポイントベースの URL は HTTPS に変更される場合があります。

料金

転送アクセラレーションを有効にし、バケットへのアクセスにアクセラレーションエンドポイントを使用する場合、OSS は追加料金を請求します。 アクセラレーションエンドポイントを使用して、転送アクセラレーションが有効になっているバケットから 1 GB のデータをダウンロードするとします。 OSS は、転送アクセラレーションと 1 GB のインターネット経由のアウトバウンドトラフィックに対して、1 GB のトラフィックに基づく料金を請求します。 料金の詳細については、「」「Object Storage Service の料金」をご参照ください。

アクセラレーションエンドポイントを使用するときに発生する転送アクセラレーション料金を表示するには、Alibaba Cloud Management コンソールの [課金管理] に移動してください。 」をご参照ください。次の表に、課金項目の詳細を示します。

課金項目	説明
AccM2MOut	中国本土内のリージョン間のダウンロードの高速化
AccM2MIn	中国本土内のリージョン間のアップロードの高速化
AccO2MOut	中国本土以外のリージョンから中国本土内にあるバケットへのダウンロードの高速化
AccO2MIn	中国本土以外のリージョンから中国本土内にあるバケットへのアップロードの高速化
AccM2OOut	中国本土内のリージョンから中国本土外にあるバケットへのダウンロードの高速化
AccM2OIn	中国本土内のリージョンから中国本土外にあるバケットへのアップロードの高速化
AccO2OOut	中国本土以外のリージョン間のダウンロードの高速化
AccO2OIn	中国本土以外のリージョン間のアップロードの高速化

よくある質問

- 転送アクセラレーションと Alibaba Cloud CDN の違いは何ですか。

機能	説明	利用イメージ

機能	説明	利用イメージ
転送アクセラレーション	転送アクセラレーションは、スマートスケジューリング、プロトコルスタックチューニング、最適なルート選択、および転送アルゴリズムの最適化と OSS サーバー側の設定を組み合わせて、エンドツーエンドのアクセラレーションソリューションを提供します。	○ オブジェクトのアップロードを高速化します。 ○ リモートオブジェクトのアップロードとダウンロードを高速化します。 ○ ラージオブジェクトのアップロードとダウンロードを高速化します。 ○ 動的なオブジェクトの更新と非ホットスポットオブジェクトのダウンロードを高速化します。
Alibaba Cloud CDN	Alibaba Cloud CDN は、OSS バケットをオリジンとして使用し、オリジンからエッジノードにコンテンツを配信します。データを読み取る際、ダウンロードを高速化するために最適なノードからキャッシュファイルが取得されます。	同じリージョンの多数のユーザーが同じ静的ファイルを同時にダウンロードする場合に、ダウンロードを高速化します。

- **カスタムドメイン名に基づいて転送アクセラレーションを実行するにはどうすればよいですか。**

  カスタムドメイン名を OSS にバインドした後、CNAME をアクセラレーションエンドポイントに送信することができます。詳細については、「高速化エンドポイントのバインド」をご参照ください。

4.4. CORS ルールの設定

クロスオリジンリソース共有 (CORS) は、HTML5 で提供される標準のクロスオリジンソリューションです。OSS では、CORS 標準を使用してクロスオリジンアクセスを実現します。OSS の PutBucketcors API を使用して、CORS ルールを設定できます。

> 説明
> - PutBucketcors API の詳細は、「CORS」をご参照ください。
> - CORS ルールの詳細は、『W3C CORS』をご参照ください。

JavaScript をサポートするブラウザーは、同一オリジンポリシーを使用してセキュリティを確保します。Web サイト A が Web ページ上で JavaScript コードを使用して、別オリジンの Web サイト B にアクセスすると、ブラウザーはリクエストを拒否します。この場合、CORS ルールを設定して、クロスオリジンリクエストを許可することができます。

操作方法

操作方法	説明
コンソール	直感的で使いやすい Web アプリケーション
Java SDK	
Python SDK	

操作方法	説明
PHP SDK	さまざまな言語の SDK デモ
Go SDK	
C SDK	
.NET SDK	

シナリオ

クロスオリジンアクセスは、実際のシナリオでよく使用されます。

たとえば、Web サイト www.a.com のバックエンドで OSS を使用します。 Web ページ上で JavaScript を使用してオブジェクトをアップロードできます。 ただし、アップロード Web ページ上のリクエストは、www.a.com にのみ送信できます。 ブラウザーは、他の Web サイトに送信されるすべてのリクエストを拒否します。 したがって、ユーザーがアップロードするデータは、www.a.com を介して転送される必要があります。 クロスオリジンアクセスが設定されている場合、データを直接 OSS にアップロードでき、www.a.com を介して転送する必要がありません。

CORS の実装

CORS は次のように実装されます。

1. CORS が有効になっている場合、オリジンを指定するための Origin フィールドが HTTP リクエストのヘッダーに 含まれています。 前の例では、Origin フィールドに www.a.com が設定されています。
2. サーバーはリクエストを受信した後、CORS ルールに基づいてオリジンからのリクエストを許可するかどうかを判断します。 リクエストが許可された場合、レスポンスヘッダーの Access-Control-Allow-Origin フィールドの値は www.a.com (このクロスオリジンを許可) です。 サーバーですべてのクロスオリジンリクエストが許可された場合、レスポンスヘッダーの Access-Control-Allow-Origin フィールドの値はアスタリスク (*) です。
3. ブラウザーは、Access-Control-Allow-Origin フィールドがレスポンスヘッダーに返されるかどうかをチェックして、クロスオリジンリクエストが許可されたかどうかを判断します。 このフィールドが返されない場合、ブラウザーはリクエストをブロックします。 リクエストが単純なリクエストではない場合、ブラウザーはまず OPTIONS リクエストを送信して、サーバーの CORS 設定を取得します。 サーバーが以降の CORS 操作を許可しない場合、ブラウザーは以降の単純でないリクエストをブロックします。

OSS では、CORS ルールを設定して、必要に応じてクロスオリジンリクエストを許可するか拒否するかを決定できます。 CORS ルールは、バケットレベルで設定できます。 詳細は、「PutBucketcors」 をご参照ください。

詳細分析

- ブラウザーには、CORS 関連のヘッダーフィールドが自動的に含まれています。 他の操作を実行する必要はありません。 CORS 操作はブラウザーにのみ適用されます。
- CORS リクエストが許可されるかどうかは、OSS 認証とは完全に独立しています。 OSS は CORS ルールを使用して、CORS 関連のヘッダーフィールドが含まれているかどうかを判断します。 ブラウザーは、そのリクエストをブロックするかどうかを判断します。
- クロスオリジンリクエストを送信する前に、ブラウザーのキャッシュ機能が無効になっていることを確認する必要があります。 たとえば、同じブラウザー内の 2 つの Web ページ (www.a.com がオリジンの Web ページと、www.b.com がオリジンの Web ページ) は、同じクロスオリジンリソースに対する

リクエストを同時に送信します。 サーバーが最初に www.a.com からリクエストを受信した場合、レスポンスヘッダーの Access-Control-Allow-Origin フィールドの値が www.a.com のリソースを返します。 サーバーが後で www.b.com からリクエストを受信すると、ブラウザーは、レスポンスヘッダーの Access-Control-Allow-Origin フィールドの値が www.a.com のキャッシュリソースを返します。 レスポンスヘッダーフィールドの値が www.b.com からのリクエストの CORS ルールと一致しないため、このリクエストは失敗します。

FAQ

一般的なエラーのトラブルシューティングについては、「OSS CORS エラーとトラブルシューティング」をご参照ください。

4.5. バケットのタグ付け

OSS では、バケットのタグ付けを設定して、バケットを分類および管理できます。 たとえば、特定のタグを持つバケットのみを一覧表示する設定ができます。

バケットのタグ付けでは、キーと値のペアを使用してバケットを識別します。 特定のタグを持つ複数のバケットを管理できます。

- バケットの所有者と承認されたユーザーのみがバケットのタグ付けを設定できます。 その他のユーザーがバケットのタグ付けを設定した場合、403 Forbidden 、エラーコード AccessDenied が返されます。
- バケットには最大 20 個のタグを設定できます。
- タグキーは必須です。 タグキーは長さが最大 64 バイトである必要があります。先頭文字列を `http://` 、 `https://` 、 `Aliyun` にすることはできません。
- タグ値はオプションです。 タグ値は最大 128 バイトである必要があります。
- タグのキーと値は UTF-8 でエンコードする必要があります。

実装モード

実装モード	説明
Java SDK	さまざまな言語の SDK デモ
Python SDK	
Go SDK	

手順

バケットにタグを追加後、同じタグを持つ複数のバケットを管理できます。 たとえば、同じタグを持つバケットを一覧表示し、RAM ユーザーによる同じタグを持つバケットの管理を許可することができます。

- 特定のタグを持つバケットの一覧表示

  特定のタグを持つバケットを一覧表示できます。 詳細については、以下の SDK デモをご参照ください。

  - Java SDK
  - Python SDK
  - Go SDK

- RAM ユーザーに特定のタグを持つバケットの管理を許可します

  多数のバケットがある場合、タグに基づいてバケットを分類し、RAM ポリシーを使用して、特定のユーザーに特定のタグを持つバケットの管理を許可できます。 たとえば、ユーザー A に keytest=valuetest のタグ設定を有するバケットを一覧表示する権限を付与できます。 RAM ポリシーは次のとおりです。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:ListBuckets"
      ],
      "Resource": [
        "acs:oss:*:1932487924256138:*"
      ],
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "oss:BucketTag/keytest": "valuetest"
        }
      }
    }
  ]
}
```

5.オブジェクト

5.1. ファイルのアップロード

5.1.1. フォームアップロード

フォームアップロードでは、OSS の PostObject API を使用して、最大サイズが 5 GB のオブジェクトをアップロードできます。

> 説明　PostObject API の詳細は、「PostObject」をご参照ください。

シナリオ

この方法は、HTML Web ページ上でオブジェクトをアップロードする場合に使用します。 典型的なシナリオは、Web アプリケーションです。 たとえば、次の表では、求人検索 Web サイトでのフォームアップロードプロセスと他のアップロードプロセスを比較しています。

	その他のアップロード方法	フォームアップロード
アクセスプロセス	1. Web ユーザーが履歴書のアップロードリクエストを送信します。 2. Web サーバーは履歴書アップロードページにレスポンスします。 3. 履歴書が Web サーバーにアップロードされます。 4. Web サーバーは履歴書を OSS にアップロードします。	1. Web ユーザーが履歴書のアップロードリクエストを送信します。 2. Web サーバーは履歴書アップロードページにレスポンスします。 3. 履歴書が OSS にアップロードされます。

- データは Web サーバーによって転送されず、 OSS に直接アップロードされるため、フォームアップロードプロセスの方が簡単です。
- 他のアップロードプロセスでは、オブジェクトは最初に Web サーバーにアップロードされます。 多数のオブジェクトをアップロードする場合、Web サーバーをスケールアウトする必要があります。 フォームアップロードプロセスでは、オブジェクトはクライアントから OSS に直接アップロードされます。 多数のオブジェクトをアップロードする場合、OSS にアップロードの負荷がかかりますが、サービス品質を確保します。

SDK デモ

詳細は、「Java SDK」をご参照ください。

アップロードの制限

- サイズ: このモードでは、オブジェクトの最大サイズは 5 GB です。
- 命名規則:
  - オブジェクト名は UTF-8 でエンコードされている要があります。
  - オブジェクト名は 1 バイト以上 1,023 バイト以下にする必要があります。
  - オブジェクト名を、スラッシュ (/) またはバックスラッシュ (\) で始めることはできません。

プロセス分析

1. POST ポリシーを作成します。

   POST リクエストのポリシーフォームフィールドは、有効なリクエストかどうかの検証に使用されます。 たとえば、ポリシーにはアップロードするオブジェクトのサイズと名前、クライアントのジャンプ先 URL、およびアップロードが成功した後にクライアントが受信する HTTP ステータスコードを指定できます。 詳細は、「Post ポリシー」をご参照ください。

   たとえば、次のポリシーでは、Web ユーザーによるデータのアップロード期限を 2115-01-27T10:56:19Z、アップロードするオブジェクトの最大サイズを 104,857,600 バイトに設定します。テスト用に長めの有効期間が設定されていますが、実用では推奨しません。

   ```
   This example uses the Python code. The policy is a JSON-formatted string.
    policy="{\"expiration\":\"2115-01-27T10:56:19Z\",\"conditions\":[[\"content-length-range\", 0, 104
   857600]]}"
   ```

2. Base64 でポリシー文字列をエンコードします。
3. OSS の AccessKey Secret を使用して、Base64 でエンコードされたポリシーに署名を追加します。
4. アップロード用の HTML ページを作成します。
5. HTML ページを開き、アップロードするオブジェクトを選択します。

次の例は、完全な Python サンプルコードを示します。

```
#coding = utf-8
import md5
import hashlib
import base64
import hmac
from optparse import OptionParser
def convert_base64(input):
   return base64.b64encode(input)
def get_sign_policy(key, policy):
   return base64.b64encode(hmac.new(key, policy, hashlib.sha1).digest())
def get_form(bucket, endpoint, access_key_id, access_key_secret, out):
   #1 Create a POST policy.
   policy="{\"expiration\":\"2115-01-27T10:56:19Z\",\"conditions\":[[\"content-length-range\", 0, 10485
76]]}"
   print("policy: %s" % policy)
   #2 Use Base64 to encode the policy string.
   base64policy = convert_base64(policy)
   print("base64_encode_policy: %s" % base64policy)
   #3 Use the AccessKey Secret of OSS to add a signature to the encoded policy.
   signature = get_sign_policy(access_key_secret, base64policy)
   #4 Create an HTML page for upload.
   form = '''
```

```
    <html>
      <meta http-equiv=content-type content="text/html; charset=UTF-8">
      <head><title>OSS form upload (through the PostObject API)</title></head>
      <body>
        <form  action="http://%s.%s" method="post" enctype="multipart/form-data">
          <input type="text" name="OSSAccessKeyId" value="%s">
          <input type="text" name="policy" value="%s">
          <input type="text" name="Signature" value="%s">
          <input type="text" name="key" value="upload/${filename}">
          <input type="text" name="success_action_redirect" value="http://oss.aliyun.com">
          <input type="text" name="success_action_status" value="201">
          <input name="file" type="file" id="file">
          <input name="submit" value="Upload" type="submit">
        </form>
      </body>
    </html>
    ''' % (bucket, endpoint, access_key_id, base64policy, signature)
    f = open(out, "wb")
    f.write(form)
    f.close()
    print("form is saved into %s" % out)
if __name__ == '__main__':
    parser = OptionParser()
    parser.add_option("", "--bucket", dest="bucket", help="specify ")
    parser.add_option("", "--endpoint", dest="endpoint", help="specify")
    parser.add_option("", "--id", dest="id", help="access_key_id")
    parser.add_option("", "--key", dest="key", help="access_key_secret")
    parser.add_option("", "--out", dest="out", help="out put form")
    (opts, args) = parser.parse_args()
    if opts.bucket and opts.endpoint and opts.id and opts.key and opts.out:
        get_form(opts.bucket, opts.endpoint, opts.id, opts.key, opts.out)
    else:
        print "python %s --bucket=your-bucket --endpoint=oss-cn-hangzhou.aliyuncs.com --id=your-acces
s-key-id --key=your-access-key-secret --out=out-put-form-name" % __file__
```

次のコード例を post_object.py として保存し、python post_object.py を実行します。

```
Usage:
python post_object.py --bucket=Your bucket name --endpoint=Bucket endpoint --id=Your AccessKey I
D --key=Your AccessKey Secret --out=Output object name
Example:
python post_object.py --bucket=oss-sample --endpoint=oss-cn-hangzhou.aliyuncs.com --id=tphpxp --k
ey=ZQNJzf4QJRkrH4 --out=post.html
```

説明

- 作成されたフォーム内の `success_action_redirect value=http://oss.aliyun.com` は、アップロードが成功した後に表示される Web ページを示します。 独自の Web ページに置き換えることができます。
- 作成されたフォーム内の `success_action_status value=201` は、アップロードが成功した後に HTTP ステータスコード 201 が返されることを示します。 別の HTTP ステータスコードに変更できます。
- 生成された HTML ページがpost.html の場合、post.html を開き、アップロードするオブジェクトを選択します。 この例では、アップロードが成功した後、OSS プロダクトページ (http://oss.aliyun.com) が表示されます。

アップロードのセキュリティと認証

許可されていないサードパーティのユーザーがバケットにデータをアップロードできないようにするため、バケットレベルおよびオブジェクトレベルのアクセス制御が提供されています。 詳細は、「アクセス制御」をご参照ください。

サードパーティのユーザーにオブジェクトのアップロードを許可するため、アカウント認証も提供されています。 詳細は、「認証済みのサードバーティによるアップロード」をご参照ください。

5.1.2. マルチパートアップロードと再開可能アップロード

OSS が提供するマルチパートアップロードと再開可能アップロードを使用することで、オブジェクトを複数のデータブロック (パート) に分割し、個別にアップロードできます。 すべてのオブジェクトパートをアップロードした後、API を呼び出して、1 つのオブジェクトに結合できます。

シナリオ

(PutObject API による) 簡易アップロードを使用して、大きなオブジェクトを OSS にアップロードする場合、ネットワークエラーが原因でアップロードが失敗する場合があります。 2 回目のアップロード試行では、オブジェクトを最初からアップロードする必要があります。 この場合、マルチパートアップロードを使用して、最後にアップロードしたパートからアップロードを再開できます。

他のアップロード方法と比較して、マルチパートアップロードは次のシナリオに適しています。

- ネットワーク接続性がよくない：あるパートのアップロードが失敗した場合、すべてのパートではなく、失敗したパートだけを再アップロードできます。
- 再開可能アップロードが必要：進行中のアップロードは、いつでも一時停止と再開が可能です。

- アップロードの高速化：OSS にアップロードするオブジェクトが大きい場合、複数のパートを同時にアップロードして処理を高速化できます。
- ストリーミングアップロード：サイズの不確定なオブジェクトをいつでもアップロードできます。 このシナリオは、ビデオ監視などの産業アプリケーションで一般的です。

マルチパートアップロード

操作方法	説明
ossutil	優れたパフォーマンスを発揮するコマンドラインツール
Java SDK	さまざまな言語の SDK デモ
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	

再開可能アップロード

マルチパートアップロード中にシステムがクラッシュした場合、アップロードを再開するには、ListMultipartUploads と ListParts API を使用して、オブジェクトのすべてのマルチパートアップロードタスクを取得し、各タスクでアップロードされたパートを一覧表示します。 これにより、最後にアップロードしたパートからアップロードタスクを再開することができます。 アップロードを一時停止してから再開する場合も、同じ原則が適用されます。

操作方法	説明
Java SDK	さまざまな言語の SDK デモ
Python SDK	
Go SDK	
C SDK	
.NET SDK	
Android SDK	
iOS SDK	

アップロードの制限

- サイズ：オブジェクトの最大サイズは、パートのサイズによって決まります。 マルチパートアップロードは、最大 10,000 個のパートをサポートします。 各パートは、100 KB 以上 (最後のパートを除く。最後のパートは小さいサイズでも可)、5 GB 未満でなければなりません。 したがって、オブジェクトのサイズは 48.8 TB 以下にします。

- 命名規則
  - オブジェクト名を UTF-8 でエンコードする必要があります。
  - オブジェクト名は 1 バイト以上 1,023 バイト以下にする必要があります。
  - オブジェクト名を、スラッシュ (/) またはバックスラッシュ (\) で始めることはできません。

マルチパートアップロードプロセス

マルチパートアップロードプロセスは次のとおりです。

1. アップロードするオブジェクトを複数のパートに分割します。
2. マルチパートアップロードタスクを初期化します (InitiateMultipartUpload APIを使用)。
3. パートを 1 つずつまたは同時にアップロードします (UploadPart API を使用)。
4. アップロードを完了します (CompleteMultipartUpload API を使用)。

マルチパートアップロードプロセス中、次の点に注意する必要があります。

- 最後のパートを除くすべてのパートは、100 KB 以上でなければなりません。 それ以外の場合、CompleteMultipartUpload API の呼び出しに失敗します。
- アップロードするオブジェクトをパートに分割した後、アップロード時に、指定された partNumber でソートされます。 ただし、順番にアップロードする必要はないため、パートを同時にアップロードできます。

  ネットワーク条件やデバイスの負荷により、多くのパートが同時にアップロードされる場合、アップロードは必ずしも高速化されません。 ネットワーク条件が良好な場合、パートサイズを増やすことを推奨します。そうでない場合はパートサイズを減らすことを推奨します。
- デフォルトでは、すべてのパートがアップロードされていても、CompleteMultipartUpload API が呼び出されていない場合、アップロードされたパートは自動的に削除されません。 AbortMultipartUpload API を呼び出してアップロードを終了し、ストレージ領域を占有しているパートを削除することができます。 アップロードされたパートを自動的に削除する方法の詳細は、「ライフサイクルルールの管理」をご参照ください。

アップロードのセキュリティと認証

許可されていない第三者ユーザーがバケットにデータをアップロードできないようにするため、バケットレベルおよびオブジェクトレベルのアクセス制御が提供されています。 詳細は、「アクセス制御」をご参照ください。

第三者ユーザーにオブジェクトのアップロードを許可するため、アカウント認証も提供されています。 詳細は、「許可された第三者によるアップロード」をご参照ください。

次のステップ

- OSS にオブジェクトをアップロードした後、アップロードコールバックを使用して、指定されたアプリケーションサーバーにコールバックリクエストを送信し、後続の操作を実行できます。
- 画像をアップロードした後、画像処理を使用できます。
- オーディオまたはビデオオブジェクトをアップロードした後、ApsaraVideo for Media Processing を使用できます。

API リファレンス

- マルチパートアップロード API:
  - MultipartUpload

- InitiateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- AbortMultipartUpload
- ListMultipartUploads
- ListParts

5.1.3. 追加アップロード

追加アップロードでは、OSS の AppendObject API を使用して、アップロード済みの追加可能オブジェクトにコンテンツを直接追加できます。

> 説明　AppendObject API の詳細は、「AppendObject」をご参照ください。

シナリオ

簡易アップロード、フォームアップロード、マルチパートアップロードと再開可能アップロードでは、アップロードが完了した後、固定コンテンツを持つ標準オブジェクトのみを作成できます。 このオブジェクトは、読み取り可能ですが、変更することはできません。 オブジェクトのコンテンツを変更するには、同じ名前のオブジェクトをアップロードして、既存のオブジェクトを上書きする必要があります。 これは、OSS と一般的なファイルシステムとの大きな違いです。

この特徴により、上記のアップロード方法は、ビデオ監視やビデオライブストリーミングなど、多くのシナリオでは不便です。ビデオデータは常にリアルタイムで生成されるためです。 上記のアップロード方法を使用する場合、ビデオストリームを小さいパートに分割してから、新しいオブジェクトとしてアップロードする必要があります。 この方法は、実用的ではありません。

- ソフトウェアアーキテクチャは複雑です。 オブジェクトの分割など、複雑な問題を考慮する必要があります。
- 作成されたオブジェクトのリストなど、メタデータを保存するためのスペースを確保する必要があります。 OSS でリクエストを受信した後、メタデータを何度も読み取り、オブジェクトを作成するかどうかを決定する必要があります。 これにより、サーバーに高い負荷がかかります。 また、クライアントは 1 つのリクエストを 2 回送信する必要があり、レイテンシが発生します。
- 小さいオブジェクトパートの場合、低レイテンシです。 ただし、オブジェクトの数が多すぎると、管理が難しくなります。 大きいオブジェクトパートの場合、高レイテンシです。

このようなシナリオで開発を簡素化し、コストを削減するため、追加アップロード (AppendObject API を使用) が提供されています。これにより、オブジェクトの末尾に直接コンテンツをを追加できます。 この方法でアップロードされたオブジェクトは追加可能オブジェクトですが、他の方法でアップロードされたオブジェクトは標準オブジェクトです。 追加されたデータは、即座に読み取り可能になります。

追加アップロードを使用することで、前述のシナリオのアーキテクチャがシンプルになります。 ビデオデータが生成されたら、追加アップロードを使用すると、即座に同じオブジェクトに追加されます。 クライアントは定期的にオブジェクトの長さを取得し、直前の値と比較する必要があります。 新しい読み取り可能データが見つかった場合、クライアントは読み取り操作を開始して、新しくアップロードされたデータを取得します。 この方法により、アーキテクチャが大幅に簡素化され、スケーラビリティが向上します。

ビデオのシナリオに加えて、ログデータの追加にも追加アップロードを使用できます。

操作方法

操作方法	説明
Java SDK	さまざまな言語の SDK デモ
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	
Android SDK	
iOS SDK	

アップロードの制限

- サイズ：このモードでは、オブジェクトの最大サイズは 5 GB です。
- 命名規則：
  - オブジェクト名は UTF-8 でエンコードされている要があります。
  - オブジェクト名は 1 バイト以上 1,023 バイト以下にする必要があります。
  - オブジェクト名を、スラッシュ (/) またはバックスラッシュ (\) で始めることはできません。
- オブジェクトタイプ：追加アップロードで作成されたオブジェクトにのみデータを追加できます。 したがって、簡易アップロード、フォームアップロード、マルチパートアップロード、再開可能アップロードで作成されたオブジェクトに新しいデータを追加することはできません。
- 以降の操作：追加アップロードで作成されたオブジェクトはコピーできません。 ただし、オブジェクトの Object Meta は変更できます。

アップロードのセキュリティと認証

許可されていない第三者ユーザーがバケットにデータをアップロードできないようにするため、バケットレベルおよびオブジェクトレベルのアクセス制御が提供されています。 詳細は、「アクセス制御」をご参照ください。

第三者ユーザーにオブジェクトのアップロードを許可するため、アカウント認証も提供されています。 詳細は、「許可された第三者によるアップロード」をご参照ください。

次のステップ

- 画像をアップロードした後、画像処理を使用できます。
- オーディオまたはビデオオブジェクトをアップロードした後、ApsaraVideo for Media Processing を使用できます。

> 説明　追加アップロードは、アップロードコールバックをサポートしていません。

5.1.4. 許可された第三者によるアップロード

このトピックでは、サーバーを介してオブジェクトを転送せず、直接 OSS にオブジェクトをアップロードすることを第三者ユーザーに許可する方法について説明します。

シナリオ

標準のクライアント/サーバーシステムアーキテクチャでは、サーバー側でクライアントからのリクエストを受信し、処理します。 OSS がバックエンドストレージサービスとして使用されている場合、クライアントはアップロードするオブジェクトをアプリケーションサーバーに送信し、アプリケーションサーバーはそのオブジェクトを OSS に転送します。 このプロセスでは、データを 2 回送信する必要があります。1 回はクライアントからサーバーへの送信、もう 1 回はサーバーから OSS への送信です。 アクセスリクエストが急増している場合、サーバーは十分な帯域幅リソースを提供して、複数のクライアントがオブジェクトを同時にアップロードできるようにする必要があります。 これは、アーキテクチャのスケーラビリティに対する課題です。

この問題を解決するため、OSS には、許可された第三者によるアップロード機能が備わっています。 この機能を使用すると、クライアントはオブジェクトをサーバーに送信せずに、OSS に直接アップロードできます。 これにより、アプリケーションサーバーのコストが削減され、大量のデータを処理する OSS の能力が最大化されます。 この場合、帯域幅と同時実行の制限を心配することなく、ビジネスに集中できます。

アップロード権限を付与する方法には、署名付き URL と次の一時的なアクセス資格情報の 2 つがあります。

署名付き URL

この方法では、OSSAccessKeyId と Signature フィールドを含むリクエスト URL を使用して、オブジェクトを直接アップロードできます。 セキュリティを確保するため、署名付き URL には有効期限があります。

- 操作方法

操作方法	説明
Java SDK	さまざまな言語の SDK デモ
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	

- 詳細は、「URL への署名の追加」をご参照ください。

一時的なアクセス資格情報

Alibaba Cloud は、Security Token Service (STS) を使用して一時的なアクセス資格情報を付与し、ユーザーを認証します。 STS は、クラウドコンピューティングユーザーに一時的なアクセストークンを提供する Web サービスです。 STS を介して、サードパーティー製アプリケーションまたは RAM ユーザー (ユーザー ID を管理可能な) に、カスタム有効期間と権限を持つアクセス資格情報を付与することができます。

- 操作方法

操作方法	説明
Java SDK	さまざまな言語の SDK デモ
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	
Android SDK	
iOS SDK	

- 操作例の詳細は、「STS が提供する一時的なアクセストークンによる OSS へのアクセス」をご参照ください。

5.1.5. RTMP ベースのストリームの取り込み

OSS では、Real-Time Messaging Protocol (RTMP) を使用して、H.264 でエンコードされたビデオストリームと Advanced Audio Coding (AAC) でエンコードされたオーディオストリームを OSS に取り込むことができます。 OSS にアップロードされたオーディオやビデオデータは、オンデマンドで再生したり、遅延の影響を受けないシナリオでライブストリーミングに使用したりできます。

RTMP に準拠してオーディオやビデオデータを OSS にアップロードする場合、次の制限に注意する必要があります。

- RTMP を使用すると、ビデオまたはオーディオストリームのみを取り込み可能で、ストリームをプルすることはできません。
- H.264 形式のビデオストリームを取り込む必要があります。
- オーディオストリームはオプションで、AAC 形式でなければなりません。 OSS は、他の形式のオーディオストリームを破棄します。
- オーディオやビデオデータのダンプに使用できるのは、HTTP Live Streaming (HLS) のみです。
- LiveChannel は、一度に 1 つのクライアントから取り込まれたストリームを受信できます。

以降のセクションでは、OSS にオーディオとビデオのストリームを取り込む方法と、ビデオオンデマンド (VOD) 再生やライブストリーミング用にアップロードされたオーディオやビデオデータを再生する方法について説明します。

OSS へのオーディオストリームやビデオストリームの取り込み

- アップストリーミング URL の取得

  SDK を使用して PutLiveChannel API を呼び出し、LiveChannel を作成して、アップストリーミング URL を取得します。 バケット ACL が公開読み取り/書き込みに設定されている場合、取得したアップストリーミング URL を直接使用できます。 それ以外の場合、署名をアップストリーミング URL に追加します。

次のコードは、Python SDK を例として使用し、署名なしと署名付きのアップストリーミング URL を取得する方法を示します。

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

次の例は、取得したアップストリーミング URL を示します。

```
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? OSSAcc
essKeyId=LGarxxxxxxHjKWg6&playlistName=t.m3u8&Expires=1472201595&Signature=bjKraZTTyzz9%
2FpYoomDx4Wgh%2FlM%3D"
```

- ストリームの取り込みに FFmpeg を使用

  次のコマンドを実行すると、FFmpeg を使用してローカルビデオファイルを OSS にアップロードできます。

  ```
  ffmpeg -i 1.flv -c copy -f flv "rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/test-channel? O
  SSAccessKeyId=LGarxxxxxxHjKWg6&Expires=1472199095&Signature=%2FAvRo7FTss1InBKgwn7Gz%2
  FUlp9w%3D"
  ```

- ストリームの取り込みに OBS を使用

  [設定] をクリックします。 **[URL]** フィールドに、前の手順で取得したアップストリーミング URL を入力し、**[OK]** をクリックします。

  次の図に示すように、アップストリーミング URL がどのように分割されるのか注意する必要があります。

OSS にアップロードされたオーディオデータやビデオデータの再生

- ライブストリーミング

  ストリームの取り込み中、HLS を使用して、次のプラットフォームにアップロードされているオーディオデータやビデオデータをさまざまな方法で再生できます。

  - Android や iOS などのモバイルプラットフォームでは、LiveChannel のストリーミング URL をブラウザーに入力します。

- macOS プラットフォームでは、Safari でコンテンツを再生します。
- PC では、VLC メディアプレーヤーをインストールして、コンテンツを再生します。

アップロードされたオーディオデータやビデオデータをライブストリーミング用にスムーズに再生するには、FragDuration を小さな値 (2 秒など) に設定します。 group of picture (GOP) に固定値 (LiveChannel の FragDuration の値と同じ) を設定することもできます。 次の図は、OBS での GOP (キーフレーム間隔) の設定方法を示します。

- VOD 再生

  ストリームの取り込み中、M3U8 オブジェクトをプッシュまたは更新するには、常にライブストリームが使用されます。 VOD 再生シナリオでは、ストリームの取り込み後に PostVodPlaylist API を呼び出して VOD 再生用の M3U8 オブジェクトをアセンブルし、オブジェクト URL を使用して、アップロードされたオーディオデータやビデオデータを再生する必要があります。

  VOD 再生シナリオでは、より大きな GOP を設定して、TS オブジェクトの数とビットレートを減らすことができます。

5.2. ファイルのダウンロード

5.3. ファイルの管理

5.3.1. アーカイブバケットの作成と使用

OSS には 3 つのストレージクラスがあり、その中でアーカイブストレージクラスの料金が最も低いです。ただし、アクセスする場合は、アーカイブデータを読み取り可能な状態に復元する必要があります。 本ページでは、アーカイブストレージクラスのバケットの作成方法と使用方法について説明します。 3 つのストレージクラスの詳細については、「ストレージクラスの概要 (Introduction to storage classes)」をご参照ください。

アーカイブバケットの作成

アーカイブバケットの作成には、コンソール、API / SDK、コマンドラインツールのいずれかを使用します。

- コンソールによるアーカイブバケットの作成

  コンソールでアーカイブバケットを作成するには、 *「アーカイブストレージ」* を Storage Class で選択します。

- API / SDK によるアーカイブバケットの作成

  Java SDK による 例は、以下のとおりです。

```
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
CreateBucketRequest createBucketRequest=new CreateBucketRequest(bucketName);
// Set the bucket ACL to public-read. The default ACL policy is private-read-write. createBucketRequ
est.setCannedACL(CannedAccessControlList.PublicRead);
// Set the storage class to Archive. The default storage class is Standard.
createBucketRequest.setStorageClass(StorageClass.Archive);
ossClient.createBucket(createBucketRequest);
```

`createBucketRequest.setStorageClass(StorageClass.Archive);` は、作成したバケットのストレージクラスがアーカイブであることを示します。

- OSS コマンドラインツールによるアーカイブバケットの作成

  ossutil の例は、以下のとおりです。

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

  `[bucket name]` は、該当するバケットに置き換えます。 `-storage-class` を `Archive` に設定します。

アーカイブバケットの使用

- アーカイブバケットへのデータのアップロード

  アーカイブバケットは、PutObject と MultipartUpload をサポートしていますが、 AppendObject はサポートしていません。 PutObject と MultipartUpload によってアップロードされたオブジェクトは、アーカイブバケットに直接保存できます。

- アーカイブバケットからのデータのダウンロード

  アーカイブバケットに格納されているオブジェクトには、リアルタイムでアクセスできません。 最初に復元リクエストを開始してから、オブジェクトが使用可能になるまで 1 分間待つ必要があります。

  アーカイブオブジェクトの復元プロセスは次のとおりです。

  i. アーカイブオブジェクトは、最初は凍結状態にあります。
  ii. 復元リクエストが送信されると、オブジェクトは復元中の状態になります。
  iii. 1 分後、オブジェクトは復元済みの状態になり、読み取り可能になります。
  iv. 復元済みの状態は、デフォルトで 1 日続き、最大 7 日間まで延長できます。 この期間が終了すると、オブジェクトは凍結状態に戻ります。

アーカイブオブジェクトを復元する際、コンソール、API / SDK、コマンドラインツールの内、どれを使用するかを選択できます。

- コンソールによるアーカイブオブジェクトの復元

  コンソールでアーカイブオブジェクトを復元するには、オブジェクトの[プレビュー] ページに移動し、[リストア] をクリックします。 オブジェクトの復元には 1 分かかります。 このプロセスの間、オブジェクトは復元中の状態にあります。

- API / SDK によるアーカイブオブジェクトの復元

  Java SDK を例に挙げて説明します。 オブジェクトを復元するには、"restoreObject" メソッドを呼び出します。

```
ObjectMetadata objectMetadata = ossClient.getObjectMetadata(bucketName, key);
// check whether the object is archive class
StorageClass storageClass = objectMetadata.getObjectStorageClass();
if (storageClass == StorageClass.Archive) {
    // restore object
    ossClient.restoreObject(bucketName, key);
    // wait for restore completed
    do {
        Thread.sleep(1000);
        objectMetadata = ossClient.getObjectMetadata(bucketName, key);
    } while (!  objectMetadata.isRestoreCompleted());

// get restored object
 OSSObject ossObject = ossClient.getObject(bucketName, key);
ossObject.getObjectContent().close();
```

- OSS コマンドラインツールによるアーカイブオブジェクトの復元

  ossutil を例に挙げて説明します。

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

  `[Bucket name]` と `[Object name]` を該当するバケットとオブジェクトに置き換えます。

5.3.2. バックツーオリジン設定の管理

バックツーオリジンルールを設定すると、OSS では元のサイトからリクエストされたデータが複数の方法で検索して読み込まれ、頻繁にアクセスされる (ホット) データの移行要件や特定のリクエストリダイレクトの要件を満たします。

この設定により、各 OSS GET リクエストの URL を一致させることができ、元サイトの検索方法が指定されます。 最大 5 つのルールを設定できます。 有効なルールと一致するまで、リクエストが設定された順序でルールと比較されます。 指定する方法は、ミラーリングまたはリダイレクトです。

ミラーリング

ミラーリングプロセスは、次のとおりです。クライアントからオブジェクトのデータがリクエストされます。 OSS でオブジェクトが存在しないと判断されると、リクエストはソース URL に転送されます。 オブジェクト が OSS を介してソース URL から返され、クライアントに送信されます。 同時に、オブジェクトデータが OSS に書き込まれるため、これから先にリクエストがあった場合、処理できます。

シナリオ例

ミラーリングライトバック機能は、データをシームレスに OSS に移行します。 つまり、ユーザーが構築したサイトや別のクラウド製品で 既に稼働しているサービスは、 サービスを中断することなく OSS に移行できます。 詳細なシナリオは、次のとおりです。

- オリジンサイトでは、新規のホットデータが生成され、またレガシーコールドデータ も保存されています。

まず、ユーザーは移行ツール ossimport を使用して、コールドデータを OSS に移行できます。移行中、 ユーザーはミラーリングライトバックを設定し、オリジンサイトの URL を OSS に設定します。 ドメイン名を OSS に切り替えたときに 新しく生成されたデータが移行されない場合でも、 ユーザーは OSS を介してそのデータにアクセスできます。 ファイルは初めてアクセスされた後に OSS に保存されます。 新しいデータが生成されなくなったオリジンサイトの ドメイン名を切り替えると、そのサイトがスキャンされ、 移行されていない全データが OSS にインポートされます。 この場合、ユーザーはミラーリングの 書き戻しを無効にすることができます。

オリジンサイトに IP アドレスが設定されている場合は、ドメイン名が OSS に移行された後も、 データはオリジンサイトにミラーリングされます。

- ただし、オリジンサイトがドメイン名の場合、ドメイン名は OSS または CDN に解決されるため、 ミラーリングは作成されません。 この場合、ユーザーは別のドメイン名を申請してオリジンサイトを ミラーリングします。

  このドメイン名と稼働中のドメイン名は、どちらも 同じ IP アドレスに解決されます。 これにより、サービスドメイン名が移行されたとき にオリジンサイトのイメージングを続行できます。

利用規約

- GetObject（）によって 404 コードが返される場合、OSS ではミラーリング書き戻しが実行され、 オリジンサイトのオブジェクトがリクエストされます。
- オリジンサイトからリクエストされた URL は、 `MirrorURL +オブジェクト` で、OSS に書き戻されるファイル名はオブジェクトです。 たとえば、バケット名が `example-bucket` で、 ミラーリング書き出しが設定されているとします。 MirrorURL は `http://www.example-domain.com/` です。 ファイル `"object.jpg"` は、このバケット内に格納されていません。 ファイルをダウンロードするため、OSS では `http://www.example.com/object.jpg` に対して GET リクエストが実行されます。結果が OSS に保存されてから、ユーザーに返されます。 ファイルは、OSS 上で `"object.jpg"` として利用できます。 これは、同じ名前のオブジェクトを OSS に移行するのと同じです。 MirrorURL が `http://www.example-domain.com/dir1/` のようなパス情報を持っている場合、プロセスは前述の例と同じです。ただし、OSS に書き込まれたオブジェクトは、 `"object.jpg"` のままですが、OSS オリジン検索 URL は `http://www.exampledomain.com/dir1/image/example_object.jpg` です。 このプロセスは、オリジンサイトのディレクトリから OSS にオブジェクトを移行するときと同じです。
- OSS に送信されたヘッダーおよびクエリ文字列情報は、 オリジンサイトには送信されません。
- データがオリジンサイトからチャンク方式で返される場合、データは OSS からユーザーに チャンク方式で返されます。
- OSS はオリジンサイトから以下のヘッダ情報を返して保存します。

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

- "MIRROR" + space + url_decode (オリジン検索 URL) という値を加えた x-oss-tag レスポンスヘッダーが、ミラーリング書き戻しファイルに追加されます。 上記の 例では、 次のようになります。

```
x-oss-tag:MIRROR http%3a%2f%2fwww.example-domain.com%2fdir1%2fimage%2fexample_object.jp
g
```

ファイルが OSS に書き戻された後、 再度上書きされない限り、ミラーリングから取得されたことを示すために、 ダウンロードのたびにこのヘッダーが追加されます。

- ミラーリング書き戻しによってファイルがすでに OSS に書き込まれていると想定した場合、 オリジンサイト上の対応するファイルが変更されても、 OSS に格納されているファイルは更新されません。 OSS に既存のファイルは、ミラーリングの書き戻し 条件を満たしていません。
- ファイルがミラーリングのソースに存在しない場合、 HTTP ステータス 404 が返されます。このステータスは、OSS を介してユーザーに転送されます。 ミラーリングソースが 200 以外のステータスコード (ネットワーク関連の原因によるファイル 取得の失敗など) が返される場合、ステータス 424 がユーザーに返されます。 これは、 `MirrorFailed` を表すエラーコードです。

リダイレクト

URL リダイレクト機能により、ユーザー定義の条件と対応するホップ構成に基づいて、3xx ホップがユーザーに返されます。 ユーザーはこのホップ機能を使用してファイルをリダイレクトし、このアクションに基づいてさまざまなサービスを提供できます。 プロセスは次のとおりです。

アプリケーションシナリオ

- データソースを OSS へ移行

  ユーザーは非同期にデータを OSS に移行できます。 このように、移行されていないデータに対するリクエストでは、 URL 書き換えメソッドを使用して 302 リダイレクトリクエストがユーザーに返されます。 次に、302 リダイレクトリクエスト内の位置に基づいて、ユーザーのクライアントのデータソースからデータが返されます。

- ページリダイレクト機能の設定

  ユーザーが特定のヘッダープレフィックスを使用してオブジェクトを非表示にしたい場合は、カスタマイズしたページをサイト訪問者に表示できます。

- 404 または 500 エラーが発生したときにリダイレクトされるページの設定

  404 または 500 エラーが発生した場合、ユーザーはライブページにリダイレクトされます。 これにより、OSS エラーがユーザーによって検出されなくなります。

参照

- コンソール： オリジンサイト検索ルールの管理

5.3.3. SelectObject API の使用

SelectObject API は、ログの分析によく使用されます。 また、ビッグデータプロダクトと組み合わせて使用することもできます。 このトピックでは、Python SDK と Java SDK を使用して SelectObject API を呼び出す方法について説明します。

概要

Object Storage Service (OSS) は、Apsara システムに基づく、安全で信頼性の高いクラウドストレージサービスです。 インターネット上に大量のデータを低コストで保存できます。 また、RESTful API、容量と処理能力の自動スケーリングもサポートしています。 OSS は、多数のメディアオブジェクトを保存できるだけでなく、多数のデータオブジェクトを保存するデータウェアハウスとしても機能します。 Hadoop 3.0 は OSS をサポートしています。 Amazon Elastic MapReduce (EMR) で Spark、Hive、Presto などのサービスを実行している場合、または MaxCompute、HybridDB、新たにリリースされた Data Lake Analytics などの Alibaba Cloud サービスを使用している場合、OSS のデータを直接読み取り、処理できます。

ただし、OSS が提供する GetObject API では、ビッグデータプラットフォームのみが OSS の全データをローカルにダウンロードして分析やフィルタリングを行うことができます。 その結果、多くのクエリシナリオでは、大量の帯域幅リソースとクライアントリソースが浪費されます。

この問題を解決するために、OSS では SelectObject API が提供されています。 これにより、ビッグデータプラットフォームではなく OSS が、条件と予測を使用してデータを事前にフィルタリングし、有用なデータのみをビッグデータプラットフォームに返すことができます。 この方法では、クライアントは少ない帯域幅リソースを使用して、少量のデータを処理し、CPU とメモリリソースの使用を最大化できます。 これにより、OSS ベースのデータウェアハウジングとデータ分析が非常に魅力的なオプションになります。

SelectObject API は、Java SDK と Python SDK でサポートされています。今後、他の言語の SDK もサポートされる予定です。 SelectObject API は、UTF-8 でエンコードされた CSV オブジェクトと JSON オブジェクトをサポートします。 CSV オブジェクト (TSV オブジェクトなどの CSV と同じようなオブジェクトを含む) は、RFC 4180 に準拠します。 CSV オブジェクトでは、行と列の区切り文字と引用文字をカスタマイズできます。 SelectObject API は、標準および低頻度アクセス (IA) ストレージクラスのオブジェクトをサポートします。 アーカイブストレージクラスのオブジェクトは、使用する前に復元する必要があります。 SelectObject API は、OSS が完全に管理するサーバー側暗号化方式 (SSE-OSS)、または Key Management Service (KMS) が管理する Customer Master Key (CMK) によるサーバー側暗号化方式 (SSE-KMS) で暗号化されたオブジェクトをサポートします。

SelectObject API は、DOCUMENT および LINES 形式の JSON オブジェクトをサポートします。 JSON DOCUMENT オブジェクトには、オブジェクトが 1 つ含まれます。 JSON LINES オブジェクトは、区切り文字で区切られたオブジェクトの行で構成されます。 ただし、JSON オブジェクト自体は、有効なオブジェクトでない場合があります。 SelectObject API は、\n や \r\n などの一般的な区切り文字をサポートしています。区切り文字を指定する必要はありません。

- サポートされる SQL 構文
  - SQL 文：SELECT、FROM、WHERE
  - データ型：string、int64、double64、decimal128、timestamp、Boolean
  - 演算：論理条件 (AND、OR、NOT)、算術式 (+、-、×、/、%)、比較演算 (>、=、<、>=、<=、!=)、および文字列操作 (LIKE と||)
- マルチパートクエリ

  SelectObject API は、GetObject API でサポートされているバイトベースのマルチパートダウンロードに類似したマルチパートクエリをサポートしています。 データは行またはスプリット単位で分割されます。 行単位のデータ分割は、一般的に使用されますが、スパースデータが分割されると負荷が不均衡になる可能性があります。 複数の行を含む各スプリットのサイズはほぼ同じなので、スプリット単位のデータ分割は、より効率的です。
- データ型

  OSS では、CSV オブジェクトのデータはデフォルトで string 型です。 CAST 関数を使用して、データ型を変換できます。 たとえば、次の SQL 文は、1 列目と 2 列目のデータを integer 型に変換して比較します。

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

また、SelectObject API を使用すると、WHERE 句のデータ型を暗黙的に変換できます。 たとえば、次の SQL 文は、1 列目と 2 列目のデータを integer 型に変換します。

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

SQL 文で CAST 関数を使用しない場合、JSON オブジェクトのデータ型は、オブジェクトのデータ型によって決まります。 標準の JSON オブジェクトは、null、Boolean、int64、double、string などのデータ型をサポートします。

Python SDK

```
import os
import oss2

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

# Initialize OSS information such as the AccessKey ID, AccessKey Secret, and endpoint.
# Obtain the information through environment variables or replace variables such as <yourAccessKeyI
d> with actual values.
#
# Use China (Hangzhou) as an example to set the endpoint to one of the following:
# http://oss-cn-hangzhou.aliyuncs.com
# https://oss-cn-hangzhou.aliyuncs.com

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

# Create an OSS bucket. All object-related methods must be called through the bucket.
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'
# Upload a CSV object.
bucket.put_object(key, content)
# Set the parameters of the SelectObject API.
csv_meta_params = {'CsvHeaderInfo': 'None',
'RecordDelimiter': '\r\n'}
select_csv_params = {'CsvHeaderInfo': 'None',
'RecordDelimiter': '\r\n',
'LineRange': (500, 1000)}
```

```
'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_call_back、select_c
sv_params）
select_content = result.read()
print(select_content)

result = bucket.select_object_to_file(key, filename,
"select * from ossobject where _3 > 44", select_call_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'
# Upload a JSON DOCUMENT object.
bucket.put_object(key, content)
select_json_params = {'Json_Type': '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'
# Upload a JSON LINES object.
bucket.put_object(key, content)
select_json_params = {'Json_Type': '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 の SelectObject API

- select_object
  - 次の例は、select_object 操作のサンプルコードを示しています。

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

    上記のサンプルコードでは、指定されたキーを持つオブジェクトに対して SQL 文を実行し、クエリ結果を返します。

    - SQL 文を sql パラメーターの値としてそのまま使用可能です。Base64 でエンコードする必要はありません。
    - progress_callback パラメーターはオプションです。 クエリの進行状況を報告するためのコールバック関数を指定します。
    - select_params は、この API の重要なパラメーターです。 select_object 操作のパラメーターとアクションを指定します。
    - headers パラメーターを使用して、リクエストに含まれるヘッダー情報を指定できます。 ヘッダー情報の機能は、GetObject API の機能と同じです。 たとえば、リクエストヘッダーにバイトフィールドを設定して、SQL 文が CSV オブジェクトでクエリ可能な範囲を指定できます。
  - 次の表に、select_params パラメーターでサポートされるパラメーターを示します。

名前	説明
Json_Type	■ このパラメーターを空のままにすると、デフォルトでオブジェクトは CSV オブジェクトになります。 ■ このパラメーターに DOCUMENT が設定されている場合、オブジェクトは JSON オブジェクトです。 ■ このパラメーターに LINES が設定されている場合、オブジェクトは JSON LINES オブジェクトです。

名前	説明
CsvHeaderInfo	CSV オブジェクトのヘッダー情報。 有効な値：*None*、*Ignore*、*Use* ■ None：このオブジェクトにヘッダー情報が含まれないことを示します。 ■ Ignore：このオブジェクトにヘッダー情報が含まれていることを示します。ヘッダー情報は SQL 文では使用されません。 ■ Use：このオブジェクトにヘッダー情報が含まれ、ヘッダー情報の列名が SQL 文で使用されることを示します。
CommentCharacter	CSV オブジェクトのコメント文字。 パラメーター値は 1 文字のみです。 デフォルト値：None。コメント文字がないことを示します。
RecordDelimiter	CSV オブジェクトの行区切り文字。 パラメーター値は、1 文字または 2 文字のみです。 デフォルト値：*\n*
OutputRecordDelimiter	select_object 操作の出力結果の行区切り文字。 デフォルト値：*\n*
FieldDelimiter	CSV オブジェクトの列区切り文字。 パラメーター値は 1 文字のみです。 デフォルト値：カンマ (,)。
OutputFieldDelimiter	select_object 操作の出力結果の列区切り文字。 デフォルト値：カンマ (,)。
QuoteCharacter	CSV オブジェクトの列の引用文字。 パラメーター値は 1 文字のみです。 デフォルト値：二重引用符 (") です。 引用符で囲まれた行区切り文字と列区切り文字は、通常の文字として処理されます。
SplitRange	マルチパートクエリのスプリット範囲。 パラメーター値は (start, end) 形式の限定された範囲で、start# から end# までのスプリットがクエリされることを示します。
LineRange	マルチパートクエリの行範囲。 パラメーター値は (start, end) 形式の限定された範囲で、start# から end# までの行がクエリされることを示します。
CompressionType	圧縮タイプ。 デフォルト値：None。 有効な値：GZIP。

名前	説明
KeepAllColumns	このパラメーターが true に設定されている場合、CSV オブジェクトの SELECT 文で除外された列は、出力結果で空のままになります。 ただし、列の位置は保持されます。 デフォルト値：false。 例： CSV オブジェクトに firstname 列、lastname 列、age 列が含まれています。 SQL 文は `select firstname, age from ossobject` です。 KeepAllColumns パラメーターが true に設定されている場合、出力結果は firstname,,age で、除外された lastname 列の位置を示すカンマ (,) が追加されます。 KeepAllColumns パラメーターが false に設定されている場合、出力結果は firstname,age です。 このパラメーターを使用すると、コードを変更する必要はありませんが、GetObject API の処理に使用されるコードを直接使用して、SelectObject API を処理できます。
OutputRawData	■ このパラメーターが true に設定されている場合、select_object 操作は元のデータをそのまま返します。 データが長時間返されない場合、タイムアウトエラーが発生することがあります。 ■ このパラメーターが false に設定されている場合、出力データはフレームにカプセル化されます。 デフォルト値：false。
EnablePayloadCrc	フレームごとに巡回冗長検査 (CRC) 値を計算するかどうかを示します。 デフォルト値：false。
OutputHeader	出力結果の先頭行のヘッダー情報。 このパラメーターは、CSV オブジェクトにのみ適用されます。
SkipPartialDataRecord	■ このパラメーターが true に設定されている場合、CSV オブジェクトの列にデータがない場合、または JSON オブジェクトにキーが存在しない場合、現在のレコードはスキップされます。 ■ このパラメーターが false に設定されている場合、データのない列は出力結果で空のままになります。 例： 行に firstname 列、lastname 列、age 列が含まれています。 SQL 文は `select _1, _4 from ossobject` です。 SkipPartialDataRecord パラメーターが true に設定されている場合、この行はスキップされます。 SkipPartialDataRecord パラメーターが false に設定されている場合、firstname,\n という結果が返されます。
MaxSkippedRecordsAllowed	スキップされる行の最大数。 デフォルト値：0。行がスキップされた場合、エラーが返されることを示します。

名前	説明
ParseJsonNumberAsString	このパラメーターが true に設定されている場合、JSON オブジェクトの数値は文字列として解決されます。 このパラメーターが false に設定されている場合、JSON オブジェクトの数値は整数または浮動小数点数として解決されます。 デフォルト値：false。 JSON オブジェクトの高精度浮動小数点数は、浮動小数点数として解決された場合、精度が低下します。 精度を維持するには、このパラメーターを true に設定し、CAST 関数を使用して、解決されたデータを 10 進数型に変換します。

- select_object 操作で返される結果：SelectObjectResult オブジェクトが返されます。 read () 関数または __iter__ メソッドを使用して、すべての結果を取得できます。 出力結果に大量のデータが含まれる場合、read() 関数ですべての結果を取得するのは、最適な方法とはいえません。 この関数は、すべての結果が返されるまでシステムをブロックし、過度のメモリリソースを使用する場合があります。

  __iter__ メソッドを使用して (結果の各チャンクに対して) すべての結果を取得し、結果のチャンクごとに処理することを推奨します。 この方法は、使用するメモリリソースが少なく、OSS サーバーでチャンクが処理された直後にクライアントで処理できます。 クライアントは、すべての結果が返されるのを待つ必要はありません。

- select_object_to_file

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

  上記のサンプルコードを使用して、指定されたキーを持つオブジェクトに対して SQL 文を実行し、クエリ結果を別のオブジェクトに書き込みます。

  その他のパラメーターは、select_object 操作のパラメーターと同じです。

- create_select_object_meta
  - 次のサンプルコードは、select_meta_params パラメーターの構文を示します。

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

    上記のサンプルコードを使用して、指定されたキーを持つオブジェクトに Select Meta を作成したり、このオブジェクトから Select Meta を取得したりします。 Select Meta には、行の総数、列の総数 (CSV オブジェクトの場合)、およびオブジェクト内のスプリットの総数が含まれます。

    オブジェクトの Select Meta が作成済みの場合、OverwriteIfExists パラメーターの値が true に設定されていない限り、Select Meta は再作成されません。

    オブジェクトの Select Meta を作成するには、オブジェクトを完全にスキャンする必要があります。

- 次の表に、select_meta_params パラメーターでサポートされるパラメーターを示します。

名前	説明
Json_Type	このパラメーターを空のままにすると、デフォルトでオブジェクトは CSV オブジェクトになります。 このパラメーターが指定されている場合、パラメーター値は LINES (オブジェクトが JSON LINES オブジェクトであることを示す) でなければなりません。 この操作は、JSON DOCUMENT オブジェクトには適用されません。
RecordDelimiter	CSV オブジェクトの行区切り文字。
FieldDelimiter	CSV オブジェクトの列区切り文字。
QuoteCharacter	CSV オブジェクトの引用文字。 引用符で囲まれた行区切り文字と列区切り文字は、通常の文字として処理されます。
CompressionType	圧縮タイプ。 このパラメーターが指定されている場合、パラメーター値は None でなければなりません。
OverwriteIfExists	作成された Select Meta が元の Select Meta を上書きするかどうかを示します。 一般的なシナリオでは、このパラメーターを設定する必要はありません。

- create_select_object_meta 操作で返される結果：行とスプリットの属性を含む GetSelectObjectMetaResult オブジェクトが返されます。 CSV オブジェクトの場合、結果の select_resp オブジェクトには columns 属性 (CSV オブジェクトの列数) が含まれます。

Java SDK

```
package samples;

import com.aliyun.oss.ClientBuilderConfiguration;
import com.aliyun.oss.model.*;
import com.aliyun.odps.Odps;
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.DefaultAcsClient;
```

```
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. AssumeRoleRequest;
import com.aliyuncs.sts.model.v20150401. AssumeRoleResponse;

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

/**
 * Examples of the create_select_object_meta and select_object operations.
 *
 */
class MulipartSelector implements Callable<Integer> {

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

    public MulipartSelector(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)
                        .withInputSerialization(
                                new InputSerialization().withCsvInputFormat(
                                        new CSVFormat().withHeaderInfo(CSVFormat.Header.None).withRecordDelimiter
("\n")
```

```
                                .withFieldDelimiter("|")))
                .withSplitRange(start, end)
                .withOutputSerialization(new OutputSerialization().withCsvOutputFormat(new CSVForm
at()).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";
    // Obtain the current Security Token Service (STS) API version.
    public static final String STS_API_VERSION = "2015-04-01";
    public static final String serviceAccessKeyId = "<AccessKey ID that can play the assumed role>";
    public static final String serviceAccessKeySecret = "<AccessKey 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 roleA
rn, String roleSessionName, String policy, ProtocolType protocolType, long durationSeconds) throws Cl
ientException {
        try {
            // Create an AcsClient instance for sending API requests.
            if (this.client == null) {
                IClientProfile profile = DefaultProfile.getProfile(REGION_CN_HANGZHOU, accessKeyId, accessK
eySecret);
                this.client = new DefaultAcsClient(profile);
            }
            // Create an AssumeRoleRequest instance and set its 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 the request and obtain the response.
            final AssumeRoleResponse response = client.getAcsResponse(request);
            return response;
        } catch (ClientException e) {
            throw e;
        }
    }


    public CredentialsProvider GetCredentialProvider()
            throws IOException {
        // Request parameters for the AssumeRole API include RoleArn, RoleSessionName, Policy, and Dur
ationSeconds.
        // You need to obtain the value of the RoleArn parameter in the Resource Access Management (R
AM) console.
        // The RoleSessionName parameter indicates the name of the session for the temporary token. Y
ou can use this parameter to identify users in audit or identify users who you want to issue tokens to.
        // However, you need to pay attention to the length and rules of the RoleSessionName parameter
. It can contain only letters, numbers, hyphens (-), and underscores (_), and cannot include spaces.
        // For more information about the rules, see the format requirements in the API reference.
```

```
        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" +
                "}";
        // You must set the protocol type to HTTPS.
        ProtocolType protocolType = ProtocolType.HTTPS;
        try {
            final AssumeRoleResponse response = assumeRole(serviceAccessKeyId, serviceAccessKeySecr
et,
                    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 tp 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 {
            CredentialsProvider provider = GetCredentialProvider();
            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>"; // You can obtain the Alibaba Cloud Resource
Name (ARN) of a RAM role in the RAM console. The RAM role must have permissions to access OSS.
    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 credentials = provider.getCredentials();
        //OSS client = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, config);
        System.out.println("Id " + credentials.getAccessKeyId());
        System.out.println("Key " + credentials.getSecretAccessKey());
        System.out.println("Token " + credentials.getSecurityToken());
        OSS client = new OSSClientBuilder().build(endpoint, credentials.getAccessKeyId(), credentials.get
SecretAccessKey(), credentials.getSecurityToken(), config);
        int totalSplits = 1;
        try {
            SelectObjectMetadata selectObjectMetadata = client.createSelectObjectMetadata(
                    new CreateSelectObjectMetadataRequest(bucketName, key)
                            .withInputSerialization(
                                    new InputSerialization().withCsvInputFormat(
```

```
                        new InputSerialization().withCsvInputFormat(
                                new CSVFormat().withHeaderInfo(CSVFormat.Header.None).withRecordDelimit
er(recordDelimiter))));

            totalSplits = selectObjectMetadata.getCsvObjectMetadata().getSplits();
            System.out.println(selectObjectMetadata.getCsvObjectMetadata().getTotalLines());
            System.out.println(totalSplits);

        }
        catch (Exception e)
        {
        e.printStackTrace();
      }

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

        ExecutorService executor = Executors.newFixedThreadPool(threadCount);
        long startTime = System.currentTimeMillis();
        List<Future<Integer>> list = new ArrayList<Future<Integer>>();
        int n = threadCount < totalSplits ? threadCount: totalSplits;
        for(int i = 0; i < n; i++) {
            int start = i * totalSplits/n;
            int end = i == n-1 ? totalSplits - 1 : (i+1)* totalSplits /n - 1;
            System.out.println("start:" + start + " end:" + end);
            Callable<Integer> task = new MulipartSelector(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.currentTimeMillis();
        System.out.println("total lines:" + totalLines);
        System.out.printf("Total time %dms\n" , (endTime - startTime));

      }
}
```

SQL 文の例

- SQL 文の例 (CSV オブジェクトの場合)

シナリオ	SQL 文
先頭の 10 行を返します。	select * from ossobject limit 10
1 列目の整数が 3 列目の整数よりも大きい場合、1 列目と 3 列目の整数を返します。	select _1, _3 from ossobject where cast(_1 as int) > cast(_3 as int)
1 列目のデータが X で始まるレコードの数を返します (like の後ろに指定する漢字は、UTF-8 でエンコードされている必要があります)。	select count(*) from ossobject where _1 like 'X%'
2 列目のデータの時間が 2018-08-09 11:30:25 以降で、3 列目のデータが 200 より大きいレコードをすべて返します。	select * from ossobject where _2 > cast('2018-08-09 11:30:25' as timestamp) and _3 > 200
2 列目の浮動小数点数の平均値、合計、最大値、最小値を返します。	select AVG(cast(_2 as double)), SUM(cast(_2 as double)), MAX(cast(_2 as double)), MIN(cast(_2 as double))
1 列目と 3 列目のデータで連結された文字列が Tom で始まり、Anderson で終わるすべてのレコードを返します。	select * from ossobject where (_1 || _3) like 'Tom%Anderson'
1 列目のデータが 3 で割り切れるすべてのレコードを返します。	select * from ossobject where (_1 % 3) == 0
1 列目のデータの範囲が 1995 から 2012 のレコードをすべて返します。	select * from ossobject where _1 between 1995 and 2012
5 列目のデータが N、M、G、または L のレコードをすべて返します。	select * from ossobject where _5 in ('N', 'M', 'G', 'L')
2 列目と 3 列目のデータの積が、100 と 5 列目のデータの合計よりも大きいレコードをすべて返します。	select * from ossobject where _2 * _3 > _5 + 100

- SQL 文の例 (JSON オブジェクトの場合)

  次の JSON オブジェクトを例として使用します。

```
{
 "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
}, ... # Other similar nodes are omitted.
]}
```

次の表に、SQL 文の例を示します。

シナリオ	SQL 文
age の値が 27 のレコードをすべて返します。	select * from ossobject.contacts[*] s where s.age = 27
すべての home の電話番号を返します。	select s.number from ossobject.contacts[*].phoneNumbers[*] s where s.type = "home"

シナリオ	SQL 文
spouse の値が null のレコードをすべて返します。	select * from ossobject s where s.spouse is null
children の値が空のままになっているレコードをすべて返します。	select * from ossobject s where s.children[0] is null 説明 他の方法で空の配列を指定できないため、上記の文を使用します。

ベストプラクティス

- マルチパートクエリで大きなオブジェクトをクエリします。

  CSV オブジェクトの列に行区切り文字が含まれていない場合、バイト単位でオブジェクトをパートに分割できます。 この方法は、オブジェクトの Select Meta を作成する必要がないため、最も簡単です。 CSV オブジェクトの列に行区切り文字が含まれているか、JSON オブジェクトをクエリする場合、次の手順を実行します。

  i. create_select_object_meta 操作を呼び出して、オブジェクトのスプリットの総数を取得します。 オブジェクトの SelectObject API を呼び出す必要がある場合、クエリの前に非同期で API 呼び出して、スキャン時間を短縮します。

  ii. クライアントのリソースに基づいて適切な同時実行性 n を選択し、スプリットの合計数を同時実行性 n で除算し、1 つのクエリに含まれるスプリット数を取得します。

  iii. リクエスト本文で split-range=1-20 などのパラメーターを設定して、マルチパートクエリを実行します。

  iv. 必要に応じて結果をマージします。

- 標準のオブジェクトには SelectObject API を使用します。 SelectObject API を使用して、マルチパートオブジェクトや追加可能オブジェクトをクエリしないことを推奨します。 内部構造の違いにより、クエリのパフォーマンスが低下する場合があります。
- JSON オブジェクトをクエリする場合、FROM 文で JSON パス範囲を絞り込みます。

  次の JSON オブジェクトを例として使用します。

```
{ contacts:[
      {"firstName":"John", "lastName":"Smith", "phoneNumbers":[{"type":"home", "number":"212-555-1
234"}, {"type":"office", "number":"646-555-4567"}, {"type":"mobile", "number":"123 456-7890"}], "addres
s":{"streetAddress": "21 2nd Street", "city":"New York", "state":NY, "postalCode":"10021-3100"}
      }
]}
```

  postalCode が 10021 で始まるレコードの streetAddress データをクエリするには、次の SQL 文を実行します。

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

  または、次の SQL 文を実行します。

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

2 番目の SQL 文の方が JSON パスの精度が高いため、クエリのパフォーマンスが向上します。

- JSON オブジェクトの高精度浮動小数点数を処理します。

  JSON オブジェクトの高精度浮動小数点数を計算する必要がある場合、ParseJsonNumberAsString パラメーターを true に設定し、CAST 関数を使用して解決済みデータを 10 進数型に変換することを推奨します。 たとえば、属性 a の値は 123456789.123456789 です。 `elect s.a from ossobject s where cast(s.a as decimal) > 123456789.12345` を実行すると、属性 a の精度を維持できます。

5.3.4. オブジェクトのタグ付け

オブジェクトのタグ付け機能を使用して、OSS オブジェクトを分類できます。 同じタグのオブジェクトのライフサイクルルールをバッチで設定できます。

> 説明 オブジェクトのタグ付け機能はベータテスト段階です。 この機能のトライアル版を申し込むには、チケットを起票し、サポートセンターへお問い合わせください。

オブジェクトのタグ付け機能は、キーと値のペアを使用してオブジェクトにタグ付けします。 オブジェクトをアップロードするときにタグを追加したり、既存のオブジェクトにタグを追加することもできます。

- オブジェクトには、異なるキーを持つタグを 10 個まで追加できます。
- キーと値の最大長は、それぞれ 128 バイトと 256 バイトです。
- キーと値は、大文字と小文字が区別されます。
- タグには、文字、数字、スペース、および次の記号を含めることができます。+ - = . _ : /
- バケット内のオブジェクトのタグを読み書きできるのは、バケットの所有者と、許可されたユーザーのみです。 オブジェクトのタグに対する読み取り権限と書き込み権限は、オブジェクトの ACL によって制限されません。
- オブジェクトを別のリージョンにコピーすると、オブジェクトのタグもリージョンにコピーされます。

適用シナリオ

オブジェクトのタグ付け機能は、フォルダーによって制限されません。 バケット内で同じタグを持つオブジェクトをすべて選択し、ライフサイクルルールをバッチで設定できます。 たとえば、定期的に生成され、長期間保存する必要のないオブジェクトにタグを追加し、そのタグを持つオブジェクトを定期的に削除するようにライフサイクルルールを設定できます。

使用法

- オブジェクトのタグ付け機能で使用できる API は、次のとおりです。
  - PutObjectTagging: オブジェクトにタグを追加します。 ターゲットオブジェクトに既にタグが設定されている場合、元のタグは新しいタグで上書きされます。
  - GetObjectTagging: オブジェクトのタグを読み取ります。
  - DeleteObjectTagging: オブジェクトのタグを削除します。
  - PutObject: PutObject を使用してオブジェクトをアップロードするときに、 `x-oss-tagging` リクエストヘッダーを指定すると、オブジェクトにタグが追加されます。
  - InitiateMultipartUpload : マルチパートアップロードタスクを開始するときに、 `x-oss-tagging` リクエストヘッダーを指定すると、オブジェクトにタグが追加されます。

- CopyObject：オブジェクトをコピーするときに、`x-oss-tagging-directive` リクエストヘッダーを指定すると、元のオブジェクトのタブをコピーするかどうかを決定できます。また、`x-oss-tagging` リクエストヘッダーを指定すると、ターゲットオブジェクトのタグを指定できます。
- GetObject：ターゲットオブジェクトのタグの読み取り権限がある場合、GetObject リクエストに対するレスポンスに `x-oss-tagging-count` ヘッダーが含まれ、ターゲットオブジェクトに追加されたタグの数が示されます。
- HeadObject：ターゲットオブジェクトのタグの読み取り権限がある場合、HeadObject リクエストに対するレスポンスに `x-oss-tagging-count` ヘッダーが含まれ、ターゲットオブジェクトに追加されたタグの数が示されます。

- 必要な権限

  オブジェクトのタグ付け機能に関連する API に必要な権限は、次のとおりです。

  - GetObjectTagging：オブジェクトのタグを取得する権限が必要です。この権限を使用すると、オブジェクトに追加されたタグを表示できます。
  - PutObjectTagging：オブジェクトのタグを設定する権限が必要です。この権限を使用すると、オブジェクトのタグを設定できます。
  - DeleteObjectTagging：オブジェクトのタグを削除する権限が必要です。この権限を使用すると、オブジェクトのタグを削除できます。

オブジェクトのタグ付けとライフサイクルルール管理

ライフサイクルルールを設定するときに、指定したプレフィックスやタグを持つオブジェクトにルールが適用されるように、条件を指定することができます。プレフィックスとタグをルールの条件として同時に設定することもできます。

- ライフサイクルルールの条件として特定のタグを設定した場合、オブジェクトのタグのキーと値の両方が、指定されたタグと一致する場合にのみ、ルールが適用されます。
- ライフサイクルルールの条件としてプレフィックスと複数のタグを指定した場合、オブジェクトのプレフィックスとタグが、指定されたプレフィックスとすべてのタグと一致する場合にのみ、ルールが適用されます。

例：

```
<LifecycleConfiguration>
<Rule>
<ID>r1</ID>
<Prefix>rule1</Prefix>
<Tag><Key>xx</Key><Value>1</Value></Tag>
<Tag><Key>yy</Key><Value>2</Value></Tag>
<Status>Enabled</Status>
<Expiration>
<Days>30</Days>
</Expiration>
</Rule>
<Rule>
<ID>r2</ID>
<Prefix>rule2</Prefix>
<Tag><Key>xx</Key><Value>1</Value></Tag>
<Status>Enabled</Status>
<Transition>
<Days>60</Days>
<StorageClass>Archive</StorageClass>
</Transition>
</Rule>
</LifecycleConfiguration>
```

上記のライフサイクルルールを設定した場合

- プレフィックスが rule1、タグが "xx=1" と ” yy=2” のオブジェクトは、30 日後に削除されます。
- プレフィックスが rule2、タグが "xx=1" のオブジェクトのストレージクラスは、60 日後にアーカイブに変更されます。

> 説明　詳細は、「ライフサイクルルールの管理」をご参照ください。

5.4. オブジェクトライフサイクル

5.4.1. ライフサイクルルールの管理

PutBucketLifecycleOSS インターフェイスを使用して OSS のバケットやオブジェクトのライフサイクルルールを設定し、期限切れのオブジェクトとパートを自動削除したり、期限切れオブジェクトのストレージクラスを IA やアーカイブに変更してストレージコストを節約することができます。

> 説明　PutBucketLifecycle インターフェイスの詳細は、「PutBucketLifecycle」をご参照ください。

ライフサイクルルールには、次の情報が含まれています。

- 一致ポリシー：ライフサイクルルールが適用されるオブジェクトとパートを指定します。
  - プレフィックスによる一致：ライフサイクルルールは、指定されたプレフィックスを持つオブジェクトとパートに適用されます。 プレフィックスが異なるオブジェクトとパートに対して、複数のライフサイクルルールを作成できます。 各ルールに指定するプレフィックスは、異なる必要があります。
  - タグによる一致：ライフサイクルルールは、指定されたタグキーとタグ値を持つオブジェクトに適用されます。 指定されたタグを持つすべてのオブジェクトにルールが適用されるように、1 つのライフサイクルルールに複数のタグを指定できます。 パートは、タグによるライフサイクルルールに一致しません。

    > 説明　オブジェクトのタグ付け機能はベータテスト段階です。 この機能のトライアル版を申し込むには、チケットを起票し、サポートセンターへお問い合わせください。 オブジェクトのタグ付け機能の詳細は、「オブジェクトのタグ付け」をご参照ください。

  - プレフィックスとタグによる一致：ライフサイクルルールは、指定されたプレフィックスと 1 つ以上の指定されたタグを持つオブジェクトに適用されます。
  - バケット全体に一致：ライフサイクルルールは、バケット内のすべてのオブジェクトとパートに適用されます。 バケットに適用されるライフサイクルルールは 1 つのみ作成できます。
- オブジェクトの有効期限ポリシー：オブジェクトの有効期限と、期限切れオブジェクトに対して実行する操作を指定します。
  - 有効期限：有効期限と、期限切れオブジェクトに対して実行する操作を指定します。 指定された日付より前に変更されたオブジェクトは期限切れになり、指定した操作が実行されます。

    > 説明　期限切れオブジェクトに対して実行可能な操作には、オブジェクトのストレージクラスを IA に変換する、オブジェクトのストレージクラスをアーカイブに変換する、オブジェクトを削除する、などがあります。

- パートの有効期限ポリシー：パートの有効期限と、期限切れのパートに対して実行する操作を指定します。
  - 有効期間：有効期間 (N 日) を指定します。 パートは、最終変更日から N 日後に削除されます。
  - 有効期限：有効期限を指定します。 指定した日付より前に変更されたすべてのパートが削除されます。

オブジェクト名のプレフィックスが、ルールに指定されたプレフィックスと一致する場合、そのオブジェクトに対してルールが適用されます。 たとえば、バケットに次のオブジェクトが保存されているとします。

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

ルールに指定されたプレフィックスが "logs/" の場合、"logs/" で始まる 3 つのオブジェクトにルールが適用されます。 ルールに指定されたプレフィックスが "doc/readme.txt" の場合、doc/readme.txt という名前のオブジェクトにのみルールが適用されます。

期限切れ削除ルールを設定することもできます。 たとえば、"logs/" で始まるオブジェクトの最終日が 30 日前の場合、指定した期限切れ削除時間に従ってオブジェクトは削除されます。

オブジェクトが期限切れルールに一致する場合、オブジェクトの GetObject または HeadObject リクエストへのレスポンスに x-oss-expiration ヘッダーが含まれます。 ヘッダーには 2 つのキーと値のペアが含まれます。expiry-date はオブジェクトの有効期限、rule-id は一致したルール ID を示します。

操作方法

方法	説明
OSS コンソール	使いやすい Web アプリケーション
Java SDK	さまざまな言語の完全な SDK デモ
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	
Node.js SDK	
Ruby SDK	

例

OSS API を使用して、バケットのライフサイクルルールを設定できます。 次の例に示すように、ライフサイクルルールは XML 形式です。

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

上の例にある要素は、次のとおりです。

- `<ID>`：ルールの一意の識別子。
- `<Status>`：ライフサイクルルールのステータス。2 つの値 (Enabled または Disabled) があります。ステータスが Enabled のルールのみが適用されます。
- `<Prefix>`：指定したプレフィックスを持つオブジェクトにのみルールが適用されます。
- `<Expiration>`：操作の有効期間。サブ要素の `<CreatedBeforeDate>` は絶対的な有効期間、`<Days>` は相対的な有効期間です。
  - CreatedBeforeDate：有効期限と、期限切れオブジェクトに対して実行される操作を指定します。指定された日付より前に変更されたオブジェクトは期限切れになり、指定した操作が実行されます。

- Days：有効期間 (N 日) と、期限切れオブジェクトに対して実行される操作を指定します。 最後に変更されてから N 日後にオブジェクトは期限切れになり、指定した操作が実行されます。

1 番目のルールでは、プレフィックスが logs/ で、10 日前に変更されたオブジェクトが削除されます。 2 番目のルールでは、プレフィックスが doc/ で、2014 年 12 月 31 日より前に変更されたオブジェクトが削除されます。 ただし、ルールのステータスが Disabled なので、このルールは有効になりません。 3 番目のルールでは、タグが "xx=1" で、60 日前に変更されたオブジェクトのストレージクラスはアーカイブに変更されます。

詳細分析

- プレフィックスとタグ
  - プレフィックスの命名規則は、オブジェクトの命名規則と同じです。
  - ルールのプレフィックスが空の場合、バケット内のすべてのオブジェクトにルールが適用されます。
  - バケットのルールに指定するプレフィックスは、一意である必要があります。 たとえば、バケットに 2 つのルールを設定し、ルールのプレフィックスがそれぞれ logs/ と logs/program の場合、エラーが返されます。
  - タグのキーを空にすることはできません。 タグには、文字、数字、スペース、および次の記号を含めることができます。

    + - = . _ : /
  - 異なるルールの一致ポリシー (プレフィックスとタグによる一致) のプレフィックスは、重複可能です。 たとえば、同じバケットにルール 1 とルール 2 が設定されているとします。 ルール 1 では、プレフィックスに `logs/` 、タグに "K1=V1" が指定されています。 ルール 2 では、プレフィックスに `logs/program` 、タグに"K1=V2" が指定されています。 ルール 1 は、プレフィックスが `logs` か `logs/program` で、タグが "K1=V1" のオブジェクトに適用されます。 ルール 2 は、プレフィックスが `logs` で、タグが "K1=V1" のオブジェクトに適用されます。
- 有効期間
  - 特定の日付にオブジェクトを削除するようにルールが設定されている場合、日付は UTC 午前 0 時で、ISO8601 形式 (2017-01-01T00:00:00.000Z など) に準拠する必要があります。 この例では、2017 年 1 月 1 日の午前 0 時以降にオブジェクトが削除されます。
  - 特定の日数以降にオブジェクトを削除するようにルールが設定されている場合、最終更新時刻 (最終変更日) と指定された日数を合計し、合計を UTC 午前 0 時のタイムスタンプに丸められます。 たとえば、オブジェクトの最終更新時刻が 2014 年 4 月 12 日の午前 01:00 で、一致ルールに指定された日数が 3 の場合、有効期限は 2017 年 4 月 16 日の 0 時です。
  - ルールに一致するオブジェクトは、指定された時間に削除されます。 通常、指定された時間の直後にオブジェクトは削除されます。
  - 変更されていないオブジェクトの場合、通常、更新時刻は作成時刻になります。 Put 操作が複数回実行されたオブジェクトの場合、最終更新時刻は、最後の Put 操作の時刻です。 オブジェクトを同じオブジェクトにコピーした場合、最終更新時刻は、オブジェクトを最後にコピーした時刻になります。
- 料金

  成功したライフサイクル非同期リクエスト操作のみが記録、課金されます。
- ルールが競合したときに実行されるアクション

- 2 つのルールに指定されたプレフィックスとタグが同じ。

ルール	プレフィックス	タグ	アクション
rule1	abc	a=1	一致するオブジェクトを 20 日後に削除する。
rule2	abc	a=1	一致するオブジェクトのストレージクラスを 20 日後にアーカイブに変更する。

結果：プレフィックスが abc、タグが "a=1" のオブジェクトは、20 日後に削除されます (最初に削除が実行されます)。 一致するオブジェクトは既に削除されているため、2 番目のルールは有効になりません。

ルール	プレフィックス	タグ	アクション
rule1	abc	a=1	一致するオブジェクトのストレージクラスを 365 日後 に IA に変更する。
rule2	abc	a=1	一致するオブジェクトのストレージクラスを 2018 年 3 月 1 日までにアーカイブに変更する。

結果：プレフィックスが abc、タグが "a=1" のオブジェクトが同時に 2 つのルールに一致する場合、オブジェクトのストレージはアーカイブに変更されます。 オブジェクトが一方のルールにのみ一致する場合、そのルールに指定されたアクションが実行されます。

- 2 つのルールに指定されたタグが同じで、ルールに指定されたプレフィックスが重複。

ルール	プレフィックス	タグ	アクション
rule1		a=1	一致するオブジェクトのストレージクラスを 20 日後に IA に変更する。
rule2	abc	a=1	一致するオブジェクトを 120 日後に削除する。

結果：タグが "a=1" の全オブジェクトのストレージクラスが、20 日後に IA に変更されます。 プレフィックスが abc、タグが "a=1" のオブジェクトは、120 日後に削除されます。

ルール	プレフィックス	タグ	アクション
rule1		a=1	一致するオブジェクトのストレージクラスを、10 日後にアーカイブに変更する。
rule2	abc	a=1	一致するオブジェクトのストレージクラスを 20 日後に IA に変更する。

結果：タグが "a=1" の全オブジェクトのストレージクラスがアーカイブに変更されます。 アーカイブオブジェクトのストレージクラスを IA に変更することができないため、プレフィックスが abc でタグが "a=1" のオブジェクトに対して、2 番目のルールは適用されません。

FAQ

- ストレージクラスを変更するか、期限切れオブジェクトを削除するようにライフサイクルルールが設定されている場合、このルールに最小保存期間はありますか。

  オブジェクトのストレージクラスを変更 (標準から IA かアーカイブ、または IA からアーカイブ) するようにライフサイクルルールが設定されている場合、このルールに最小保存期間は不要です。 ただし、有効期限が切れたオブジェクトを削除するように設定されたライフサイクルルールには、最小保存期間が必要です。 IA ストレージクラスのオブジェクトの場合、最小保存期間は 30 日、アーカイブストレージクラスのオブジェクトの場合は 60 日です。

  期限切れオブジェクトを削除するように設定されたライフサイクルルールの最小保存期間とは、オブジェクトが作成されてから削除されるまでの期間を示します。 この期間中にオブジェクトが変更された場合 (CopyObject 操作か AppendObject 操作などで)、最小保存期間は最終変更日から計算されます。

  例 1：オブジェクトの作成から 10 日後にストレージクラスを IA からアーカイブに変更します。 オブジェクトの作成時間は変更されません。 変更したオブジェクトは、50 日間以上保存する必要があります。

例 2: オブジェクトの作成から 10 日後に、CopyObject 操作でストレージクラスを IA からアーカイブに変更する場合、削除には 20 日分の料金が発生します。 オブジェクトの作成時間は更新されます。 変更したオブジェクトは、60 日間以上保存する必要があります。

- リクエストの課金ロジック

  ライフサイクルルールに従ってストレージクラスを変更したり、期限切れオブジェクトを削除した場合、リクエストが生成されます。 リクエストは、OSS で課金されます。 例:

  - ライフサイクルルールに従って、1000 個のオブジェクトのストレージクラスが標準からアーカイブに変更された場合、1000 個の POST リクエストが生成されます。
  - ライフサイクルルールに従って、1000 個の期限切れオブジェクトが削除された場合、1000 個の削除リクエストが生成されます。

  詳細は、「課金方法」をご参照ください。

- ライフサイクルルールに従ってストレージクラスの変更や期限切れオブジェクト削除が実行された場合、記録されますか。

  ライフサイクルルールに従って実行されたストレージクラスの変更や期限切れオブジェクトの削除は、ログに記録されます。 ログのフィールドは次のとおりです。

  - Operation
    - CommitTransition: オブジェクトのストレージクラスを変更するようにルールが設定されていることを示します。
    - ExpireObject: 期限切れオブジェクトを削除するようにルールが設定されていることを示します。
  - Sync Request
    - lifecycle: ライフサイクルルールによって実行される操作を示します。

5.4.2. ライフサイクルルールの設定例

このトピックでは、ライフサイクルルールの設定例を示します。

OSS API を使用して、バケット内のオブジェクトのライフサイクルルールを設定できます。 次の例に示すように、ライフサイクルルールは XML 形式です。

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

上記の例では、次の 3 つのライフサイクルルールが設定されています。

- 1 番目のルールは、logs/ というプレフィックスで始まり、10 日前に変更されたオブジェクトが削除されることを示します。
- 2 番目のルールは、doc/ というプレフィックスで始まり、2014 年 12 月 31 日より前に変更されたオブジェクトが削除されることを示します。 ただし、このルールは Disabled ステータスなので、有効になりません。
- 3 番目のルールは、タグ "xx=1" が設定され、60 日前に変更されたオブジェクトのストレージクラスがアーカイブに変更されることを示します。

ライフサイクルルールを設定するとき、次の要素を設定する必要があります。

- <ID>: 一意のルール ID を示します。

- <Status>：ライフサイクルルールのステータスを 2 つの値 (Enabled または Disabled) で示します。ステータスが Enabled のルールのみが適用されます。
- <Prefix>：特定のプレフィックスで始まるオブジェクトに対してのみルールが適用されることを示します。
- <Expiration>：期限切れオブジェクトに対して実行される操作を示します。 サブ要素 <CreatedBeforeDate> と<Days> は、それぞれ絶対的な有効期限と相対的な有効期限を示します。
  - <CreatedBeforeDate>：有効期限と、期限切れオブジェクトに対して実行される操作を指定します。 指定した日付より前に変更されたオブジェクトの期限が切れ、オブジェクトに対して指定した操作が実行されます。
  - <Days>：有効期間 (N 日) と、期限切れオブジェクトに対して実行される操作を指定します。 オブジェクトが最後に変更されてから N 日後に期限が切れ、指定した操作がオブジェクトに対して実行されます。

5.4.3. よくある質問

このドキュメントでは、ライフサイクルルールを使用してオブジェクトを管理するときに発生する可能性のある問題に答えます。

ストレージクラスを変更するか、期限切れオブジェクトを削除するようにライフサイクルルールが設定されている場合、このルールに最小保存期間はありますか。

オブジェクトのストレージクラスを変更 (標準から IA かアーカイブ、または IA からアーカイブ) するようにライフサイクルルールが設定されている場合、このルールに最小保存期間は不要です。 ただし、有効期限が切れたオブジェクトを削除するように設定されたライフサイクルルールには、最小保存期間が必要です。 IA ストレージクラスのオブジェクトの場合、最小保存期間は 30 日、アーカイブストレージクラスのオブジェクトの場合は 60 日です。

期限切れオブジェクトを削除するように設定されたライフサイクルルールの最小保存期間とは、オブジェクトが作成されてから削除されるまでの期間を示します。 この期間中にオブジェクトが変更された場合 (CopyObject 操作か AppendObject 操作などで)、最小保存期間は最終変更日から計算されます。

例 1：オブジェクトの作成から 10 日後にストレージクラスを IA からアーカイブに変更します。 オブジェクトの作成時間は変更されません。 変更したオブジェクトは、50 日間以上保存する必要があります。

例 2：オブジェクトの作成から 10 日後に、CopyObject 操作でストレージクラスを IA からアーカイブに変更する場合、削除には 20 日分の料金が発生します。 オブジェクトの作成時間は更新されます。 変更したオブジェクトは、60 日間以上保存する必要があります。

リクエストの課金ロジック

ライフサイクルルールに従ってストレージクラスを変更したり、期限切れオブジェクトを削除した場合、リクエストが生成されます。 リクエストは、OSS で課金されます。 例：

- ライフサイクルルールに従って、1000 個のオブジェクトのストレージクラスが標準からアーカイブに変更された場合、1000 個の POST リクエストが生成されます。
- ライフサイクルルールに従って、1000 個の期限切れオブジェクトが削除された場合、1000 個の削除リクエストが生成されます。

詳細は、「課金方法」をご参照ください。

ライフサイクルルールに従ってストレージクラスの変更や期限切れオブジェクト削除が実行された場合、記録されますか。

ライフサイクルルールに従って実行されたストレージクラスの変更や期限切れオブジェクトの削除は、ログに記録されます。 ログのフィールドは次のとおりです。

- Operation
  - CommitTransition：オブジェクトのストレージクラスを変更するようにルールが設定されていることを示します。
  - ExpireObject：期限切れオブジェクトを削除するようにルールが設定されていることを示します。
- Sync Request

  lifecycle：ライフサイクルルールによって実行される操作を示します。

6.ログ管理

6.1. アクセスログの設定

OSS にアクセスすると、数多くのアクセスログが生成されます。 バケットに対するアクセスログ機能を有効にすると、OSS はバケットに格納されているアクセスログに 1 時間ごとに自動的にアクセスし、指定したバケット (ターゲットバケット) にオブジェクトを生成します。 生成されるオブジェクトは、OSS の命名規則に準拠しています。 アクセスログを分析するには、Alibaba Cloud Data Lake Analytics を使用するか、Spark クラスターを使用します。 ターゲットバケットのライフサイクルルールで、ログオブジェクトのストレージクラスが「Archive 」に変換されるように設定し、ログオブログジェクトをアーカイブすることもできます。

アクセスログが格納されるオブジェクトの命名規則

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

命名規則の構成フィールドは、次のとおりです。

- `TargetPrefix` は、アクセスログが格納されるオブジェクト名のプレフィックスを示します。 このフィールドはユーザーが定義します。空白でも構いません。
- `YYYY-mm-DD-HH-MM-SS` は、オブジェクトが作成された年、月、日、時、分、および秒を示します (桁数に注意してください)。
- `UniqueString` は、OSS によって生成される文字列 (UUID) であり、ログオブジェクトを一意に識別するために使用されます。

OSS アクセスログを格納するオブジェクト名の例は、次のとおりです。

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

上記の例の説明は、次のとおりです。

- `MyLog-` は、ユーザーが指定したオブジェクトのプレフィックスです。
- `oss-example` は、ソースバケットの名前です。
- `2017-09-10-04-00-00` は、オブジェクトが作成された時刻です。
- `0000` は、OSS によって生成された文字列です。

ログファイルの形式

以下では、ログファイルの構成フィールドについて説明します。 ログファイルでは、構成フィールドがスペースで区切られて、左から右へ順番に結合されます。

名前	例	説明
Remote IP	119.xx.xx.11	リクエストを開始した IP アドレスを示します (プロキシまたはファイアウォールによって、このフィールドがブロックされる可能性があるのでご注意ください)。

名前	例	説明
Reserved	-	予約フィールドであることを示します。
Reserved	-	予約フィールドであることを示します。
Time	[02/May/2012:00:00:04 +0800]	OSS がリクエストを受信した時刻を示します。
Request-URI	“GET /aliyun-logo.png HTTP/1.1”	ユーザーリクエストの URI (クエリ文字列を含む) を示します。
HTTP Status	200	OSS から返された HTTP ステータスコードを示します。
SentBytes	5576	ユーザが OSS からダウンロードしたトラフィック量を示します。
RequestTime (ms)	71	リクエストを完了するために使用された時間量をミリ秒単位で示します。
Referer	`http://www.aliyun.com/product/oss`	リクエストを行った HTTP リファラーを示します。
User-Agent	curl/7.15.5	HTTP User-Agent のヘッダーを示します。
HostName	oss-example.oss-cn-hangzhou.aliyuncs.com	リクエストによってアクセスされるドメイン名を示します。
Request ID	505B016950xxxxxx032593A4	リクエストを一意に識別するために使用される UUID を示します。
LoggingFlag	true	アクセスログ機能が有効になっているかどうかを示します。
Requester Aliyun ID	16571xxxxxx83691	リクエストを送信した Alibaba Cloud ID を示します。匿名アクセスの場合は「 - 」です。
Operation	GetObject	リクエストタイプを示します。
Bucket	oss-example	リクエストバケット名を示します。
Key	/aliyun-logo.png	ユーザーがリクエストしたキーを示します。
ObjectSize	5576	オブジェクトサイズを示します。

名前	例	説明
Server Cost Time (ms)	17	OSS サーバーでリクエストの処理に要した時間の長さをミリ秒単位で示します。
Error Code	NoSuchBucket	OSS から返されるエラーコードを示します。
Request Length	302	ユーザーリクエストの長さをバイト単位で示します。
UserID	16571xxxxxx83691	バケット所有者の ID を示します。
Delta DataSize	280	バケットサイズの変動を示します。バケットサイズが変更されていない場合は `-` になります。
Sync Request	-	リクエストが CDN バックツーオリジンであるかどうかを示します。バックツーオリジンリクエストでない場合は `-` になります。
Reserved	-	予約フィールドであることを示します。

詳細分析

- ソースバケットとターゲットバケットは、同じバケットでも、同じアカウントで所有し、同じリージョンに属する異なるバケットでも構いません。 複数のソースバケットのログを同じターゲットバケットに格納することもできます。 この場合、異なるソースバケットのログに対して、異なる "TargetPrefix" 値を指定することを推奨します。
- OSS では、バケットのアクセスログを 1 時間ごとに格納するオブジェクトが生成されます。 ただし、過去 1 時間のリクエストは、その 1 時間前に生成されたオブジェクト、または 1 時後に生成されるオブジェクトに記録される場合があります。
- バケットのアクセスログを格納するオブジェクトが OSS で生成されるたびに、PUT 操作とその操作で占有されるストレージ領域が記録されます。 ただし、PUT 操作によって生成されるトラフィックは記録されません。 アクセスログが格納される生成済みのオブジェクトに対しては、一般的な OSS 操作を実行できます。
- OSS では、プレフィックス `「x-」` の付いたすべてのクエリ文字列パラメーターは、無視されます。 ただし、これらのパラメーターはアクセスログに記録されます。 大量のアクセスログから特定のリクエストを簡単に識別するためには、リクエストの URL にプレフィックス `「x-」` を付けた クエリ文字列パラメータを追加します。 例:

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

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

  上記の 2 つのリクエストに対して、OSS から同じ結果が返されます。 ただし、 `x-user = admin` パラメーターを使用して検索することで、リクエストを簡単に見つけることができます。

- `-` が、OSS ログの任意のフィールドに表示される場合があります。 データが不明であるか、フィールドが現在のリクエストに対して無効であることを示します。
- 今後、OSS ログの末尾にフィールドが追加される予定です。 ログ処理ツールを開発する場合、開発者は潜在的な互換性の問題に注意することをお推奨します。

参照情報

- OSS コンソールでのログ設定についての詳細は、「ログの設定 (Set logging)」をご参照ください。
- 関連 API の詳細は、「PutBucketLogging」、「DeleteBucketLogging」、および「GetBucketLogging」をご参照ください。
- 関連する Java SDK についての詳細は、「ログの設定 (Set logging)をご参照ください。

6.2. リアルタイムログ照会

OSS にアクセスする際、多数のアクセスログが生成されます。 リアルタイムログ照会機能は、OSS と Log Service (LOG) を組み合わせて、OSS コンソールで OSS アクセスログをクエリし、稼働状況の監査、アクセス統計の収集、例外のバックトラッキング、トラブルシューティングなどを実装できます。 この機能により、作業効率の向上と、データに基づく決断が可能になります。

リアルタイムログ照会とアクセスログの比較

- リアルタイムログ照会
  - 3 分以内にログが Log Service インスタンスに転送されるので、OSS コンソールでリアルタイムにログが表示されます。
  - ログ分析サービスと標準的な分析レポートにより、データを簡単にクエリできます。
  - Raw ログをリアルタイムにクエリし、分析できるほか、バケット、オブジェクト名、API 操作、時間を基準にログをフィルタリングできます。
- アクセスログ
  - バケットのログ機能を有効にできます。 事前定義された命名規則に基づき、1 時間単位でオブジェクトが自動生成され、バケットのアクセスログの保存と、指定されたバケットへのオブジェクトの書き込みを行います。
  - Alibaba Cloud Data Lake Analytics を使用するか、Spark クラスターを構築して、アクセスログを分析できます。
  - 長期のアーカイブのため、ログオブジェクトのストレージクラスをアーカイブに変更するよう、バケットのライフサイクル管理ルールを設定できます。

設定方法

コンソール：リアルタイムログ照会の有効化

クエリ方法

リアルタイムログ照会機能には、次のクエリ方法があります。

- リアルタイム Raw ログクエリ

  Raw ログの期間とクエリ文を指定して、次の操作を容易に実行できます。

  - API など、ある期間内のフィールドの分布を分析します。

- 今後のクエリに使用できるように、フィールドを基準に分析対象レコードをフィルタリングします。たとえば、前日のオブジェクト削除操作をバケット、オブジェクト名、API 名でフィルタリングしたり、削除時間とアクセス IP アドレスをクエリするなどします。
- ある期間のバケットのページビュー (PV)、ユニークアクセス (UV)、最大レイテンシなど、OSS アクセスレコードの統計を収集します。

- ログレポートクエリ

  リアルタイムログ照会機能では、すぐに利用可能な 4 つのレポートが提供されます。

  - **Access Center**：PV、UV、トラフィック、インターネットのアクセス分布など、全体的な稼働ステータスが表示されます。
  - **Audit Center**：読み取り、書き込み、削除など、オブジェクトに対する操作の統計が表示されます。
  - **Operation Center**：リクエスト数や失敗した操作の分布など、アクセスログの統計が表示されます。
  - **Performance Center**：インターネット経由のダウンロードやアップロードのパフォーマンス、異なるネットワークやオブジェクトサイズでの送信のパフォーマンス、保存したオブジェクトとダウンロードしたオブジェクトとの差分リストなど、パフォーマンスの統計が表示されます。

- Log Service コンソールでのクエリ

  Log Service コンソールで、OSS アクセスログを確認できます。

課金方法

OSS のリアルタイムログ照会機能では、過去 7 日間のログを無料でクエリできます。 設定したログ保存期間が 7 日より長い場合、超過した日数分の料金が別途発生します。 インターネット経由で Log Service のデータを読み取りまたは書き込みした場合、追加料金が発生します。

課金基準の詳細は、Log Service の「課金方法」をご参照ください。

7.静的 Web サイトホスティング

7.1. 静的 Web サイトホスティングの設定

OSS の PutBucketWebsite API を使用して、バケットを静的 Web サイトホスティングモードに設定し、バケットエンドポイントからアクセスできます。

> 説明 PutBucketWebsite API の詳細は、「PutBucketWebsite」をご参照ください。

選択したバケットが中国 (杭州) にある場合、設定が有効になった後、静的 Web サイトのドメインは、次のようになります。

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

> 説明 デフォルトのエンドポイントを使用して、インターネット経由で中国本土または香港にある OSS の Web ページオブジェクトにアクセスした場合、`Content-Disposition:'attachment=filename;'` がレスポンスヘッダーに自動的に追加されます。 つまり、ブラウザーから Web オブジェクトにアクセスすると、オブジェクトは添付としてダウンロードされます。 カスタムドメインを使用して OSS にアクセスした場合、この情報はレスポンスヘッダーに追加されません。 カスタムドメインを使用して OSS にアクセスする方法の詳細は、「カスタムドメインのバインド」をご参照ください。

OSS の次の機能を使用すると、OSS でホストされる静的 Web サイトの管理が容易になります。

- インデックスドキュメントのサポート

  インデックスドキュメントとは、ユーザーが静的 Web サイトのルートドメインに直接アクセスしたとき、OSS によって返されるデフォルトのインデックスページ (index.html など) のことです。 バケットを静的 Web サイトホスティングモードに設定した場合、インデックスドキュメントを指定する必要があります。

- エラードキュメントのサポート

  エラードキュメントとは、ユーザーが静的 Web サイトにアクセスしたときに HTTP 4XX エラー (404 NOT FOUND など) が発生した場合、OSS によって返されるエラーページのことです。 エラードキュメントを指定すると、エンドユーザーに適切なエラーメッセージを表示できるようになります。

たとえば、インデックスドキュメントを index.html、エラードキュメントを error.html、バケット名を oss-sample、エンドポイントを oss-cn-hangzhou.aliyuncs.com に設定したとします。

- `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/` と `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/directory/` にアクセスする場合、実際は `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/index.html` にアクセスします。
- `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/object` にアクセスした際、オブジェクトが存在しなかった場合、`http://oss-sample.oss-cn-hangzhou.aliyuncs.com/error.html` が返されます。

操作方法

操作方法	説明
コンソール	直感的で使いやすい Web アプリケーション
Java SDK	さまざまな言語の SDK デモ
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	
Node.js SDK	
Ruby SDK	

詳細分析

- 静的 Web サイトでは、すべての Web ページが静的コンテンツ (クライアントで実行される JavaScript などのスクリプトを含む) で構成されます。 OSS では、サーバーで処理する必要があるコンテンツ (PHP、JSP、APS.NET などのコンテンツ) はサポートされていません。
- カスタムドメインを使用してバケットベースの静的 Web サイトにアクセスするには、カスタムドメインをバインドします。
- バケットを静的 Web サイトホスティングモードに設定する場合
  - インデックスドキュメントは必須、エラードキュメントはオプションです。
  - 指定するインデックスドキュメントとエラードキュメントは、バケット内のオブジェクトでなければなりません。

  > 説明　アーカイブバケットを使用する場合、標準オブジェクトか復元済みアーカイブオブジェクトをインデックスドキュメントおよびエラードキュメントとして指定する必要があります。 そうしないと、静的 Web サイトにアクセスできません。

- バケットを静的 Web サイトホスティングモードに設定した後
  - 静的 Web サイトのルートドメインへの匿名アクセスの場合、インデックスドキュメントが返されます。静的 Web サイトのルートドメインへの許可アクセスの場合、GetBucket 操作の結果が返されます。
  - ユーザーが静的 Web サイトまたは OSS に存在しないオブジェクトのルートドメインにアクセスした場合、指定されたオブジェクトが返されます。 このリクエストと、生成されたトラフィックに対して、課金されます。

参照

- チュートリアル: カスタムドメイン名を使用して静的 Web サイトをホストする
- チュートリアル: 静的 Web サイトホスティングの設定

7.2. チュートリアル：カスタムドメイン名を使用して静的 Web サイトをホストする

Alibaba Cloud Object Storage Service（OSS）で静的 Web サイトをホストするとします。 ドメイン（examplewebsite.com など）を登録したので、 `http://examplewebsite.com` および `http://www.examplewebsite.com` へのリクエストは OSS コンテンツから処理する必要があります。 OSS でホスティングする 既存の静的 Web サイトを持っている場合でも、最初から始める場合でも、この例を使用して Alibaba Cloud OSS で Web サイトをホストする方法を学ぶことができます。

前提条件

このチュートリアルでは、以下のサービスについて説明します。

- ドメイン名登録

  `examplewebsite.com` などの登録ドメイン名がない場合は、登録する登録機関を選択してください。 Alibaba Cloud はドメイン名登録サービスも提供しています。 詳しくは、「Alibaba Cloud Domain サービス」をご参照ください。

- Alibaba Cloud OSS

  Alibaba Cloud OSS を使用してバケットを作成し、サンプルの Web サイトページをアップロードし、他のユーザーがコンテンツを閲覧できるように権限を設定してから、 Web サイトホスティング用にバケットを設定します。 この例では、 `http://examplewebsite.com` および `http://www.examplewebsite.com` に対するリクエストを許可したいので、2つのバケットを作成します。 コンテンツを 1 つのバケットだけでホストし、コンテンツをホストする バケットに要求をリダイレクトするように他のバケットを構成します。

- ALibaba Cloud DNS

  ドメインネームシステム（DNS）プロバイダとして、Alibaba Cloud DNS を構成します。 この例では、ドメイン名を Alibaba Cloud DNS に追加し、OSS 割り当てアクセスドメイン名の代わりにドメイン名を使用して OSS バケットにアクセスできるように CNAME レコードを定義します。

  この例では、Alibaba Cloud DNS を使用しています。 Alibaba Cloud DNS を使用すること をお勧めします。 ただし、さまざまなレジストラを使用して、 OSS バケットを指す CNAME レコードを定義できます。

> **説明**
>
> このチュートリアルでは、ドメイン名として `examplewebsite.com` を使用しています。 このドメイン名を登録したものに置き換えます。

手順 1：ドメインを登録する

すでに登録済みドメインをお持ちの場合は、この手順をスキップできます。 Web サイトをホストしたことがない場合は、 まず `examplewebsite.com` などのドメインを登録します。 Alibaba Cloud Domain サービスを使用してドメインを登録できます。

詳細については、「Alibaba Cloud Domainクイックスタートでドメイン名を購入する」をご参照ください。

手順 2: バケットを作成して設定し、データをアップロードする

`examplewebsite.com` などのルートドメインと http://www.examplewebsite.com などのサブドメインの両方からの要求をサポートするために、2 つのバケットを作成します。`http://www.examplewebsite.com` . 一方のバケットはコンテンツを格納するために使用され、もう一方のバケットは コンテンツを格納するバケットに要求をリダイレクトするために使用されます。

手順 2.1: 2つのバケットを作成する

このステップでは、Alibaba Cloud アカウントの認証情報を使用して Alibaba Cloud OSS コンソールにログオンし、次の 2 つのバケットを作成します。

- originbucket: コンテンツを保存する
- redirectbucket: リクエストを originbucket にリダイレクトする

1. OSS コンソールにログインします。
2. たとえば、originbucket と redirectbucket の 2 つのバケットを作成します。1 つはコンテンツを格納するためのもの、 もう 1 つはコンテンツを格納するバケットに要求をリダイレクトするためのものです。 2つのバケットのアクセス 制御リスト（ACL）を「公開読み取り」に設定して、誰もがバケットの内容を 見ることができるようにします。

   詳細な手順については、「バケットの作成」をご参照ください。
3. originbucket と redirectbucket のアクセスドメイン名を書き留めます。 後の手順でそれらを使用します。 次の図に示すように、 バケットの概要タブページでバケットのアクセスドメイン名を見つけることができます。
4. ウェブサイトのデータを originbucket にアップロードします。

   あなたはあなたのコンテンツをルートドメインバケット originbucket の外にホストし、 あなたはサブドメインバケット redirectbucket のリクエストを ルートドメインバケット originbucket にリダイレクトします。 どちらのバケットにもコンテンツを保存 できます。

   この例では、コンテンツを originbucket バケットでホスト します。 コンテンツは、テキストファイル、写真、ビデオなど、あらゆる種類のファイルにすることができます。 Web サイトをまだ作成していない場合は、この例では 2 つのファイルしか必要ありません。 一方のファイルは Web サイトのホームページとして使用され、もう一方のファイルは Web サイトのエラーページとして使用されます。

   たとえば、次の HTML を使用して index.html という名前のファイルを 1 つ作成し、 それをバケットにアップロードできます。 後の手順では、このファイル名を Web サイトのデフォルトのホームページとして使用します。

```
<html>
  <head>
     <title>Hello OSS! </title>
     <meta charset = "utf-8">
  </head>
  <body>
     <p>Now host on Alibaba Cloud OSS</p>
     <p>This is the index page</p>
  </body>
</html>
```

次の HTML を使用して error.html という名前の別のファイルを作成し、それをバケットにアップロードします。 このファイルは、Web サイトの 404 エラーページとして使用されています。 後の手順で、このファイル名を Web サイトのデフォルトの 404 ページとして使用します。

```
<html>
<head>
  <title>Hello OSS! </title>
  <meta charset="utf-8">
</head>
<body>
  <p>This is the 404 error page</p>
</body>
</html>
```

手順 2.2: Webサイトホスティング用のバケットを設定する

Web サイトホスティング用のバケットを設定すると、OSS に 割り当てられたアクセスドメイン名を使用して Web サイトにアクセスできます。

このステップでは、originbucket を Web サイトとして設定します。

1. OSS コンソールにログインします。
2. バケット名リストから、*originbucket* を選択します。
3. **[基本設定]**タブをクリックして、Static Page 領域を見つけます。
4. **[編集]**をクリックしてから、以下の情報を入力します。
   - **デフォルトホームページ**: インデックスページ（Web サイトの index.html に相当）。 バケットに保存されている HTML ファイルのみを 使用できます。 この例では、 *index.html* と入力します。
   - **デフォルトの 404 ページ**: 誤まったパスにアクセスしたときに返されるデフォルトの404ページ。 バケットに保存されている HTML ファイルと画像ファイルのみを使用できます。 このフィールドを 空白のままにすると、デフォルトの 404 ページが無効になります。 この例では、 *error.html* と入力します。
5. **[保存]**をクリックします。

手順 2.3: リダイレクト用のインデックスページを設定する

エラーページとo riginbucket のデフォルトのホームページを設定したので、 redirectbucket のデフォルトのホームページも設定する必要があります。

リダイレクト用の インデックスページを設定するには、次の手順を実行します。

1. OSS コンソールにログインします。
2. バケット名リストから、*redirectbucket* を選択します。
3. **[基本設定]**タブをクリックして、 静的ページ領域を見つけます。
4. **[編集]**をクリックし、 *index.html* と 既定のホームページテキストボックスに入力します。
5. **[保存]**をクリックします。

手順 3: ドメイン名を OSS バケットにバインドする

ルートドメイン `examplewebsite.com` と OSS バケット **originbucket** ができたので、OSS によって割り当てられたドメイン名の代わりに独自のドメイン名を使用して OSS バケットにアクセスできるように、ドメインを OSS バケットにバインドします。

この例では、ドメイン `examplewebsite.com` を OSS バケット **originbucket** にバインドする前に、 OSS に割り当てられたドメイン名 originbucket.oss-cn-beijing.aliyuncs.com を使用してバケット originbucket にアクセスする必要があります。 ドメイン `examplewebsite.com` をバインドしたら、 この `examplewebsite.com` を使用して OSS バケットにアクセスできます。

同様に、OSS が割り当てたドメイン名 originbucket.oss-cn-beijing.aliyuncs.com の代わりに `www.examplewebsite.com` を使用して OSS バケットにアクセスできるように、サブドメイン `www.examplewebsite.com` を OSS バケット **redirectbucket** に バインドする必要もあります。

ルートドメイン `examplewebsite.com` を OSS バケットの **originbucket** にバインドするには、次の手順に従います。

1. OSS コンソールにログインします。
2. バケット名リストから、 *originbucket* を選択します。
3. **[ドメイン名]**タブをクリックします。
4. **[ユーザードメインのバインド]**をクリックして[ユーザードメインのバインド]ダイアログボックスを開きます。
5. ユーザードメインテキストボックスに、 ルートドメイン *examplewebsite.com* を入力します。
6. CNAME レコードを originbucket に定義します。
   - ドメイン名が Alibaba Cloud アカウントで解決されたら、 **Add CNAME Record Automatically** スイッチを開くことができます。 次に**[送信]**をクリックします。
   - ドメイン名が Alibaba Cloud のプライマリアカウントで解決されていない場合、 **CNAME レコードを自動的に追加**スイッチは無効になっています。 CNAME レコードを手動で追加するには、 次の手順を実行し、 **[送信]**をクリックします。
     a. Alibaba Cloud DNS にドメイン名を追加します。
        - ドメイン名が Alibaba Cloud に登録されている場合は、 自動的に Alibaba Cloud の DNS リストに追加されます。 このステップはスキップできます。
     b. Alibaba Cloud DNS コンソールで、自分のドメイン名を見つけます。
     c. ドメイン名をクリックするか、 **[設定]**リンクをクリックします。
     d. **[レコードの追加]**をクリックします。

e. レコードを追加ダイアログボックスで、Type ドロップダウンボックスから *CNAME* を選択し、Value テキストボックスにバケットの OSS ドメイン名を入力します。 この例では、**originbucket.oss-cn-beijing.aliyuncs.com** と入力します。

f. **[確認]**をクリックします。

7. 上記の手順に従って、サブドメイン `www.examplewebsite.com` を OSS バケット **redirectbucket** にバインドします。

手順 4：Web サイトのリダイレクトを設定する

Web サイトのホスティング用にバケットを設定し、カスタムドメインを OSS バケットにバインドしたので、**redirectbucket** を設定します。 `http://www.examplewebsite.com` へのすべての要求を `http://examplewebsite.com` にリダイレクトするようにします。

Web サイトのリダイレクトを設定するには、次の手順に従います。

1. OSS コンソールにログインします。
2. バケット名リストから、*redirectbucket* を選択します。
3. **[基本設定]**タブをクリックすると、Back-to-Origin があります。
4. **[編集]**をクリックしてから**[ルールの作成]**をクリックします。
5. 次のように 404 リダイレクトルールを作成します。
   i. Back-to-Origin で、*リダイレクト*を選択します。
   ii. Back-to-Origin When で、*HTTP Status* コードを **404** に設定します。
   iii. Back-to-Origin URL で、*プレフィックスまたはサフィックスを追加*を選択し、**originbucket** のドメイン名を入力します。 この例では、**examplewebsite.com** と入力します。
   iv. **[確認]**をクリックします。

手順 5：（オプション）Alibaba Cloud CDN でウェブサイトをスピードアップ

あなたはあなたのウェブサイトのパフォーマンスを向上させるために Alibaba クラウドコンテンツデリバリーネットワーク（CDN）を使用することができます。 CDN はあなたのウェブサイトファイル（HTML、画像、ビデオなど）を世界中のデータセンターから利用可能にします。 これらはエッジノードと呼ばれます。 訪問者があなたのウェブサイトからファイルを要求すると、Alibaba Cloud CDN は自動的にその要求を最も近いエッジノードにあるファイルのコピーにリダイレクトします。 これにより、訪問者が遠くにあるデータセンターからコンテンツを要求した場合よりもダウンロード時間が短縮されます。

Alibaba Cloud CDN は、指定した期間、コンテンツをエッジノードにキャッシュします。 有効期限よりも長い期間キャッシュされているコンテンツを訪問者がリクエストした場合、CDN はオリジンサーバーをチェックして、コンテンツの新しいバージョンが利用可能かどうかを確認します。 それ以降のバージョンが利用可能な場合、CDN は 新しいバージョンをエッジノードにコピーします。 元のコンテンツに加えた変更は、訪問者がコンテンツを要求したときにエッジノードに複製されます。 ただし、有効期限が切れる前は、コンテンツは以前のバージョンのままです。 元のコンテンツに行った変更がリアルタイムで CDN キャッシュに自動的に更新されるように、**CDN キャッシュの自動更新**スイッチを開くことをお勧めします。

CDN を使って **originbucket** を高速化するには、次の手順に従います。

1. CDN コンソールに CDN ドメインを追加します。 詳しい手順については、手順 2 をご参照ください。 CDNクイックスタートに CDN ドメインを追加します。
2. Alibaba Cloud DNS コンソールで CNAME レコードを定義します。 詳しい手順については、Alibaba Cloud DNSの設定を参照。

3. OSS コンソールで自動更新 CDN キャッシュスイッチを開きます。
   i. OSS コンソールにログインします。
   ii. バケット名リストから、 *originbucket* を選択します。
   iii. [ドメイン名]タブをクリックします。
   iv. 自動更新の CDN キャッシュを開く。
4. CDN で *redirectbucket* をスピードアップするために前のステップに従ってください。

手順 6：Web サイトをテストする

Web サイトが正しく機能していることを確認するには、ブラウザで次の URL を試してください。

URL	結果
`http://examplewebsite.com`	originbucket の索引文書を表示します。
originbucket に存在しないファイルのURL（ `http://examplewebsite.com/abc` など）	originbucket のエラードキュメントを表示します。
`http://www.examplewebsite.com`	リクエストを `http://examplewebsite.com` にリダイレクトします。
`http://www.examplewebsite.com/abc`	リクエストを `http://examplewebsite.com/abc` にリダイレクトします。

> **説明** 場合によっては、予想される動作を確認するために Web ブラウザのキャッシュを消去する必要があります。

クリーンアップ

静的な Web サイトを学習課題としてのみ作成した場合は、 アカウントに不要な料金が請求されないように、割り当てた Alibaba Cloud リソースを必ず削除します。 Alibaba Cloud リソースを削除すると、 その Web サイトは利用できなくなります。

クリーンアップするには、次の手順に従います。Alibaba Cloud CDN コンソールで

1. ドメイン名を無効にしてから削除します。
2. Alibaba Cloud DNS コンソールで CNAME レコードを削除します。
3. Alibaba Cloud OSS コンソールで OSS ファイルとバケットを削除します。

8.モニタリングサービス

8.1. モニタリングサービスの概要

OSS モニタリングサービスは、基本的なシステムの運用状態、パフォーマンス、メータリングなどのメトリックデータを詳細に示します。 カスタムアラームサービスを併せて使用することにより、リクエストのトラッキング、使用率の分析、ビジネストレンドに関する統計の収集、およびシステムの障害を迅速に発見し診断することできます。

OSS メトリックのインジケーターは、基本サービスインジケーター、パフォーマンスインジケーター、メータリングインジケーターなどのインジケーターグループに分類されます。 詳しくは、「モニタリングインジケーターリファレンス」をご参照ください。

高いリアルタイムパフォーマンス

リアルタイムのパフォーマンスモニタリングでは、潜在的なピーク/谷間の問題を明らかにし、実際の変動を表示し、ビジネスシナリオの分析と評価に関する見通しを提供します。 OSS のリアルタイムメトリックインジケーター (メータリングインジケーターを除く) を使用することにより、1 分未満の出力遅延で、分単位でのメトリックデータの収集と集約が可能になります。

メータリングインジケーターの説明

メータリングインジケーターでは、次の機能を使用します。

- メータリングエントリは、時間単位で収集、集約、出力されます。
- ただし、出力遅延が最大で 30 分発生する可能性があります。
- メータリングの時刻は、関連する統計期間の開始時刻を表します。
- メータリングのデータ取得終了時刻は、当月最後のメータリングデータ統計期間の終了時刻です。 当月にメータリングデータが生成されなかった場合、メータリングデータの取得終了時刻は、当月初日の午前 0 時です。
- 最大量のメータリングエントリが通知のためプッシュされます。 正確なメータリングデータについては、[課金情報管理] に移動し、「使用状況レコード」をクリックします。

たとえば、ユーザーが毎分平均 10 回、リクエストを行ってデータをアップロードするとします。 この場合、2016-05-10 08:00:00 から 2016-05-10 09:00:00 の時間、ユーザーの put クラスリクエスト数の測定データ値は 600 回 (10 * 60 分) になります。データ時間が 2016-05-10 08:00:00 の場合、データは 09:30:00 に出力されます。 このデータが 2016-05-01 00:00:00 から現在までの間で最後の測定モニタリングデータである場合、測定データ取得の当月終了時刻は、2016-05-10 09:00:00 です。 ユーザーが 2016 年 5 月に測定データを生成しなかった場合、測定収集の終了時刻は 2016-05-01 00:00:00 です。

OSS アラームサービス

最大 1,000 個のアラームルールを設定することができます。

アラームルールは、他のメトリックインジケーターに設定し、アラームモニタリングに追加することができます。 また、1 つのメトリックインジケーターに複数のアラームルールを設定することもできます。

- アラームサービスの詳細については、「アラームサービスの概要」をご参照ください。
- OSS アラームサービスの使用方法については、「アラームサービスユーザーガイド」をご参照ください。
- OSS メトリックインジケーターの詳細については、「モニタリングインジケーターリファレンス」をご参照ください。

メトリックデータ保持ポリシー

メトリックデータは 31 日間保持され、有効期限が切れると自動的に消去されます。 メトリックデータをオフラインで分析したり、過去のメトリックデータをダウンロードし、31 日間以上保存するには、適切なツールまたは入力コードを使用し、Cloud Monitor のデータストレージを読み取ります。 詳しくは、「API を介したメトリックデータのアクセス」をご参照ください。

コンソールには過去 7 日間までのメトリックデータが表示されます。 7 日前より古い履歴メトリックデータを表示するには、Cloud Monitor SDK を使用します。 詳しくは、「API を介したメトリックデータのアクセス」をご参照ください。

API を介したメトリックデータへのアクセス

CloudMonitor の API を使用すると、OSS メトリックデータにアクセスできます。 使用方法については、以下をご参照ください。

- CloudMonitor API Reference
- Cloud monitoring SDK リファレンス
- メトリック項目リファレンス

モニタリング、診断、トラブルシューティング

次のドキュメントには、OSS 管理に関連するモニタリング、診断、およびトラブルシューティングの詳細が記載されています。

- リアルタイムサービスモニタリング

  モニタリングサービスを使用して、OSS の実行状態とパフォーマンスをモニターする方法について説明します。

- トラッキングと診断

  OSS モニタリングサービスとログ機能を使用して問題を診断する方法、およびトラッキングと診断のため、ログファイル内の関連情報を関連付ける方法について説明します。

- トラブルシューティング

  一般的な問題と、対応するトラブルシューティングの方法について説明します。

考慮事項

OSS バケットはグローバルに一意である必要があります。 バケットを削除した後、削除したバケットと同じ名前の別のバケットを作成すると、削除したバケットに設定されていたモニタリングルールとアラームルールが新しいバケットに適用されます。

8.2. モニタリングサービスのユーザーガイド

OSS モニタリングエントリ

OSS モニタリングサービスは Alibaba Cloud Console で利用することができます。 次のいずれかの方法で OSS モニタリングサービスにアクセスすることができます。

- 「OSS コンソール」にログインし、OSS 概要ページの右側にある [管理] をクリックします。
- OSS モニタリングサービスを表示するには、「CloudMonitor コンソール」にログインします。

OSS モニタリングページ

OSS モニタリングページは、次の 3 つのタブで構成されています。

- ユーザー概要
- バケットリスト
- アラームルール

OSS モニタリングページは自動更新をサポートしていません。 最新のデータを表示するには、右上にある **[更新]** をクリックします。

OSS コンソールにログインするには、**「Object Storage Serviceコンソールに移動」**をクリックします。

ユーザー概要

ユーザー概要ページには、ユーザーモニタリング情報、最新の月の統計、ユーザーレベルのメトリックインジケーターなど、モニタリング情報がユーザーレベルで表示されています。

- ユーザーモニタリング情報

  このモジュールでは、バケットおよび関連アラームルールの合計数が表示されます。

  - ■ パラメーターは次のとおりです。: 作成したすべてのバケットを表示するには、**[バケット番号]** の横の番号をクリックします。
    - **[アラームルール数]**、[アラーム中]、[禁止数]、または [アラーム済み] の横の数字をクリックすると、情報の詳細が表示されます。
    - **[アラーム中]** は、アラームステータスのアラームを表しています。
    - **[禁止数]** は、現在無効になっているアラームを表しています。
    - [アラーム済み] は、最近アラームステータスに変更されたアラームを参照します。

- 月間統計

  このモジュールは、当月の初日の午前 0 時からメータリングデータ取得終了時刻までの期間中に使用した課金 OSS リソースに関する情報を表示します。 次のインジケーターが表示されます。

  - ストレージ使用率
  - インターネット送信量
  - Put リクエスト数
  - Get リクエスト数

  各値の単位の桁は、自動的に調整されます。 選択した値の上にカーソルを合わせると、正確な値が表示されます。

- ユーザーレベルのメトリックインジケーター

  このモジュールはユーザーレベルのメトリックチャートとテーブルを表示し、[モニタリングサービス概要] と [クエリステータス詳細] で構成されています。

  対応するメトリックチャートまたはテーブルを表示するには、事前に決められた時間範囲を選択するか、カスタム時間ボックスで時間範囲を定義します。

  - 時間範囲オプションは以下から選択します。: 1 時間、6 時間、12 時間、1 日、7 日。 既定のオプションは 1 時間です。
  - カスタム時間ボックスを使用すると、開始時刻と終了時刻を分レベルで定義できます。

メトリックチャート / テーブルは、次の表示モードをサポートしています。

- 凡例の非表示: 次の図に示すように、凡例をクリックすることで、対応するインジケーター曲線を非表示にすることができます。
- メトリックチャートの右上にある

  

  アイコンをクリックすると、チャートが拡大されます。 テーブルはズームできません。
- メトリックチャートの右上にある

  

  アイコンをクリックすると、表示されているメトリックインジケーターのアラームルールを設定できます。 詳しくは、「アラームサービスユーザーガイド」をご参照ください。 注記: テーブルおよびメータリング参照インジケーターのアラームルールは設定できません。
- グラフの曲線領域内にカーソルを置き、マウスの左ボタンを押しながらマウスをドラッグすると、時間範囲が拡張されます。 [ズームのリセット] をクリックすると、元の時間範囲に戻ります。

- サービスモニタリング概要

  [サービスモニタリング概要] ページには、次の主要なメトリックチャートが表示されます。

  - ユーザーレベルの可用性/有効なリクエスト率。可用性と有効なリクエストの割合の 2 つのメトリックインジケーターが含まれます。
  - ユーザーレベルのリクエスト / 有効なリクエスト。リクエストの合計数と有効なリクエストの数の 2 つのインジケーターが含まれます。
  - ユーザーレベルのトラフィック。インターネット送信量、インターネット受信量、イントラネット送信量、イントラネット受信量、CDN 送信量、CDN 受信量、リージョン間レプリケーションの送信量、リージョン間レプリケーションの受信量の 8 つのメトリックインジケーターが含まれます。
  - ユーザーレベルのリクエストステータスの分布。選択した時間範囲内の各タイプのリクエストの数と割合を表示するテーブルです。

- リクエストステータスの詳細

  [リクエストステータスの詳細] ページでは、リクエストステータスの分布のメトリックデータが次のメトリックチャートに表示されます。

  - サーバーエラーのリクエスト回数
  - サーバーエラーのリクエスト率
  - ネットワークエラーのリクエスト回数
  - ネットワークエラーのリクエスト率
  - クライアントエラーのリクエスト回数。リソースが見つからないことを示すエラーリクエストの数、権限付与エラーリクエストの数、クライアント側タイムアウトエラーリクエストの数、その他のクライアント側エラーリクエストの数の 4 つのメトリックインジケーターが含まれます。
  - クライアントエラーのリクエスト率。リソースが見つからないことを示すエラーリクエストの割合、権限付与エラーリクエストの割合、クライアント側タイムアウトエラーリクエストの割合、その他のクライアント側エラーリクエストの割合の 4 つのメトリックインジケーターが含まれます。
  - リダイレクトのリクエスト回数。正常に終了したリクエストの数とリダイレクトのリクエストの数の 2 つのメトリックインジケーターが含まれます。

- 成功リダイレクト率。正常に終了したリクエストの割合とリダイレクトリクエストの割合の 2 つのメトリックインジケーターが含まれます。

バケットリスト

- バケットリスト情報

  [バケットリスト] タブページには、バケット名、リージョン、作成時刻、当月のメータリング統計、および関連操作などの情報が表示されます。

  - 表示パラメーターは次のとおりです。当月のメータリング統計には、各バケットのストレージサイズ、インターネッ送信量、Put リクエスト数、Get リクエスト数が表示されます。
  - [モニタリングチャート] または対応するバケット名をクリックすると、バケットモニタリングビューページに移動します。
  - 目的のバケットの横にある [アラームルール] をクリックするか、または [アラームルール] タブに移動すると、バケットのすべてのアラームルールが表示されます。
  - 左上の検索ボックスに目的のバケット名を入力して、バケットを表示します (ファジーマッチでの検索が可能です)。
  - 目的のバケット名の前にあるチェックボックスをオンにし、[アラームルールの設定] をクリックすると、アラームルールをバッチで設定できます。 詳しくは、「アラームサービスユーザーガイド」をご参照ください。

- バケットレベルのモニタリングビュー

  バケットリストで目的のバケット名の横の [モニタリングチャート] をクリックすると、バケットモニタリングビューページに移動します。

  バケットモニタリングビューでは、次の 6 つのインジケータグループに基づくメトリックチャートが表示されます。

  - モニタリングサービスの概要
    - リクエストステータスの詳細
    - 測定参照
    - 平均レイテンシ
    - 最大レイテンシ
    - 成功リクエストカテゴリ

  測定参照を除いて、他のインジケーターは 60 の集計粒度で表示されます。 バケットレベルのメトリックチャートの既定の時間範囲は過去 6 時間ですが、ユーザーレベルのメトリックチャートの既定の時間範囲は過去 1 時間です。 左上の [バケットリストに戻る]をクリックし、[バケットリスト] タブに戻ります。

- モニタリングサービスの概要

  このインジケーターグループはユーザーレベルのサービスモニタリングの概要と同様ですが、前者はバケットレベルでメトリックデータを表示します。 主なメトリックチャートは次のとおりです。

  - 有効リクエスト可用性。可用性と有効なリクエストの割合の 2 つのメトリックインジケーターが含まれます。
  - 合計 / 有効リクエスト。リクエストの合計数と有効なリクエストの数の 2 つのメトリックインジケーターが含まれます。
  - オーバーフロー。インターネット送信量、インターネット受信量、イントラネット送信量、イントラネット受信量、CDN 送信量、CDN 受信量、リージョン間レプリケーションの送信量、リージョン間レプリケーションの受信量の 8 つのメトリックインジケーターが含まれます。
  - リクエストステータス回数。これは、選択した時間範囲内の各タイプのリクエストの数と割合を表示するテーブルです。

- リクエストステータスの詳細

  このインジケーターグループは、ユーザーレベルのリクエストステータスの詳細と同様ですが、メトリックデータをバケットレベルで表示します。 主なメトリックチャートは次のとおりです。

  - サーバーエラー回数
  - サーバーエラー率
  - ネットワークエラー回数
  - ネットワークエラーリクエスト率
  - クライアントエラーリクエスト回数。リソースが見つからないことを示すエラーリクエストの数、権限付与エラーリクエストの数、クライアント側タイムアウトエラーリクエストの数、その他のクライアント側エラーリクエストの数の 4 つのメトリックインジケーターが含まれます。
  - クライアントエラーリクエスト率。リソースが見つからないことを示すエラーリクエストの割合、権限付与エラーリクエストの割合、クライアント側タイムアウトエラーリクエストの割合、その他のクライアント側エラーリクエストの割合の 4 つのメトリックインジケーターが含まれます。
  - リダイレクトリクエスト回数。正常に終了したリクエストの数とリダイレクトリクエストの数の 2 つのメトリックインジケーターが含まれます。
  - 成功リダイレクト率。正常に終了したリクエストの割合とリダイレクトリクエストの割合の 2 つのメトリックインジケーターが含まれます。

- 測定参照

  メータリング参照グループでは、メータリングインジケーターが 1 時間単位の収集および表示粒度で表示されます。次の図をご参照ください。

  メータリングメトリックチャートは以下が含まれます。

  - 割当サイズ
  - フローの計測
  - 請求リクエスト。Get リクエスト数と Put リクエスト数が含まれます。

  バケットの作成後、新しいデータが現時点から 1 時間収集され、収集されたデータは 30 分以内に表示されます。

- 平均レイテンシ

  このインジケーターグループでは、API モニタリングの平均レイテンシインジケーターが含まれます。 メトリックチャートは次のとおりです。

  - getObject 平均レイテンシ
  - headObject 平均レイテンシ
  - putObject 平均レイテンシ
  - postObject 平均レイテンシ
  - オブジェクト付加の平均レイテンシ
  - パーツアップロードの平均レイテンシ
  - コピーパーツアップロードの平均レイテンシ

  各メトリックチャートには、対応する平均 E2E レイテンシと平均サーバーレイテンシが表示されます。 下図をご参照ください。

- 最大レイテンシ

  このインジケーターグループでは、API モニタリングの最大レイテンシインジケーターが含まれます。 メトリックチャートには以下が含まれます。

  - getObject 最大レイテンシ
  - headObject 最大レイテンシ
  - putObject 最大レイテンシ
  - postObject 最大レイテンシ
  - オブジェクト付加の最大レイテンシ
  - パーツアップロードの最大レイテンシ
  - コピーパーツアップロードの最大レイテンシ

  各メトリックチャートには、対応する最大 E2E レイテンシと最大サーバーレイテンシが表示されます。 次の図をご参照ください。

- 成功リクエストカテゴリ

  このインジケーターグループでは、API モニタリングの正常なリクエスト数のインジケーターが含まれます。 メトリックチャートには以下が含まれます。

  - getObject 成功回数
  - headObject 成功回数
  - putObject 成功回数
  - Post オブジェクト成功回数
  - オブジェクト付加成功回数
  - パーツアップロード成功回数
  - コピーパーツアップロード成功回数
  - オブジェクト削除成功回数
  - deleteObjects 成功回数

  次の図をご参照ください。

アラームルール

[アラームルール] タブページでは、すべてのアラームルールを表示および管理できます。次の図をご参照ください。

[アラームルール] タブページの説明と使用方法については、「アラームサービスユーザーガイド」をご参照ください。

その他のリンク

モニタリングサービスの重要なポイントとユーザーガイドの詳細については、「モニタリング、診断、トラブルシューティング」内の関連する章をご参照ください。

8.3. アラームサービスユーザーガイド

アラーム連絡先およびアラーム連絡先グループの基本的な概念と設定を理解するため、本ユーザーガイドをお読みになる前に次のドキュメントをお読みいただくことを推奨します。

- アラームサービスの概要
- アラーム連絡先の管理

なお、OSS アラームルールは OSS メトリック項目に従って開発されています。 つまり、OSS アラームルールは OSS メトリック項目と同様の範囲で分類されています。 ユーザーレベルとバケットレベルの 2 つのアラーム範囲があります。

アラームルールページ

アラームルールページでは、OSS モニタリングアラームに関連するアラームルールを表示、変更、有効化、無効化、および削除することができます。 各種アラームルールのアラーム履歴も表示可能です。 スクリーンショットの例は次のとおりです。

- 修正するには、目的のアラームルールの横にある **[修正]** をクリックします。
- 削除するには、目的のアラームルールの横にある **[削除]**をクリックします。 複数のアラームルールを選択してから、テーブル下部にある **[削除]** をクリックすることで、一括でアラームルールを削除することもできます。
- アラームルールが *[有効]* ステータスになっている場合は、アラームルールの横にある **[停止]** をクリックして無効化します。 アラームルールが停止されると、このルールのアラーム情報は表示されなくなります。 複数のアラームルールを選択してから、テーブル下部にある **[禁止]** をクリックすることで、一括してアラームルールを無効化することもできます。
- アラームルールが *[禁止]* ステータスになっている場合は、目的のアラームルールの横にある **[有効]** をクリックして有効にします。 アラームルールは例外を検出するため再開され、アラーム情報を送信します。 複数のアラームルールを選択してから、テーブル下部にある **[有効]** をクリックすることで、一括してアラームルールを有効にすることもできます。
- このルールに対応する過去のアラームに関する情報を表示するには、目的のアラームルールの横にある **[アラーム履歴]** をクリックします。

アラーム履歴の概念

- アラーム履歴は、選択したアラームルールの過去のステータス変更を参照します。 通常ステータスからアラームステータスへの切り替え、またはアラームステータスから通常ステータスへの切り替えなどの操作は、ステータス変更と見なされます。 また、チャンネルの無音と呼ばれるステータス変更も使用可能です。
- トリガーされたアラームが 24 時間有効のまま通常のステータスに戻らないでいると、**チャンネルの無音**が発生します。 この場合、新しいアラーム通知は 24 時間送信されません。

- アラームの履歴情報は 1 か月間保持され、この期間内に一度に最大 3 日間のデータを照会することができます。 1 か月以上経過したアラーム履歴情報は自動的に削除され、照会することができなくなります。

アラーム連絡先リストや連絡先の詳細など、アラームに関する詳細を表示するには、目的のアラームの横にある **[表示]** をクリックします。 特定の詳細を表示するスクリーンショットの例は、次のとおりです。

アラームルールの検索

アラームルールページの下部にある制御情報に基づき、検索したアラームルールを迅速に見つけることができます。:

- アラーム範囲ドロップダウンボックス: [すべて] と [バケットレベル] 。 *[すべて]* をクリックすると、すべてのユーザーレベルおよびバケットレベルのアラームルールが表示されます。
- [バケット] ドロップダウンボックス: [アラーム範囲] ドロップダウンボックスで *[バケットレベル]* をクリックした場合、このボックスには現在のユーザーのバケットが一覧表示されます。 このバケットのすべてのアラームルールを表示するバケットをクリックします。
- [モニター対象項目] ドロップダウンボックスには、ユーザーレベルとバケットレベルのメトリック項目を含む、すべての OSS メトリック項目が一覧表示されます。 *[モニター対象項目]* をクリックすると、すべてのモニター対象項目に対するユーザーレベルとバケットレベルのアラームルールが表示されます。
- [アラームステータス]ドロップダウンボックスには、[OK] や [Alarm] などのアラームステータスが一覧表示されます。
- [有効ステータス] のドロップダウンボックスには、[有効] や [禁止] などの有効ステータスが一覧表示されます。
- アラームルールの表示

  [アラームルール] タブをクリックして、すべてのアラームルールを表示します。 また、ドロップダウンボックスで *[バケットレベル]* をクリックしてから、目的のバケットの名前をクリックすることで、そのバケットのアラームルールを確認することができます。 [メトリック項目]、[アラームステータス]、[認証ステータス] などのドロップダウンボックスをクリックすることで、返された情報をフィルタリングすることができます。
- 特定のバケットのアラームルールの表示

  特定のバケットのアラームルールを表示する場合は、[アラーム範囲]ドロップダウンボックスで [バケットレベル] をクリックし、[バケット] ドロップダウンボックスで対象のバケットの名前をクリックします。 **[バケットリスト]** で対象のバケットの **[アラームルール]** をクリックし、[アラーム] タブに移動します。 このタブでは、このバケットのすべてのアラームルールが表示されます。 [メトリック項目] 、[アラームステータス]、ドロップダウンボックス、および [ステータスステータス] ドロップダウンボックスを使用することで、現在の範囲の特定の条件に一致するアラームルールをより適切にフィルタリングすることができます。
- 特定のメトリック項目に関連するアラームルールの表示

  このメトリック項目のすべてのアラームルールを表示するには、[メトリック項目] ドロップダウンボックスで特定のメトリック項目をクリックします。
- 特定のアラームステータスにあるアラームルールの表示

  [アラームステータス]ドロップダウンボックスで、[アラーム] などのアラームステータスをクリックし、現在このステータスにあるすべてのアラームルールを表示します。
- 特定の認証ステータスにあるアラームルールの表示

特定の認証ステータスにあるすべてのアラームルールを表示するには、[認証ステータス] ドロップダウンボックスで、[無効化] などの認証ステータスをクリックします。

アラームルールの追加

[バケットリスト] タブでバケットを指定したら、**[アラームルール設定]** をクリックしてアラームルールを設定します。 または、特定のバケットの [ユーザー概要] タブまたは[モニタリング表示] タブのメトリックチャートでアラームアイコンをクリックし、**[アラームルール一括設定]** ページを開き、複数のアラームルールを設定します。

次の例では、ユーザーレベルでアラームを設定する方法について説明します。 後の方で使用される用語と概念の詳細については、CloudMonitor の「アラームサービスの概要」をご参照ください。

1. **[アラームルール]** のパラメーターを次のように設定します。
   - " Alarm dimension" では、設定するアラームルールのモニタリング範囲を指定します。 範囲がバケットレベルに設定されている場合は、アラームルールを設定する目的のバケットを指定する必要があります。
   - "Monitored items" では、選択したアラーム範囲のすべてのメトリック項目を指定します。 クイック検索ボックスを使用することで、メトリック項目を簡単に見つけることができます。
   - "Statistics interval" は、統計測定間の間隔の長さを指定します。 既定では 5 分に設定されています。
   - " Last times" では、 メトリック項目の値がいくつかの連続した統計サイクルで継続的にしきい値を超える場合、トリガーされるアラームの統計サイクルの数を指定します。
   - "Statistics method" では、このメトリック項目に対して計算される統計インジケーターを指定します。 "Statistics method" は、OSS モニタリングサービスを対象に*モニタリング値*として設定されます。
   - 追加のメトリック項目のアラームルールを設定するには、**[アラームルールの追加]** をクリックします。
   - アラームルールを削除するには、削除するアラームルールの横にある **[削除]** をクリックします。
2. **[次へ]** をクリックすると、**[アラームタイプの設定]** ページが表示されます。

   「アラーム連絡先管理」に従ってアラームコントラクトグループを設定した場合、アラームコントラクトグループはインターフェイスに表示されます。 アラーム連絡先グループを設定していない場合、**[連絡先グループをすばやく作成]** をクリックし、表示されるプロンプトに従ってグループを作成します。
3. **[OK]** をクリックします。

- バケットリストにアラームルールを追加

  [バケットリスト] タブでは、複数のバケットに同じアラームルールを同時に追加できます。 アラームルールを設定するバケットをクリックし、[カスタムモニタリングアラームルールの設定] をクリックして、前の**[アラームルールの追加]**で説明したアラームルール設定ページに移動します。

  > 説明　バッチ設定中は、アラーム範囲はバケットレベルであり、メトリック項目はバケットレベルのメトリック項目である必要があります。

- メトリックチャートにアラームルールを追加

[ユーザー概要] タブまたは [モニタリングチャート] タブで、目的のバケットについて、メトリックチャートの右上の

をクリックして、このメトリックチャートに関連付けられているメトリック項目のアラームルールを設定します。

> 説明　メトリックチャートのアラームアイコンをクリックした場合、アラームルールページに表示されるアラーム範囲は事前に決定されており、メトリックチャートに対応するメトリック項目に対してのみアラームルールを設定します。

考慮事項

現在、アラームルールはバケットへの事前の関連付けを必要とせずに作成できます。 バケットを削除しても、関連付けられているアラームルールは削除されません。 バケットを削除する前に、まず対応するアラームルールを削除することを推奨します。

8.4. 測定項目リファレンス

本ページでは、OSS モニタリングサービスの測定データにアクセスするため、API または CloudMonitor SDK で使用するパラメーターについて説明します。

プロジェクト

OSS モニタリングサービスの測定データは、同一のプロジェクト名 acs_oss を使用します。

Java SDK により記述されたサンプルコード:

```
QueryMetricRequest request = new QueryMetricRequest();
Request. setproject ("acs_oss ");
```

StartTime と EndTime

CloudMonitor の時間パラメーター値の範囲は、[StartTime、EndTime] の形式です。 StartTime に起因するデータは収集されませんが、EndTime に起因するデータにはアクセスされます。

CloudMonitor 保存ポリシーでは、データが 31 日間保存されるように指定しています。つまり、StartTime と EndTime の間隔は31日を超えることはできず、31 日の収集期間外のデータにはアクセスすることはできません。

その他の時間パラメーターの詳細については、「CloudMonitor API リファレンス」をご参照ください。

Java SDK により記述されたサンプルコード:

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

ディメンション

OSS 測定項目は、適用シナリオに基づき、ユーザーレベルまたはバケットレベルに分類されます。 これらの異なるレベルでの測定データのアクセスに関して、ディメンションの値は異なります。

- ユーザーレベルの測定データへアクセスする場合、ディメンションを設定する必要はありません。

- ディメンションアクセスをバケットレベルの測定データに設定するには、以下をご参照ください。

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

your_bucket_name は、アクセス対象のバケットの名前を示します。

注記: ディメンションは JSON 文字列であり、OSS 測定インジケーターに対するキーと値のペアを 1 つだけ含みます。

Java SDK により記述されたサンプルコード:

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

期間

メータリングインジケーターを除き、すべての OSS 測定インジケーターの既定の集計粒度は 60 秒です。メータリングインジケーターの既定の集計粒度は 3,600 秒です。

Java SDK により記述されたサンプルコード:

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

測定項目

「モニタリング指標リファレンス」では、次の測定項目について説明しています。

測定項目	測定項目名	単位	レベル
Useravailability	ユーザーレベルの可用性	%	ユーザーレベル
UserRequestValidRate	ユーザーレベルの有効リクエスト率	%	ユーザーレベル
UserTotalRequestCount	ユーザーレベルのリクエスト数	回	ユーザーレベル
UserValidRequestCount	ユーザーレベルの有効リクエスト数	回	ユーザーレベル
UserInternetSend	ユーザーレベルのインターネットアウトバウンドトラフィック	バイト	ユーザーレベル
UserInternetRecv	ユーザーレベルのインターネットインバウンドトラフィック	バイト	ユーザーレベル
UserIntranetSend	ユーザーレベルのイントラネットアウトバウンドトラフィック	バイト	ユーザーレベル
UserIntranetRecv	ユーザーレベルのイントラネットインバウンドトラフィック	バイト	ユーザーレベル

測定項目	測定項目名	単位	レベル
UserCdnSend	ユーザーレベルの CDN アウトバウンドトラフィック	バイト	ユーザーレベル
UserCdnRecv	ユーザーレベルの CDN インバウンドトラフィック	バイト	ユーザーレベル
UserSyncSend	ユーザーレベルのリージョン間レプリケーションのアウトバウンドトラフィック	バイト	ユーザーレベル
UserSyncRecv	ユーザーレベルのリージョン間レプリケーションのインバウンドトラフィック	バイト	ユーザーレベル
UserServerErrorCount	ユーザーレベルのサーバーサイトエラーリクエスト数	回	ユーザーレベル
UserServerErrorRate	ユーザーレベルのサーバーサイトエラーリクエスト率	%	ユーザーレベル
UserNetworkErrorCount	ユーザーレベルのネットワークサイトエラーリクエスト数	回	ユーザーレベル
UserNetworkErrorRate	ユーザーレベルのネットワークサイトエラーリクエスト率	%	ユーザーレベル
UserAuthorizationErrorCount	ユーザーレベルのクライアントサイト権限付与エラーリクエスト数	回	ユーザーレベル
UserAuthorizationErrorRate	ユーザーレベルのクライアントサイト権限付与エラーリクエスト率	%	ユーザーレベル
UserResourceNotFoundErrorCount	ユーザーレベルのリソース不明クライアントサイトエラーリクエスト数	回	ユーザーレベル
UserResourceNotFoundErrorRate	ユーザーレベルのリソース不明クライアントサイトエラーリクエスト率	%	ユーザーレベル
UserClientTimeoutErrorCount	ユーザーレベルのクライアントサイトタイムアウトエラーリクエスト数	回	ユーザーレベル

測定項目	測定項目名	単位	レベル
UserClientOtherErrorRate	ユーザーレベルのクライアントサイトタイムアウトエラーリクエスト率	%	ユーザーレベル
UserClientOtherErrorCount	他のユーザーレベルのクライアントサイトエラーリクエスト数	回	ユーザーレベル
UserClientOtherErrorRate	他のユーザーレベルのクライアントサイトエラーリクエスト率	%	ユーザーレベル
UserSuccessCount	成功したユーザーレベルのリクエスト数	回	ユーザーレベル
UserSuccessRate	成功したユーザーレベルのリクエスト率	%	ユーザーレベル
UserRedirectCount	ユーザーレベルのリダイレクトリクエスト数	回	ユーザーレベル
UserRedirectRate	ユーザーレベルのリダイレクトリクエスト率	%	ユーザーレベル
Availability	可用性	%	バケットレベル
RequestValidRate	有効リクエスト率	%	バケットレベル
TotalRequestCount	リクエスト数	回	バケットレベル
ValidRequestCount	有効リクエスト数	回	バケットレベル
InternetSend	インターネットアウトバウンドトラフィック	バイト	バケットレベル
InternetRecv	インターネットインバウンドトラフィック	バイト	バケットレベル
IntranetSend	イントラネットアウトバウンドトラフィック	バイト	バケットレベル
IntranetRecv	イントラネットインバウンドトラフィック	バイト	バケットレベル
CdnSend	CDN アウトバウンドトラフィック	バイト	バケットレベル
CdnRecv	CDN インバウンドトラフィック	バイト	バケットレベル

測定項目	測定項目名	単位	レベル
SyncSend	リージョン間レプリケーションのアウトバウンドトラフィック	バイト	バケットレベル
SyncRecv	リージョン間レプリケーションのインバウンドトラフィック	バイト	バケットレベル
ServerErrorCount	サーバーサイトエラーリクエスト数	回	バケットレベル
ServerErrorRate	サーバーサイトエラーリクエスト率	%	バケットレベル
NetworkErrorCount	ネットワークサイトエラーリクエスト数	回	バケットレベル
NetworkErrorRate	ネットワークサイトエラーリクエスト率	%	バケットレベル
AuthorizationErrorCount	クライアントサイト権限付与エラーリクエスト数	回	バケットレベル
AuthorizationErrorRate	クライアントサイト権限付与エラーリクエスト率	%	バケットレベル
ResourceNotFoundErrorCount	リソース不明クライアントサイトエラーリクエスト数	回	バケットレベル
ResourceNotFoundErrorRate	リソース不明クライアントサイトエラーリクエスト率	%	バケットレベル
ClientTimeoutErrorCount	クライアントサイトタイムアウトエラーリクエスト数	回	バケットレベル
ClientOtherErrorRate	クライアントサイトタイムアウトエラーリクエスト率	%	バケットレベル
ClientOtherErrorCount	他のクライアントサイトエラーリクエスト数	回	バケットレベル
ClientOtherErrorRate	他のクライアントサイトエラーリクエスト率	%	バケットレベル
SuccessCount	成功リクエスト数	回	バケットレベル
SuccessRate	成功リクエスト率	%	バケットレベル

測定項目	測定項目名	単位	レベル
RedirectCount	リダイレクトリクエスト数	回	バケットレベル
RedirectRate	リダイレクトリクエスト率	%	バケットレベル
GetObjectE2eLatency	GetObject リクエストの平均 E2E レイテンシ	ミリ秒	バケットレベル
GetObjectServerLatency	GetObject リクエストの平均サーバーレイテンシ	ミリ秒	バケットレベル
MaxGetObjectE2eLatency	GetObject リクエストの最大 E2E レイテンシ	ミリ秒	バケットレベル
MaxGetObjectServerLatency	GetObject リクエストの最大サーバーレイテンシ	ミリ秒	バケットレベル
HeadObjectE2eLatency	HeadObject リクエストの平均 E2E レイテンシ	ミリ秒	バケットレベル
HeadObjectServerLatency	HeadObject リクエストの平均サーバーレイテンシ	ミリ秒	バケットレベル
MaxHeadObjectE2eLatency	HeadObject リクエストの最大 E2E レイテンシ	ミリ秒	バケットレベル
MaxHeadObjectServerLatency	HeadObject リクエストの最大サーバーレイテンシ	ミリ秒	バケットレベル
PutObjectE2eLatency	PutObject リクエストの平均 E2E レイテンシ	ミリ秒	バケットレベル
PutObjectServerLatency	PutObject リクエストの平均サーバーレイテンシ	ミリ秒	バケットレベル
MaxPutObjectE2eLatency	PutObject リクエストの最大 E2E レイテンシ	ミリ秒	バケットレベル
MaxPutObjectServerLatency	PutObject リクエストの最大サーバーレイテンシ	ミリ秒	バケットレベル
PostObjectE2eLatency	PostObject リクエストの平均 E2E レイテンシ	ミリ秒	バケットレベル
PostObjectServerLatency	PostObject リクエストの平均サーバーレイテンシ	ミリ秒	バケットレベル
MaxPostObjectE2eLatency	PostObject リクエストの最大 E2E レイテンシ	ミリ秒	バケットレベル

測定項目	測定項目名	単位	レベル
MaxPostObjectServerLatency	PostObject リクエストの最大サーバーレイテンシ	ミリ秒	バケットレベル
AppendObjectE2eLatency	AppendObject リクエストの平均 E2E レイテンシ	ミリ秒	バケットレベル
AppendObjectServerLatency	AppendObject リクエストの平均サーバーレイテンシ	ミリ秒	バケットレベル
MaxAppendObjectE2eLatency	AppendObject リクエストの最大 E2E レイテンシ	ミリ秒	バケットレベル
MaxAppendObjectServerLatency	AppendObject リクエストの最大サーバーレイテンシ	ミリ秒	バケットレベル
UploadPartE2eLatency	UploadPart リクエストの平均 E2E レイテンシ	ミリ秒	バケットレベル
UploadPartServerLatency	UploadPart リクエストの平均サーバーレイテンシ	ミリ秒	バケットレベル
MaxUploadPartE2eLatency	UploadPart リクエストの最大 E2E レイテンシ	ミリ秒	バケットレベル
MaxUploadPartServerLatency	UploadPart リクエストの最大サーバーレイテンシ	ミリ秒	バケットレベル
UploadPartCopyE2eLatency	UploadPartCopy リクエストの平均 E2E レイテンシ	ミリ秒	バケットレベル
UploadPartCopyServerLatency	UploadPartCopy リクエストの平均サーバーレイテンシ	ミリ秒	バケットレベル
MaxUploadPartCopyE2eLatency	UploadPartCopy リクエストの最大 E2E レイテンシ	ミリ秒	バケットレベル
MaxUploadPartCopyServerLatency	UploadPartCopy リクエストの最大サーバーレイテンシ	ミリ秒	バケットレベル
GetObjectCount	成功 GetObject リクエスト数	回	バケットレベル

測定項目	測定項目名	単位	レベル
HeadObjectCount	成功 HeadObject リクエスト数	回	バケットレベル
PutObjectCount	成功 PutObject リクエスト数	回	バケットレベル
PostObjectCount	成功 PostObject リクエスト数	回	バケットレベル
AppendObjectCount	成功 AppendObject リクエスト数	回	バケットレベル
UploadPartCount	成功 UploadPart リクエスト数	回	バケットレベル
UploadPartCopyCount	成功 UploadPartCopy リクエスト数	回	バケットレベル
DeleteObjectCount	成功 DeleteObject リクエスト数	回	バケットレベル
DeleteObjectsCount	成功 DeleteObjects リクエスト数	回	バケットレベル

次のテーブルでは、集計粒度が 3,600 秒であるメータリングインジケーターの測定項目の一覧を示します。

測定項目	測定項目名	単位	レベル
MeteringStorageUtilization	ストレージのサイズ	バイト	返される測定データは、ディメンションが設定されている場合、バケットレベルに属します。 ディメンションが設定されていない場合は、返される測定データはユーザーレベルに属します。
MeteringGetRequest	Get リクエスト数	回	返される測定データは、ディメンションが設定されている場合、バケットレベルに属します。 ディメンションが設定されていない場合は、返される測定データはユーザーレベルに属します。

測定項目	測定項目名	単位	レベル
MeteringPutRequest	Put リクエスト数	回	返される測定データは、ディメンションが設定されている場合、バケットレベルに属します。ディメンションが設定されていない場合は、ユーザーレベルに属します。
Meteringinternettx	インターネットアウトバウンドトラフィックの量	バイト	返される測定データは、ディメンションが設定されている場合、バケットレベルに属します。ディメンションが設定されていない場合は、ユーザーレベルに属します。
MeteringCdnTX	CDN アウトバウンドトラフィックの量	バイト	返される測定データは、ディメンションが設定されている場合、バケットレベルに属します。ディメンションが設定されていない場合は、ユーザーレベルに属します。
MeteringSyncRX	リージョン間レプリケーションのインバウンドトラフィックの量	バイト	返される測定データは、ディメンションが設定されている場合、バケットレベルに属します。ディメンションが設定されていない場合は、ユーザーレベルに属します。

Java SDK により記述されたサンプルコード:

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

8.5. モニタリングインジケーターのリファレンス

OSS インジケーターでは、適用シナリオに基づき、ユーザーレベルまたはバケットレベルでモニタリングすることができます。

一般的な時系列メトリックインジケーターに加え、システムは既存のメトリックインジケーターに関する統計を分析および収集します。そのため、メトリックデータの確認や課金ポリシーの照合を簡単に行うことができます。 リクエストステータスの分布やその月のメータリング統計など、指定された期間の統計インジケーターが提供されます。 このリファレンスガイドでは、インジケーターについて詳しく説明します。

メータリングインジケーターおよび統計インジケーターを除き、すべてのインジケーターは分レベル (1 分ごと) で収集されます。 メータリングインジケーターは、時間レベル (1 時間ごと) で収集されます 。

ユーザーレベルのインジケーター

ユーザーレベルのインジケーターは、ユーザーのアカウントレベルから使用される OSS システムの全体的な状況をモニタリングするインジケーター情報を参照します。つまり、ユーザーのアカウントにおけるすべてのバケット関連モニタリングデータの概要です。 ユーザーレベルのインジケーターは、3 つのモニタリングインジケーターの詳細から構成されています。:当月のメータリング統計、サービスモニタリングの概要、およびリクエストステータスの詳細で構成されています。

サービスモニタリングの概要

「サービスモニタリングの概要」内のインジケーターは、基本的なサービスインジケーターです。 サービスモニタリング概要のインジケーターの詳細は次のとおりです。

インジケーター	単位	説明
Availability	%	ストレージサービスを使用するシステムの可用性を示すインジケーター。 これは次の式によって得られます。： 1 - すべてのリクエストの中でのサーバー終了エラーリクエストの割合 (サーバー終了エラーのリターンコードは 5xx です) 。
Valid requests rate	%	すべてのリクエストにおける、有効リクエストの割合。 有効リクエストの詳細については、次の説明をご参照ください。 説明
Requests	回	OSS サーバーで受信および処理されたリクエストの合計数
Valid requests	回	リターンコードが 2xx または 3xx のリクエストの合計数
Internet outbound traffic	バイト	ダウンストリームインターネットトラフィック
Internet inbound traffic	バイト	アップストリームインターネットトラフィック
Intranet outbound traffic	バイト	サービスシステムのダウンストリームイントラネットトラフィック
Intranet inbound traffic	バイト	サービスシステムのアップストリームイントラネットトラフィック
CDN outbound traffic	バイト	CDN アクセラレーションサービスがアクティブ化されているときのダウンストリーム CDN トラフィック。つまり、発信元検索トラフィック
CDN inbound traffic	バイト	CDN アクセラレーションサービスがアクティブになっているときのアップストリーム CDN トラフィック

インジケーター	単位	説明
Outbound traffic of cross-region replication	バイト	クロスリージョンレプリケーション機能がアクティブになっているときに、データレプリケーションプロセスで生成されたダウンストリームトラフィック
Inbound traffic of cross-region replication	バイト	クロスリージョンレプリケーション機能がアクティブになっているときに、データレプリケーションプロセスで生成されたアップストリームトラフィック

また、上記の特定のモニタリングインジケーターに加え、一定期間にわたるリクエストステータスの分布についての統計も提供されます。 統計情報は主に、返されたステータスコード、もしくは OSS エラーコード (モニタリング期間内のリクエストの合計数と割合) に基づいています。

リクエストステータスの詳細

リクエストステータスの詳細インジケーターでは、各種リクエストに関連付けられたリターンステータスコード、または OSS エラーコードに基づき、情報のモニタリングがリクエストされます。 リクエストステータスの詳細インジケーターの詳細は次のとおりです。

インジケーター	単位	説明
Server-site error requests	回	リターンコード 5xx で示されるシステムレベルのエラーを伴うリクエストの合計数
Server-site error requests rate	%	すべてのリクエストにおける、サーバー側エラーを伴ったリクエストの割合
Network error requests	回	HTTP ステータスコードが 499 のリクエストの合計数
Network error requests rate	%	すべてのリクエストにおける、ネットワークエラーを伴ったリクエストの割合
Client-end authorization error requests	回	リターンコード 403 を伴ったリクエストの合計数
Client-end authorization error requests rate	%	すべてのリクエストにおける、クライアント側の権限付与エラーを伴ったリクエストの割合
Client-end error requests indicating resource not	回	リターンコード 404 を伴ったリクエストの合計数
リソースがないことを示すクライアント側エラーリクエスト率	%	すべてのリクエストにおける、リソースが見つからないことを示すクライアント側エラーを伴ったリクエストの割合

インジケーター	単位	説明
Client-end time-out error requests	回	リターンステータスコードが 408、または戻り OSS エラーコードが RequestTimeout のリクエストの合計数
Client-end time-out error requests rate	%	すべてのリクエストにおける、クライアント側のタイムアウトエラーを伴ったリクエストの割合
Other client-end error requests	回	リターンコード 4xx が示されている、他のクライアント側エラーを伴ったリクエストの合計数
Other client-end error requests rate	%	すべてのリクエストにおける、その他のクライアント側のエラーが発生したリクエストの割合
Successful requests	回	リターンコードが 2xx のリクエストの合計数
Successful requests rate	%	すべてのリクエストにおける、成功したリクエストの割合
Redirect requests	回	リターンコードが 3xx のリクエストの合計数
Redirect requests rate	%	すべてのリクエストにおける、リダイレクトリクエストの割合

当月のメータリング統計

当月のメータリング統計は、その月の最初の日の 00:00 から同月のメータリングの終了時刻までの間、データが収集されます。

現在使用可能なメータリングインジケーターの詳細は次のとおりです。

インジケーター	単位	説明
Storage size	バイト	メータリング統計収集期限までに、指定されたユーザーのすべてのバケットにより占有されている合計ストレージサイズ
Internet outbound traffic	バイト	当月の最初の日の 00:00 からメータリング統計収集期限までの、ユーザーのインターネットアウトバウンドトラフィック合計バイト
Put requests	回	当月の最初の日の 00:00 からメータリング統計収集期限までの、ユーザーの Put リクエスト合計数

インジケーター	単位	説明
Get requests	回	当月の最初の日の 00:00 からメータリング統計収集期限までの、ユーザーの Get リクエスト合計数

バケットレベルのインジケーター

バケットレベルのインジケーターは、特定のバケットの OSS 操作をモニタリングするために使用され、ビジネスシナリオに適用されます。 バケットレベルのインジケーターでは、当月のメータリング統計と、サービスモニタリングの概要やリクエストステータスの詳細 (アカウントレベルでモニターされる) などの基本的なサービスインジケータ項目に加えて、メータリングインジケーター、メータリング参照、レイテンシ、成功したリクエスト操作カテゴリなどのパフォーマンスインジケーターが含まれます。

サービスモニタリングの概要

サービスモニタリングの概要インジケーターでは、ユーザーレベルの説明と同様に、基本的なインジケーターですが、バケットレベルで表示されるメトリックデータが使用されます。

リクエストステータスの詳細

リクエストステータスの詳細インジケーターでは、ユーザーレベルの説明と同様に、バケットレベルで表示されるメトリックデータが使用されます。

当月のメータリング統計

統計方法は、ユーザーレベルでの当月のメータリング統計で列挙されているものと同様ですが、バケットレベルのインジケーターでは、バケットレベルでリソースの使用状況の統計を収集します。 バケットレベルでの当月のメータリング統計の詳細は次のとおりです。

インジケーター	単位	説明
Storage size	バイト	メータリング統計収集期限の前に、指定されたバケットが占有するストレージサイズ
Internet outbound traffic	バイト	指定されたバケットの当月の最初の日の 00:00 からメータリング統計収集期限までの、合計インターネットアウトバウンドトラフィック
Put requests	回	当月の最初の日の 00:00 からメータリング統計収集期限までの、ユーザーの Put リクエスト合計数
Get requests	回	当月の最初の日の 00:00 からメータリング統計収集期限までの、ユーザーの Get リクエスト合計数

メータリングインジケーター

メータリングインジケーターは時系列でモニタリングされ、時間レベル (1 時間ごと) で収集、集計されます。 メータリングインジケーターの詳細は以下のとおりです。

インジケーター	単位	説明
Storage size	バイト	特定のバケットが 1 時間に使用するストレージの平均サイズ
Internet outbound traffic	バイト	指定されたバケットが 1 時間に使用するインターネットアウトバウンドトラフィックの合計
Put requests	回	指定されたバケットが 1 時間に行った Put リクエストの合計数
Get requests	回	指定されたバケットが 1 時間に行った Get リクエストの合計数

レイテンシ

レイテンシリクエストでは、システムパフォーマンスを直接示し、平均レイテンシと最大レイテンシの 2 つのインジケーターを使用してモニタリングします。 各 インジケーターは、分レベル (1 分ごと) で収集、集計されます。 また、各インジケーターは、OSS API リクエスト操作タイプに基づき分類されることで、各種操作に応答するシステムのパフォーマンスをより正確に示します。 現在、バケット関連操作 (メタ操作を除く) のデータ操作を含む API のみがモニターされます。

また、パフォーマンスのホットスポットの分析や、環境問題の分析を容易にするため、 E2E リンク と サーバーリンクから、レイテンシモニタリングインジケーターが収集されます。

- E2E レイテンシとは、成功したリクエストを OSS に送信する際の E2E レイテンシを表します。レイテンシには、OSS リクエストの読み取りに必要な処理時間、応答の送信処理に必要な時間、および応答確認メッセージの受信処理に必要な時間を含みます。
- サーバーのレイテンシは、E2E レイテンシを含むネットワーク遅延を除く、成功したリクエストを処理する OSS のレイテンシです。

パフォーマンスインジケーターは、成功したリクエストをモニターするために使用されることに注意してください (リターンステータスコード 2xx)。

次のテーブルに、具体的なメトリックインジケーター項目を示します。

インジケーター	単位	説明
Average E2E latency of GetObject requests	ミリ秒	リクエスト API が GetObject である成功したリクエストの平均 E2E レイテンシ
Average server latency of GetObject requests	ミリ秒	リクエスト API が GetObject である成功したリクエストの平均サーバーレイテンシ API is GetObject
Maximum E2E latency of GetObject requests	ミリ秒	リクエスト API が GetObject である成功したリクエストの最大 E2E レイテンシ
Maximum server latency of GetObject requests	ミリ秒	リクエスト API が GetObject である成功したリクエストの最大サーバーレイテンシ

インジケーター	単位	説明
Average E2E latency of HeadObject requests	ミリ秒	リクエスト API が HeadObject である成功したリクエストの平均 E2E レイテンシ
Average server latency of HeadObject requests	ミリ秒	リクエスト API が HeadObject である成功したリクエストの平均サーバーレイテンシ
Maximum E2E latency of HeadObject requests	ミリ秒	リクエスト API が HeadObject である成功したリクエストの最大 E2E レイテンシ
Maximum server latency of HeadObject requests	ミリ秒	リクエスト API が HeadObject である成功したリクエストの最大サーバーレイテンシ
Average E2E latency of PutObject requests	ミリ秒	リクエスト API が PutObject である成功したリクエストの平均 E2E レイテンシ
Average server latency of PutObject requests	ミリ秒	リクエスト API が PutObject である成功したリクエストの平均サーバーレイテンシ
Maximum E2E latency of PutObject requests	ミリ秒	リクエスト API が PutObject である成功したリクエストの最大 E2E レイテンシ
Maximum server latency of PutObject requests	ミリ秒	リクエスト API が PutObject である成功したリクエストの最大サーバーレイテンシ
Average E2E latency of PostObject requests	ミリ秒	リクエスト API が PostObject である成功したリクエストの平均 E2E レイテンシ
Average server latency of PostObject requests	ミリ秒	リクエスト API が PostObject である成功したリクエストの平均サーバーレイテンシ
Maximum E2E latency of PostObject requests	ミリ秒	リクエスト API が PostObject である成功したリクエストの最大 E2E レイテンシ
Maximum server latency of PostObject requests	ミリ秒	リクエスト API が PostObject である成功したリクエストの最大サーバーレイテンシ
Average E2E latency of AppendObject requests	ミリ秒	リクエスト API が AppendObject である成功したリクエストの平均 E2E レイテンシ

インジケーター	単位	説明
Average server latency of AppendObject requests	ミリ秒	リクエスト API が AppendObject である成功したリクエストの平均サーバーレイテンシ
Maximum E2E latency of AppendObject requests	ミリ秒	リクエスト API が AppendObject である成功したリクエストの最大 E2E レイテンシ
Maximum server latency of AppendObject requests	ミリ秒	リクエスト API が AppendObject である成功したリクエストの最大サーバーレイテンシ
Average E2E latency of UploadPart requests	ミリ秒	リクエスト API が UploadPart の成功したリクエストの平均 E2E レイテンシ
Average server latency of UploadPart requests	ミリ秒	リクエスト API が UploadPart である成功したリクエストの平均サーバーレイテンシ
Maximum E2E latency of UploadPart requests	ミリ秒	リクエスト API が UploadPart である成功したリクエストの最大 E2E レイテンシ
Maximum server latency of UploadPart requests	ミリ秒	リクエスト API が UploadPart である成功したリクエストの最大サーバーレイテンシ
Average E2E latency of UploadPartCopy requests	ミリ秒	リクエスト API が UploadPartCopy である成功したリクエストの平均 E2E レイテンシ
Average server latency of UploadPartCopy requests	ミリ秒	リクエスト API が UploadPartCopy である成功したリクエストの平均サーバーレイテンシ
Maximum E2E latency of UploadPartCopy requests	ミリ秒	リクエスト API が UploadPartCopy である成功したリクエストの最大 E2E レイテンシ
Maximum server latency of UploadPartCopy requests	ミリ秒	リクエスト API が UploadPartCopy である成功したリクエストの最大サーバーレイテンシ

成功したリクエスト操作の分類項目

成功したリクエストのモニタリングでは、レイテンシのモニタリングと連携し、アクセスリクエストをある程度処理するシステムの能力を示します。 同様に、現在はバケット関連操作でのデータ操作を含む API だけがモニターされます。 以下に具体的なインジケーター項目を示します。

インジケータ	単位	説明
Successful GetObject requests	回	リクエスト API が GetObject である成功したリクエストの数
Successful HeadObject requests	回	リクエスト API が HeadObject である成功したリクエストの数
Successful PutObject requests	回	リクエスト API が PutObject である成功したリクエストの数
Successful PostObject requests	回	リクエスト API が PostObject である成功したリクエストの数
Successful AppendObject requests	回	リクエスト API が AppendObject である成功したリクエストの数
Successful UploadPart requests	回	リクエスト API が UploadPart である成功したリクエストの数
Successful UploadPartCopy requests	回	リクエスト API が UploadPartCopy である成功したリクエストの数
Successful DeleteObject requests	回	リクエスト API が DeleteObject である成功したリクエストの数
Successful DeleteObjects requests	回	リクエスト API が DeleteObjects である成功したリクエストの数

8.6. サービスモニタリング、診断、トラブルシューティング

従来のアプリケーションと比較し、インフラストラクチャ構築と O&M クラウドアプリケーションのユーザーのコストは削減されているにもかかわらず、クラウドアプリケーションは複雑なモニタリング、診断、トラブルシューティングを行います。 OSS ストレージサービスでは、各種モニタリングおよびログ情報を提供しているため、プログラムの動作全体を把握し、問題を迅速に発見および特定するのに役立ちます。

概要

本ページでは、OSS モニタリングサービス、ログ、およびその他のサードパーティーのツールを使用して、OSS の問題のモニタリング、診断、およびトラブルシューティングを行う方法について説明します。

- OSS の実行ステータスとパフォーマンスをリアルタイムでモニターし、迅速なアラーム通知を提供します。
- 問題の特定に役立つ効果的な方法とツールを提供します。
- 一般的な OSS 関連の問題を迅速に解決するための方法を提供します。

本ページは次のように構成されています。

- OSS リアルタイムモニタリング: OSS モニタリングサービスを使用して、OSS の実行ステータスとパフォーマンスを継続的にモニターする方法について説明します。

- トラッキングと診断: OSS モニタリングサービスおよびログ機能を使用して問題を診断する方法、およびトラッキングおよび診断を行うにあたり、関連情報をログファイルに関連付ける方法について説明します。
- トラブルシューティング: 一般的な問題と対応するトラブルシューティング方法について説明します。
-

OSS モニタリング

運用状況の概略

- 有効なリクエストの可用性と割合

  これは、システムの安定性とユーザーがシステムを正しく使用できるかどうかを示す重要なインジケーターです。 100 %より低い値は、一部のリクエストが失敗したことを示します。

  可用性は、負荷分散のためのパーティション移行などのシステム最適化要因により、一時的に 100% を下回る可能性があります。 このような場合、OSS SDK は、このタイプの断続的な障害を処理するための関連する再試行メカニズムを提供し、サービスを終了しないでおくことができます。

  また、有効なリクエストの割合が 100 %を下回る場合、使用状況に基づいて問題を分析する必要があります。 リクエスト分散統計またはリクエスト状況詳細を使用することで、リクエストエラーの実際のタイプを判別します。 「トラッキングと診断」を使用することにより、問題の原因を特定しトラブルシューティングを実行することができます。 一部のビジネスシナリオでは、有効なリクエスト率は 100 %を下回ることが予想されます。 たとえば、まずオブジェクトが存在することを確認してから、そのオブジェクトの存在に基づいて特定の操作を実行する必要があります。 オブジェクトが存在しない場合、その存在を検査する読み取りリクエストは、404 エラーコード (リソースが存在しないエラー) を返します。 これにより、必然的に有効リクエスト率は 100 %未満になります。

  高いシステム可用性を必要とする企業では、予想されるしきい値をインジケーターが下回ったとき、トリガーされるアラームルールを設定することができます。

- リクエスト合計数と有効リクエスト数

  このインジケーターは、トラフィックの合計量の分析観点からシステム運用状況を示します。 有効なリクエスト数がリクエストの合計数と等しくない場合、一部のリクエストが失敗したことを示します。

  特にリクエスト数が急激に増減する場合、リクエスト合計数と有効リクエスト数の変動を見ることができます。 そのような場合、フォローアップ操作が必要です。 アラームルールを設定することで、プロンプト通知を確実に受け取れるようにすることができます。 定期的なビジネスの場合、定期的なアラームルールを設定します (定期的なアラームはすぐに使用可能になります)。 詳しくは、「アラームサービスユーザーガイド」をご参照ください。

- ステータス分散統計情報のリクエスト

  可用性または有効リクエスト率が 100 %を下回る場合 (または有効リクエスト数が全体のリクエストの合計数と等しくない場合) 、リクエスト状況の分散統計を調べて、リクエストのエラータイプを迅速に判別することできます。 このメトリックインジケーターの詳細については、「モニタリングインジケーターリファレンス」をご参照ください。

リクエストステータスの詳細なモニタリング

リクエスト状況の詳細では、リクエスト状況の分散統計に基づき、リクエストのモニタリングステータスについての詳細を提供します。 特定のタイプのリクエストを、より詳細にモニターすることができます。

パフォーマンスモニタリング

モニタリングサービスでは、パフォーマンスのモニタリングのインジケーターとして使用される以下のメトリック項目を提供します。

- 平均レイテンシ: E2E 平均レイテンシとサーバー平均レイテンシ
- 最大レイテンシ: E2E 最大レイテンシとサーバー最大レイテンシ
- 成功したリクエストの分類項目
- トラフィック

  前述のメトリック項目 ("トラフィック" を除く) では、API 操作タイプに基づいて分類されたモニタリングを実装しています。

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

  レイテンシインジケーターは、API 操作タイプがリクエストを処理するために必要な平均時間または最大時間を示します。 E2E のレイテンシは、端末相互間のレイテンシのインジケーターです。 リクエストを処理するために必要な時間に加え、リクエストの読み取りに必要な時間、応答の送信に必要な時間、およびネットワーク転送による遅延時間が含まれます。 サーバーのレイテンシには、サーバー側のリクエストを処理するために必要な時間だけが含まれ、クライアント側のネットワークのレイテンシは含まれません。 したがって、E2E レイテンシが突如増加したものの、サーバーのレイテンシに大きな変化がない場合、OSS システムの障害が原因ではなく、ネットワークの不安定性によってパフォーマンスが低下していると判断することができます。

  前述の API に加えて、"成功したリクエスト操作分類項目" では、次の 2 つの API 操作タイプのリクエスト数もモニターされます。

  - DeleteObject
  - Deleteobjects

  トラフィックインジケーターは、ユーザーまたは特定のバケットの全体的な状況をモニターするために使用されます。 インターネット、イントラネット、CDN オリジン検索、ドメイン間レプリケーションなどのシナリオにおける、ネットワークリソースの使用状況をモニターします。

  パフォーマンスタイプのインジケーターでは、平均レイテンシが急激に増加したり、通常のリクエストレイテンシベースラインを長期間にわたって維持したりするなど、突然の変化や異常な変化に注視する必要があります。 パフォーマンスインジケーターに対応するアラームルールを設定することにより、インジケーターがしきい値を下回った場合、またはしきい値を超えた場合に、関係している担当者にすぐに通知されるようにすることができます。 定期的なピークと谷があるビジネスでは、週ごと、曜日ごと、時間ごとの比較に定期的なアラームルールを設定することができます (定期的なアラームはすぐに使用できます)。

課金情報のモニタリング

プレス時は、OSS モニタリングサービスは、ストレージスペース、アウトバウンドインターネットトラフィック、Put リクエスト、および Get リクエスト (ドメイン間レプリケーションアウトバウンドトラフィックと CDN アウトバウンドトラフィックを除く) のみをモニタリングします。 課金情報データのアラーム設定、または API 読み取り操作はサポートされていません。

OSS モニタリングサービスでは、1 時間ごとにバケットレベルの課金情報モニタリングデータを収集します。 特定のバケットのモニタリングビューでは、継続的なモニタリング傾向のグラフを確認することができます。 モニタリングビューを使用することで、ビジネスの OSS リソース使用量の傾向を分析し、将来のコストを見積もることができます。 次の図をご参照ください。

また、OSS モニタリングサービスは、毎月消費されるユーザーおよびバケットレベルのリソースの量に関する統計も提供します。 たとえば、月の最初の日から開始されたアカウントまたはバケットによって消費された OSS リソースの合計量があります。 これらの統計は毎時更新されます。 これにより、次の図に示すように、当月のリソース使用量と計算コストをリアルタイムで把握することができます。

> 説明　モニタリングサービスでは、提供された課金情報データは最大限プッシュされますが、実際の請求額とは若干の差異が生じる可能性があります。 課金情報センターのデータが、実際の請求額として使用されることをご了承ください。

トラッキングと診断

問題の診断

- パフォーマンスの診断

  アプリケーションパフォーマンスの決定には、多くの主体的要因が関係しています。 特定のビジネスシナリオでのビジネスニーズの満足度をベースラインとして使用して、パフォーマンスの問題が発生するかどうかを判断する必要があります。また、クライアントがリクエストを開始すると、パフォーマンス上の問題を引き起こす可能性のある要因が、リクエストチェーンのどこからでも発生する可能性があります。たとえば、OSS の過負荷、クライアントの TCP 設定の問題、または基本的なネットワークアーキテクチャのトラフィックのボトルネックにより、問題が発生する可能性があります。

  したがって、パフォーマンスの問題を診断するときは、最初に合理的なベースラインを設定する必要があります。 次に、モニタリングサービスが提供するパフォーマンスインジケーターを使用して、パフォーマンス上の問題の潜在的な根本原因を特定します。 次に、関連するログで詳細な情報を探し、障害をさらに診断してトラブルシューティングを行います。

  トラブルシューティングのセクションでは、パフォーマンスに関する一般的な問題と、トラブルシューティング方法の例を多く挙げています。 こちらは参考用としてご利用ください。

- エラー診断

  クライアントアプリケーションからのリクエストに障害が発生すると、クライアントはサーバーからエラー情報を受け取ります。 モニタリングサービスは、これらのエラーを記録し、リクエストに影響を与える可能性のある各種エラーの統計を表示します。 サーバーログ、クライアントログ、およびネットワークログから個々のリクエストに関する詳細情報を取得することもできます。 一般的に、返された HTTP ステータスコード、OSS エラーコード、および OSS エラー情報は、リクエストの失敗の原因を示しています。

  エラー応答情報の詳細については、「OSS のエラー応答」をご参照ください。

- ログ機能の使用

  OSS は、ユーザーリクエストに対してサーバーログ機能を提供します。 この機能は、端末相互間の詳細なリクエストログをトラッキングするのに役立ちます。

  ログ機能の有効化と使用方法については、「ログの設定」をご参照ください。

ログサービスの命名規則とレコード形式の詳細については、「サーバーアクセスログ」をご参照ください。

- ネットワークログツールの使用

  多くの状況において、ログ機能を使用してストレージログとクライアントアプリケーションのログデータを記録することで、問題を診断することができます。 ただし、状況によっては、ネットワークログツールを使用し、詳細情報を取得することが必要になる場合があります。

  これは、クライアントとサーバー間で交換されるトラフィックをキャプチャすることで、クライアントとサーバー間で交換されたデータと、その基礎的なネットワークステータスに関する詳細情報を得られるため、問題の調査に役立つからです。 たとえば、ユーザーのリクエストによってエラーが報告されたにもかかわらず、サーバーログにリクエストが表示されない場合もあります。 このような場合、OSS ログ機能によって記録されたレコードを使用して、問題の原因がクライアントにあるかどうかを確認したり、またはネットワークモニタリングツールを使用してネットワークの問題をチェックすることができます。

  Wireshark は、最も一般的なネットワークログ分析ツールの 1 つです。 このフリープロトコルアナライザーは、パケットレベルで動作し、各種ネットワークプロトコルの詳細なパケット情報を表示します。 Wireshar は、パケットの損失や接続の問題をトラブルシューティングするのに役立ちます。

E2E のトラッキングと診断

リクエストはクライアントアプリケーションプロセスによって開始され、ネットワーク環境を介して OSS サーバーに渡され、そこで処理されます。 次に、ネットワーク環境を介してサーバーから応答が送信され、クライアントによって応答が受信されます。 これは端末相互間でのトラッキングプロセスです。 クライアントアプリケーションログ、ネットワークトラッキングログ、およびサーバーログを関連付けることで、問題の根本原因をトラブルシューティングし、潜在的な問題を発見するための詳細な情報が提供されます。

OSS では、提供される RequestID は、各種ログからの情報の関連付けに使用される識別子として機能します。 また、ログのタイムスタンプでは、特定のログの時間範囲を迅速に照会できるだけでなく、この期間中における、リクエストイベントやその他のクライアントアプリケーション、ネットワーク、サービスシステムイベントの発生時点を確認することもできます。 ログのタイムスタンプは、問題の分析と調査に役立ちます。

- RequestID

  OSS はリクエストを受け取るたびに、一意のサーバーリクエスト ID と、その RequestID を割り当てます。 異なるログでは、RequestID はさまざまなフィールドにあります。

  - OSS ログ機能によって記録されたサーバーログでは、RequestID は "Request ID" 欄にあります。
  - ネットワークトラッキングのプロセス (たとえば、データストリームをキャプチャするために Wireshark を使用する場合) では、RequestID は応答メッセージの x-oss-request-id ヘッダー値です。
  - クライアントアプリケーションでは、クライアントコードを使用し、クライアントログに RequestID を手動で出力する必要があります。 最新の Java SDK バージョンでは、プレス時における通常のリクエストに対する RequestID 情報の出力がすでにサポートされています。 getRequestId 操作を使用することで、各種 API によって返された結果から RequestID を取得することができます。 すべての OSS SDK バージョンでは、異常なリクエストに対する RequestID を出力することができます。 この情報を取得するには、OSSException の getRequestId メソッドを呼び出します。

- タイムスタンプ

タイムスタンプを使用することで、関連するログエントリを検索することができます。 クライアントの時間とサーバーの時間との間には、多少のずれがある可能性に注意する必要があります。 クライアントでは、タイムスタンプを使用することで、ログ機能によって記録されたサーバーログエントリを検索します。 検索の際には、15 分を加算または減算する必要があります。

トラブルシューティング

一般的なパフォーマンス関連の問題

- 低い平均サーバーレイテンシかつ高い平均 E2E レイテンシ

  平均 E2E レイテンシと平均サーバレイテンシの違いについてはすでに説明しました。 したがって、高い E2E レイテンシと低いサーバーレイテンシは、次の 2 つの原因が考えられます。

  - クライアントアプリケーションの応答速度が遅い
  - ネットワーク要因

  クライアントアプリケーションの応答速度が遅いのには、いくつかの原因が考えられます。

  - 使用可能な接続数またはスレッド数が制限されている
    - 関連するコマンドを使用して、TIME_WAIT ステータスにシステムに多数の接続があるかどうかを確認します。 多数の接続がある場合、この問題を解決するため、コアパラメーターを調整します。
    - 使用可能なスレッド数が制限されている場合、まずクライアント CPU、メモリ、ネットワーク、またはその他のリソースに影響を与えるボトルネックをチェックします。 ボトルネックが見つからない場合は、同時スレッド数を適切に増やします。
    - 問題が解決しない場合、クライアントコードを最適化する必要があります。 たとえば、非同期アクセスメソッドを使用します。 また、パフォーマンス分析機能を使用してクライアントアプリケーションのホットスポットを分析し、必要な最適化を実行することもできます。
  - CPU、メモリ、帯域幅などのリソースが不足している
    - この種の問題の場合、まず関連するシステムモニタリング機能を使用し、クライアントリソースのボトルネックを検出する必要があります。 次に、クライアントコードを最適化してリソースの使用を合理化するか、クライアントリソースを増やします (コア数またはメモリを増やします)。

  ネットワークレイテンシの問題の調査

  一般的に、ネットワーク要因による高い E2E レイテンシは一時的です。 Wiresharkを使用することで、パケット損失の問題など、一時的および持続的なネットワークの問題を調査できます。

- 低い平均 E2E レイテンシかつ低い平均サーバーレイテンシであるものの、高いクライアントリクエストレイテンシの場合

  クライアントのリクエストレイテンシが長い場合、最も可能性の高い原因は、リクエストがサーバーに届いていないためです。 そのため、クライアントリクエストがサーバーに届いていない理由を調べる必要があります。

  2 つのクライアント側の要因により、クライアントリクエストの送信レイテンシが長くなる可能性があります。

  - 使用可能な接続またはスレッドの数が限られている場合: 前のセクションで説明した解決策をご参照ください。

- クライアントリクエストが複数回再試行される場合: この場合、再試行情報に基づき、リクエストが再試行された原因を見つけて解決する必要があります。 次の手順を実行して、クライアントに再試行の問題があるかどうかを判断します。
  - クライアントログを確認します。 詳細なログエントリは、再試行が発生したかどうかを示します。 例として、OSS Java SDK を使用すると、次の警告レベルまたは情報レベルのログエントリを検索することができます。 そのようなエントリがログに見つかった場合、リクエストが再試行されたことを示します。

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

  - クライアントのログレベルがデバッグの場合は、次のログエントリを検索します (ここでも、例として OSS Java SDK を使用しています)。 そのようなエントリが存在する場合、リクエストが再試行されたことを示します。

    ```
    Retrying on
    ```

  クライアントに問題がない場合は、パケット損失などの潜在的なネットワークの問題をチェックする必要があります。 Wireshark などのツールを使用することで、ネットワークの問題を調査することができます。

- 高い平均サーバーレイテンシ

  ダウンロードまたはアップロード中のサーバーのレイテンシが長い場合は、次の 2 つの要因が考えられます。

  - 多数のクライアントが同一の小さなオブジェクトに頻繁にアクセスしている

    この状況では、ログ機能によって記録されたサーバーログを表示して、小さなオブジェクトまたは小さなオブジェクトグループが短期間に頻繁にアクセスされているかどうかを判断します。

    ダウンロードシナリオでは、パフォーマンスを向上させるため、このバケットの CDN サービスを有効化することを推奨します。 これにより、トラフィックの使用料を削減できます。 アップロードシナリオの場合、これがビジネスに影響を与えないのであれば、このオブジェクト (バケット) に対する書き込み権限を取り消すことを検討してください。

  - 内部システム要因

    内部システムの問題や最適化では解決できない問題については、クライアントログまたはログ機能で記録されたログにある RequestID を弊社システムスタッフにご提出ください。ご提出いただいた RequestID は問題解決に役立てることができます。

サーバーエラー

サーバー側エラーの数が増加した場合、2 つのシナリオを考慮する必要があります。

- 一時的な増加

  この種の問題では、クライアントプログラムで再試行ポリシーを調整し、指数関数的バックオフなどの合理的な譲歩メカニズムを採用する必要があります。 これにより、システムの最適化、アップグレード、およびその他の操作 (システム負荷分散のためのパーティション移行など) により、一時的にサービスが利用できなくなるのを回避できるだけでなく、ビジネスピーク時の高いプレッシャーも回避することができます。

- 持続的な増加

サーバー側のエラーの数が持続的に増加する場合、バックエンドのスタッフにクライアントログ、またはログ機能によって記録されたログの RequestID をご提供ください。問題の発見に役立てることができます。

ネットワークエラー

サーバーがリクエストを処理しているときに接続が失われると (サーバー側の問題によるものではない)、ネットワークエラーが発生するため、HTTP リクエストヘッダーを返すことができません。 そのような状況では、システムはこのリクエストに対して 499 の HTTP ステータスコードを記録します。 次の状況では、サーバーはリクエストステータスコードを 499 に変更する可能性があります。

- 受信した読み取り/書き込みリクエストを処理する前に、サーバーが接続が利用できないことを検出すると、そのリクエストは 499 として記録されます。
- サーバーがリクエストを処理しているときにクライアントが優先的に接続を閉じると、リクエストは 499 として記録されます。

要約すると、クライアントが自主的にリクエストを閉じるか、またはクライアントがネットワークから切断された場合、リクエストプロセス中にネットワークエラーが発生します。クライアントが自主的にリクエストを閉じた場合、クライアントコードをチェックすることで、クライアントが OSS から切断された原因と時間を特定することができます。 クライアントがネットワーク接続を失った場合は、Wireshark などのツールを使用してネットワーク接続の問題を調査します。

クライアントエラー

- クライアント権限付与エラーの増加

  クライアント権限付与エラーの増加を検出した場合、またはクライアントが多数の 403 リクエストエラーを受信した場合、一般的に以下の問題が原因です。

  - ユーザーがアクセスしたバケットドメイン名が正しくない
    - ユーザーが第 3 レベルまたは第 2 レベルドメイン名を使用してバケットにアクセスする場合、ドメイン名で示されるリージョン内にそのバケットがないと、403 エラーが発生する可能性があります。 たとえば、杭州リージョンでバケットを作成したけれども、あるユーザーがドメイン名 Bucket.oss-cn-shanghai.aliyuncs.com を使用してバケットへのアクセスを試みたとします。 この場合、バケットのリージョンを確認し、ドメイン名の情報を修正する必要があります。
    - CDN アクセラレーションサービスを有効化した場合、CDN が正しくないオリジン取得ドメイン名をバインドすると、この問題が発生することがあります。 この場合、CDN オリジン取得ドメイン名がバケットの第 3 レベルドメイン名であることをチェックします。
  - JavaScript クライアントを使用しているときに 403 エラーが発生した場合、Web ブラウザーで "同一ソースポリシー" のセキュリティ制限が実装されているため、CORS (クロスオリジンリソース共有) 設定の問題が原因である可能性があります。 この場合、バケットの CORS 設定をチェックし、エラーを修正する必要があります。 CORS 設定については、「CORS」をご参照ください。

- アクセス制御の問題は、4 つのタイプに分けられます。
  - アクセスにプライマリ AK を使用する場合、AK が無効な場合は、AK設定でエラーをチェックする必要があります。
  - アクセスに RAM サブアカウントを使用する場合は、そのサブアカウントが正しいサブアカウント AK を使用していることと、そのサブアカウントに適切なアクセス許可があることを確認する必要があります。
  - アクセスに一時的な STS トークンを使用する場合は、一時的なトークンが有効期限切れになっていないことを確認する必要があります。 トークンの有効期限が切れている場合は、新しいトークンを申請します。
  - アクセス制御にバケットまたはオブジェクト設定を使用する場合は、アクセスするバケットまたはオブジェクトが関連操作をサポートしていることをチェックする必要があります。
- サードパーティーのダウンロード (署名付き URL を使用してOSSリソースにアクセスする) を許可した際、以前はアクセスが正常だったのに突然 403 エラーが報告された場合、URL が有効期限切れになっている可能性があります。
- RAM サブアカウントで OSS ユーティリティを使用すると、403 エラーが発生することもあります。これらのユーティリティには、ossftp、ossbrowser、および OSS コンソールクライアントが含まれます。 ログイン時に関連する AK 情報を正しく入力したにもかかわらず、システムがエラーをスローした場合、AK がサブアカウント AK であり、このサブアカウントが GetService およびその他の操作に対するアクセス許可を持っていることをチェックする必要があります。

- クライアント側の "存在しないリソース" エラーの増加

  クライアントが 404 エラーを受け取った場合、存在しないリソースまたは情報にアクセスしようとしていることを意味します。 モニタリングサービスが「リソースが存在しません」エラーの増加を検出した場合、おそらく次のいずれかの問題が原因になっています。
  - サービスの使用方法: たとえば、別の操作を実行する前に、まずオブジェクトが存在することをチェックする必要があり、doesObjectExist メソッドを呼び出したとします (例としてJava SDK を使用)。ここでオブジェクトが存在しない場合、クライアントは値 "false" を受け取ります。 しかしながら、サーバーは実際には 404 リクエストエラーを生成します。 そのため、このビジネスシナリオでは、404 エラーは正常です。
  - クライアントまたは他のプロセスが、以前にこのオブジェクトを削除: ログ機能によって、記録されたサーバーログの関連するオブジェクト操作を検索することにより、この問題を確認します。
  - ネットワーク障害によるパケット損失と再試行:例えば、クライアントは、特定のオブジェクトを削除するため、削除操作を行うことがあります。 そのリクエストはサーバーに届き、削除操作を正常に実行します。 しかしながら、ネットワーク上での送信中に応答パケットが失われると、クライアントは再試行を開始します。 この 2 番目のリクエストは 404 エラーを生成します。クライアントログとサーバーログを使用することで、ネットワークの問題が 404 エラーを生成していることを確認できます。
    - クライアントアプリケーションログの再試行リクエストをチェックします。
    - このオブジェクトに対する 2 つの削除操作がサーバーログに表示されていること、および最初の削除操作の HTTP ステータスが 2xx であることをチェックします。

- 低い有効リクエスト率、およびその他の多数のクライアント側リクエストエラー

有効リクエスト率とは、2xx / 3xx の HTTP ステータスコードを全体のリクエストの割合として返すリクエストの数です。 4XX または 5XX のステータスコードは失敗したリクエストを示し、有効リクエスト率を下げます。 他のクライアント側のリクエストエラーは、以下のエラーを除くリクエストエラーを示します。: サーバーエラー (5xx)、ネットワークエラー (499)、クライアント許可エラー (403)、存在しないリソースエラー (404)、クライアントタイムアウトエラー (408 または OSS エラーコード: RequestTimeout 400)。

ログ機能によって記録されたサーバーログをチェックし、これらのリクエストによって発生した特定のエラーを判別します。 OSS エラー応答を確認し、OSS から返される一般的なエラーコードの一覧を参照します。 次に、クライアントコードを調べて、これらのエラーの具体的な原因を発見し、解決します。

ストレージ容量の異常な増加

アップロードリクエスト内の対応する増加なしに、ストレージ容量が異常に増加した場合、これは一般的に削除の問題により引き起こされています。 このような場合は、次の 2 つの要素をチェックします。

- クライアントアプリケーションがストレージオブジェクトを定期的に削除し、領域を解放するために特定のプロセスを使用する場合: このリクエストの調査プロセスは次のとおりです。
  i. 削除リクエストが失敗すると、ストレージオブジェクトが予期したとおりに削除されない可能性があるため、有効リクエスト率が減少したかどうかを確認します。
  ii. リクエストのエラータイプを調べて、有効リクエスト率が低下した具体的原因を見つけます。 その後、特定のクライアントログをまとめて、詳細なエラー情報を確認します (たとえば、ストレージスペースを解放するために使用される STS の一時的なトークンが期限切れになっている可能性があります)。
- クライアントがストレージオブジェクトを削除するために LifeCycle を設定した場合: コンソールまたは API を使用して、現在のバケットの LifeCycle 値が以前と同じであることをチェックします。 以前と同じではない場合は、構成を変更し、ログ機能によって記録されたサーバーログを使用して、この値の以前の変更に関する情報を見つけます。 LifeCycle が正常であるものの非アクティブである場合は、チケットを起票し、サポートセンターへお問い合わせください。

その他の OSS の問題

トラブルシューティングのセクションで問題が解決されなかった場合は、次の方法のいずれかを使用し問題を診断してトラブルシューティングを行います。

1. OSS モニタリングサービスを表示し、予想されるベースライン動作と比較して何らかの変更があったかどうかを確認します。 モニタービューを使用することで、この問題が一時的なものか持続的なものか、およびどのストレージ操作が影響を受けているのかを判別することができます。
2. モニタリング情報は、ログ機能によって記録されたサーバーログデータを検索し、問題の開始時に発生した可能性のあるエラーに関する情報を検索するのに役立ちます。 モニタリング情報は、問題の発見と解決に役立つ可能性があります。
3. サーバーログの情報が不十分な場合は、クライアントアプリケーションを調査するためにクライアントログを使用するか、または Wireshark などのネットワークツールを使用してネットワークに問題がないかをチェックします。

9.イベントの通知

OSS では、イベント通知を設定して、リソースに関連する操作を適切なタイミングで確認できます。

イベント通知の作成時にイベント通知ルールをカスタマイズすると、 OSS 内の指定したオブジェクトに対する操作が適切なタイミングで通知されます。 例:

- "新しいデータが、画像コンテンツ共有プラットフォーム、オーディオおよびビデオプラットフォームから OSS にアップロードされました。"
- "OSS の関連コンテンツが更新されました。"
- "OSS 上の重要なファイルが削除されます。"
- "OSS 上のデータの同期が完了しました。"

> 注意 RTMP インジェストで ts および m3u8 ファイルが作成されても通知を送信されません。

OSS イベント通知は非同期であり、通常の OSS 操作には影響しません。 イベント通知を構成するには、ルールおよびメッセージ通知.

- ルール: 通知を送信する OSS リソースの使用条件を指定します。
- メッセージ通知: Alibaba Cloud MNS により、複数の通知方法を提供します。

下図に、OSS リソースのイベント通知に関する全体的なアーキテクチャを示します。

実装モード

イベント通知の設定の詳細については、「イベント通知の設定」をご参照ください。

イベントタイプ

イベントタイプ	説明
ObjectCreated:PutObject	オブジェクトの作成/上書き: 簡単なアップロード。
ObjectCreated:PostObject	オブジェクトの作成/上書き: フォームのアップロード。
ObjectCreated:CopyObject	オブジェクトの作成/上書き: オブジェクトのコピー。
ObjectCreated:InitiateMultipartUpload	オブジェクトの作成/上書き: マルチパートアップロードタスクが初期化されました。
ObjectCreated:UploadPart	オブジェクトの作成/上書き: パーツがアップロードされました。
* oss: ObjectCreated: UploadPartCopy	オブジェクトの作成/上書き: パーツは、既存のオブジェクトからデータをコピーすることによりアップロードされました。
ObjectCreated:CompleteMultipartUpload	オブジェクトの作成/上書き: マルチパートアップロードタスクが完了しました。
ObjectCreated:AppendObject	オブジェクトの作成/上書き: 追加アップロード。

イベントタイプ	説明
ObjectDownloaded:GetObject	オブジェクトのダウンロード：単純なダウンロード。
ObjectRemoved:DeleteObject	オブジェクトが削除されました。
ObjectRemoved:DeleteObjects	複数のオブジェクトが削除されました。
ObjectReplication:ObjectCreated	オブジェクトの操作を同期しました：オブジェクトが作成されました。
ObjectReplication:ObjectRemoved	オブジェクトの操作を同期しました：オブジェクトは削除されました。
ObjectReplication:ObjectModified	オブジェクトの操作を同期しました：オブジェクトは上書きされました。

通知メッセージの形式

OSS ベースのイベント通知のメッセージコンテンツは、Base64 でエンコードされます。 デコードされたコンテンツは JSON 形式です。 例:

```
"events": {
   "eventName": "",  //The event type.
   "eventSource": "", //The resource on which operations were performed and triggered event notification. The value is acs:oss.
   "eventTime": "", //The time when the event was triggered. The returned data is in the ISO 8601 standard in the yyyy-MM-ddTHH:mm:ssZ format.
   "eventVersion": "", //The version number. The version number is 1.0.
   "oss": {
      "bucket": {
         "arn": "", //The ARN for the bucket. The format is "acs:oss:region:uid:bucket".
         "name": "", //The name of the bucket.
         "ownerIdentity": ""}, //The owner of the bucket.
      "object": {
         "deltaSize": , //The changed size of the object. If you create an object, this value is the size of the object. If you overwrite an existing object, this value is the difference between the new and existing objects, which may be a negative value.
         "eTag": "", //The ETag of the object. This value is the same as the ETag header value in the response to the GetObject() request.
         "key": "", //The name of the object.
         "position":, //Required only for the ObjectCreated:AppendObject event. This parameter indicates the position calculated based on the size of the object last uploaded. The value starts from 0.
         "readFrom": , //Required only for the ObjectDownloaded:GetObject event. This parameter indicates the beginning of the range of Bytes to read data from the object. The value starts from 0 for the range request. Otherwise, the value is 0.
         "readTo": , //Required only for the ObjectDownloaded:GetObject event. This parameter indicates the ending of the range of Bytes to read data from the object. The value increases by one for the ending Byte in the range request. Otherwise, the value is the actual size of the object.
         "size": }, //The size of the object.
      "ossSchemaVersion": "", //The version of the schema. The value is 1.0.
      "ruleId": "GetObject"}, //The rule ID that matches the event.
      "region": "", //The region in which the bucket is located.
      "requestParameters": {
         "sourceIPAddress": ""}, //The source IP address in the request.
      "responseElements": {
         "requestId": ""}, //The ID of the request.
      "userIdentity": {
         "principalId": ""}, //The user ID of the requester.
      "xVars": {  //The custom parameters in the callback of OSS.
         "x:callback-var1":"value1",
         "x:vallback-var2":"value2"}}]}
```

例：


```
{"events": [{
    "eventName": "ObjectDownloaded:GetObject",
    "eventSource": "acs:oss",
    "eventTime": "2016-07-01T11:17:30.000Z",
    "eventVersion": "1.0",
    "oss": {
        "bucket": {
            "arn": "acs:oss:cn-shenzhen:11489********46818:event-notification-test-shenzhen",
            "name": "event-notification-test-shenzhen",
            "ownerIdentity": "11489********46818"},
        "object": {
            "deltaSize": 0,
            "eTag": "0CC175B9C0F1B6xxxxxx99E269772661",
            "key": "test",
            "readFrom": 0,
            "readTo": 1,
            "size": 1},
        "ossSchemaVersion": "1.0",
        "ruleId": "GetObjectRule"},
        "region": "cn-shenzhen",
        "requestParameters": {
            "sourceIPAddress": "140.xx.xx.90"},
        "responseElements": {
            "requestId": "5776514Axxxxxxx542425D2B"},
        "userIdentity": {
            "principalId": "11489********46818"},
        "xVars": {
            "x:callback-var1":"value1",
            "x:vallback-var2":"value2"}}]}
```

10.トラブルシューティング

11.非表示

11.1. コンプライアンス保持戦略

11.1.1. FAQ

このトピックは、OSS リソースのコンプライアンス保持ポリシーを使用する際のよくある質問に回答します。

コンプライアンス保持ポリシーの利点は何ですか。

コンプライアンス保持ポリシーにより、業界規制のコンプライアンス要件に準拠した方法でデータが保存されます。 コンプライアンス保持ポリシーによって保護されているデータは、保護期間内に変更または削除することはできません。 ただし、RAM ポリシーまたはバケットポリシーを使用してデータを保護する場合、変更または削除できる可能性があります。

コンプライアンス保持ポリシーをいつ設定する必要がありますか。

変更や削除が許可されていない重要データ (医療記録、技術文書、契約書など) を長期間保存する場合、バケットにデータを保存し、そのバケットにコンプライアンス保持ポリシーを設定することでデータを保護できます。

オブジェクトにコンプライアンス保持ポリシーを設定できますか。

バケットにのみコンプライアンス保持ポリシーを設定できます。ディレクトリやオブジェクトには設定できません。

オブジェクトを保存するバケットにコンプライアンス保持ポリシーを設定した後、オブジェクトのストレージクラスを変更できますか。

バケットにコンプライアンス保持ポリシーを設定した後、ライフサイクルルールを設定して、保護対象のバケット内のオブジェクトのストレージクラスを変換すると、ストレージ料金を節約できます。

コンプライアンス保持ポリシーが設定されているバケットを削除するにはどうすればよいですか。

- バケットにオブジェクトが保存されていない場合、バケットを直接削除できます。
- 保護期間外のオブジェクトがバケットに保存されている場合、バケットを削除しようとするとエラーが発生します。 この場合、最初にバケット内のオブジェクトを削除してから、バケットを削除します。
- 保護期間内のオブジェクトがバケットに保存されている場合、バケットを削除することはできません。

アカウントに滞納請求書がある場合、コンプライアンス保持ポリシーの保護期間内のオブジェクトはそのまま保持されますか。

アカウントに滞納請求書がある場合、契約条件に基づいてデータ保持ポリシーがデータに適用されます。

許可された RAM ユーザーとしてコンプライアンス保持ポリシーを設定できますか。

コンプライアンス保持ポリシーに関連する API が公開されていて、RAM ポリシーに統合されています。RAM ポリシーによって許可された RAM ユーザーを使用することで、コンソールまたは API や SDK を使用して、コンプライアンス保持ポリシーを作成または削除できます。

11.2. アクセス制御

11.2.1. プライマリアカウントを使用せずにバケットにアクセスする

たとえば、ユーザーはモバイル開発者であり、開発、テスト、その他の機能のためのバケット ram-test-dev を 1 つだけ持っているとします。 ユーザーは、このバケットにアクセスするためには、プライマリアカウントの使用を中止する必要があります。 プライマリアカウントを使用しないことで、AccessKey およびパスワードの漏洩により引き起こされる問題を回避することができます。 次の例では、AccessKey を自分の AccessKey に置き換えます。 手順は次のとおりです。

1. コンソールで、[製品とサービス] > [サービスリソースのアクセス管理] をクリックします。

   > 説明　サービスを一度も使用したことがない場合は、最初にサービスを有効にする必要があります。

2. **[ユーザー]** をクリックし、**[ユーザー管理]**ページに移動します。
3. このページでは、ユーザーの作成がされていないことを示します。 プライマリアカウントと同一の OSS アクセス権限を持つサブアカウントを作成するには、右上にある **[新しいユーザー]** をクリックします。 このユーザーの作成では、**[このユーザーの AccessKey を自動生成する]** をクリックします。
4. 次に、このアカウントの AccessKey が生成されるので、後で使用するため保存します。
5. **ユーザー管理**インターフェイスに戻ると、ram_test という名前の新しく作成されたアカウントが表示されます。 作成時には、このサブアカウントにはまだ権限がありません。右側の **[権限付与]** リンクをクリックし、このサブアカウントに OSS の完全なアクセス権限を付与します。

権限付与の後、サブアカウントのコンソールにログイン権限、またはその他の権限を付与する場合は、右側の **[管理]** リンクをクリックします。

これで、アップロード操作とダウンロード操作をテストできます。 この例では、AccessKey は ram_test の AccessKey です。 テスト中では、ram_test の AccessKey を自分の AccessKey に置き換えます。

```
$./osscmd get
oss://ram-test-dev/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOhue******Frogv -k OmV
wFJO3qcT0******FhOYpg3p0KnA
100% The object test.txt is downloaded to test.txt, please check.
0.069(s) elapsed
```

```
$./osscmd put test.txt oss://ram-test-dev/test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOhue*****
*Frogv -k OmVwFJO3qcT0******FhOYpg3p0KnA
100%
Object URL is: http://ram-test-dev.oss-cn-hangzhou.aliyuncs.com/test.txt
Object abstract path is: oss://ram-test-dev/test.txt
ETag is "E27172376D49FC609E7F46995E1F808F"
0.108(s) elapsed
```

ご覧のとおり、このサブアカウントは基本的にすべての操作で使用することができるため、プライマリアカウントの AccessKey の漏洩を回避することができます。

11.2.2. 読み取り権限と書き込み権限の分離

ユーザーがアプリケーションサーバーを使用して外部サービスを提供する場合、OSS はバックエンドの静的リソースを格納します。 この場合、攻撃リスクを軽減するため、アプリケーションサーバーには OSS に対して読み取り専用の権限を付与することを推奨します。 読み取り権限と書き込み権限を分離することで、アプリケーションサーバーに対し、読み取り専用権限を持つユーザーのアクセスを許可するように設定することができます。

1. アカウント ram_test_pub を作成します。 次の図に示すように、権限付与管理エリアで [ReadOnly] を選択します。
2. これで、サブアカウントの AccessKey を使用してアップロードとダウンロードのアクセス許可をテストすることができます。 ここでの AccessKey は ram_test_pub の AccessKey であり、テスト中、自分の AccessKey に置き換えます。

```
$./osscmd get oss://ram-test-dev/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOhue*****
*Frogv -k OmVwFJO3qcT0******FhOYpg3p0KnA
100% The object test.txt is downloaded to test.txt, please check.
0.070(s) elapsed
```

```
$. /Osscmd put test.txt OSS: // Ram-test-dev/test.txt -- Host = porteroohue ****** frogv-K OmVwFJO3q
cT0 * FhOYpg3p0KnA?
100% Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '5646E49
C1790CF0F531BAE0D'), ('date', 'Sat, 14 Nov 2015 07:37:00 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
 <Code>AccessDenied</Code>
 <Message>AccessDenied</Message>
 <RequestId>5646E49C1790CF0F531BAE0D</RequestId>
 <HostId>ram-test-dev.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
put Failed!
```

上記の例に関して、ram_test_pub アカウントを使用してファイルをアップロードすることはできないと結論付けることができます。

11.2.3. バケットのアクセス許可の分離

このセクションでは別のシナリオを紹介します。 他のユーザーが最新のアプリを使用している場合は、個人用のバケットを使用して自分のアプリデータを保存することができます。 そのバケットが ram-test-app であるとします。 アクセス許可を分離することを考慮し、アプリケーションサーバーに対して、ram-test-app へのアクセスを許可しないように設定する必要があります。つまり、アカウント ram_test_pub は ram-test-dev の読み取りのみが許可されます。 これは RAM 権限システムによっても実現できます。 手順は次のとおりです。

1. システムには既定のバケットレベルのポリシーがないため、カスタマイズポリシーを作成する必要があります。

   バケットアクセスポリシーは次のとおりです。 詳しくは、「RAM ポリシーの詳細」および「OSS 権限付与に関するよくあるご質問」をご参照ください。

```
{
"Version": "1",
"Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "oss:ListObjects",
   "oss:GetObject"
  ],
  "Resource": [
   "acs:oss:*:*:ram-test-dev",
   "acs:oss:*:*:ram-test-dev/*"
  ]
 }
]
}
```

設定後、カスタマイズ権限付与ポリシーリスト内でポリシーを確認します。

2. ユーザー権限付与管理で、選択した権限付与ポリシーリストにこのポリシーを追加します。 また、ユーザー管理権限付与ポリシーで、以前に許可されたすべての OSS 読み取り権限を取り消すことができます。 > >
3. 設定した権限の有効性をテストします。
   - ram-test-dev のオブジェクトにアクセスします。

     ```
     $./osscmd get oss://ram-test-dev/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOh
     ue******Frogv -k OmVwFJO3qcT0******FhOYpg3p0KnA
     100% The object test.txt is downloaded to test.txt, please check.
     0.047(s) elapsed
     ```

   - ram-test-app 内のオブジェクトにはアクセスできません。

```
$./osscmd get oss://ram-test-app/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOh
ue******Frogv -k OmVwFJO3qcT0******FhOYpg3p0KnA
Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '5
646ED53F9EEA2F3324191A2'), ('date', 'Sat, 14 Nov 2015 08:14:11 GMT'), ('content-type', 'applicatio
n/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>5646ED53F9EEA2F3324191A2</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
  </Error>
Error Status:
403
get Failed!
```

- ファイルは oss-test-app にアップロードできません。

```
$./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOh
ue******Frogv -k OmVwFJO3qcT0******FhOYpg3p0KnA
100% Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '5
646ED7BB8DE437A912DC7A8'), ('date', 'Sat, 14 Nov 2015 08:14:51 GMT'), ('content-type', 'applicati
on/xml')]
Error Body:
<? XML version = "1.0" encoding = "UTF-8 "? >
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>5646ED7BB8DE437A912DC7A8</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error status:
403
put Failed!
```

上記の設定を使用し、ram-test-dev と ram-test-app のアクセス許可は正しく分離されました。

前述のセクションでは、サブアカウントのアクセス許可制御機能を使用することで権限を分離し、情報漏洩の潜在的なリスクを最小限に抑える方法について説明しました。

より複雑なアクセス制御を実装したい場合は、「RAM ユーザーガイド」をご参照ください。

11.2.4. STS の一時的なアクセス許可

これまでのドキュメントでは、RAM ユーザー機能のみを使用していました。 RAM ユーザー機能のみを使用したユーザーアカウントは、長期的な通常の使用に用いられます。 情報が漏えいした場合、 RAM ユーザーのアクセス許可をすぐに無効にできないと、重大なリスクとなります。

前の例では、開発者のアプリがユーザーに対して、OSS バケット am-test-app へのデータのアップロードを許可しており、現在、アプリのユーザーは多数いると仮定しています。 この場合、アプリはどのようにしてデータのアップロード権限を多数のユーザーに安全に付与し、複数のユーザー間で確実にストレージをわけることができるのでしょうか。

そのようなシナリオでは、STS を使用して一時的なアクセスをユーザーに許可する必要があります。 STS を使用することで、指定したユーザーに最低限必要な権限を付与することにより、そのユーザーを制限する複雑なポリシーを指定することができます。

ロールの作成

前のドキュメントの例に基づき、アプリユーザーは個人データを保存するためのバケット、ram-test-app を持っているとします。 ロールは次のように作成されます。

1. 前のドキュメントで示したプロセスを使用し、ram_test_app という名前の RAM ユーザーアカウントを作成します。 アカウントは引き受けるロールの権限を継承するため、このアカウントにはいかなる権限も付与しないようにします。
2. ロールを作成します。ここでは、各ユーザーがそれぞれ読み取り操作を実行してファイルをアップロードするため、2 つのロールを作成する必要があります。
   - RAM コンソールにログインし、[ロール] > [新しいロール] とクリックします。 > .
   - ロールタイプを選択します。 ここではユーザーロールをクリックする必要があります。
   - ロールタイプ情報を入力します。 これは、このロールが自身の Alibaba Cloud アカウントによって使用されていたためです。 既定の設定を使用します。
   - ロールの基本情報の設定
3. ロールが作成された時点では、権限はありませんでした。 そのため、前述のプロセスを使用し、カスタマイズ権限付与ポリシーを作成する必要があります。 権限付与ポリシーは次のとおりです。

```
{
"Version": "1",
"Statement": [
 {
  "Effect": "allow ",
  "Action": [
   "oss:ListObjects",
   "Oss: GetObject"
  ],
  "Resource": [
   "acs:oss:*:*:ram-test-app",
   "acs:oss:*:*:ram-test-app/*"
  ]
 }
]
}
```

これは ram-test-app に対する読み取り専用の権限を示しています。

4. ポリシーを作成した後、ロール管理ページで、RamTestAppReadOnly ロールに対して、ram-test-app の読み取り専用権限を付与します。
5. 同じ手順を実行して、RamTestAppWrite ロールを作成し、カスタマイズ権限付与ポリシーを使用し、ram-test-app の書き込み権限を付与します。 権限付与ポリシーは次のとおりです。

```
{
"Version": "1",
"Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "oss:DeleteObject",
   "oss:ListParts",
   "oss:AbortMultipartUpload",
   "oss:PutObject"
  ],
  "Resource": [
   "acs:oss:*:*:ram-test-app",
   "acs:oss:*:*:ram-test-app/*"
  ]
 }
]
}
```

これで、RamTestAppReadOnly とRamTestAppWrite の 2 つのロールが作成されました。それぞれ、ram-test-app に対する読み取り専用権限および書き込み権限を持っています。

一時的なアクセス許可

ロールを作成したら、それらを使用して OSS への一時的なアクセス権を付与します。

準備

ロールを引き受けるには権限付与が必要です。権限付与を行わない場合、RAM ユーザーなら誰でもこれらのロールを引き受けることができてしまい、それが予測不可能なリスクにつながる可能性があります。したがって、対応するロールを引き受けるため、RAM ユーザーは明示的に設定された権限を持っている必要があります。

1. [カスタマイズ権限付与ポリシーの管理] にて、 2 つのカスタマイズ権限付与ポリシーを作成します。

```
{
"Statement": [
 {
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Resource": "acs:ram::1894xxxxxx722283:role/ramtestappreadonly"
 }
],
"Version": "1"
}
```

同じ方法で、別のカスタマイズ権限付与ポリシーを作成します。

```
{
"Statement": [
 {
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Resource": "acs:ram::1894xxxxxx722283:role/ramtestappwrite"
 }
],
"Version": "1"
}
```

ここでは、Resource の後に入力されたコンテンツがロール ID になります。 ロール ID は、[ロール] の [ロールの詳細] にあります。 >

2. アカウント ram_test_app に 2 つの権限付与ポリシーを付与します。

STS を使用してアクセス許可を付与

これで、アクセス権限を付与するため、STS を正式に使用するためのプラットフォームの準備が整いました。

ここではシンプルな STS Python コマンドラインツール sts.py を使用します。 呼び出し方法は次のとおりです。

```
$python ./sts.py AssumeRole RoleArn=acs:ram::1894xxxxxx722283:role/ramtestappreadonly RoleSess
ionName=usr001 Policy='{"Version":"1","Statement":[{"Effect":"Allow","Action":["oss:ListObjects","oss:
GetObject"],"Resource":["acs:oss:*:*:ram-test-app","acs:oss:*:*:ram-test-app/*"]}]}' DurationSeconds=
1000 --id=id --secret=secret
```

- RoleArn: 引き受けるロール ID を示します。 ロール ID は [ロール] の [ロールの詳細] で確認できます。 >
- RoleSessionName: 一時的な認証情報の名前を示します。 一般的に、異なるアプリケーションユーザーを使用することで、RoleSessionName を分けることを推奨します。

- Policy: ロールが引き受けられたときに追加される権限の制限を示します。
- DurationSeconds: 一時的な認証情報の有効期限を秒レベルで示します。 最小値は 900、最大値は 3,600 です。
- id and secret: ロールを引き受ける RAM ユーザーの AccessKey を示します。

ここでは、"Policy" が意味する内容について説明します。 ここで言及されているポリシーは、ロールが引き継がれた後、一時的な認証情報の許可を制限するために使用されます。 結局のところ、一時的な認証情報によって取得された権限は、ロールと渡されたポリシーとで重複する権限です。

ロールが引き受けられた場合、柔軟性を高めるためにポリシーを入力します。 たとえば、ファイルをアップロードするとき、ユーザーごとに異なるアップロードパス制限を追加します。 次の例で詳細を示します。

以下で、STS の機能をテストします。 バケットをテストするにあたり、まずコンソールを使用し、コンテンツ ststest を含む ram-test-app 内にファイル test.txt を移動させます。

まず、RAM ユーザーアカウント ram_test_app を使用してファイルに直接アクセスします。 次に、AccessKey をテストで使用する自分のアクセスキーに置き換えます。

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd get oss://ram-test-app/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOhue*****
*Frogv -k OmVwFJO3qcT0******FhOYpg3p0KnA
Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '564A94D
444F4D8B2225E4AFE'), ('date', 'Tue, 17 Nov 2015 02:45:40 GMT'), ('content-type', 'application/xml')]
Error Body:
<?  xml version="1.0" encoding="UTF-8"? >
<Error>
 <Code>AccessDenied</Code>
 <Message>AccessDenied</Message>
 <RequestId>564A94D444F4D8B2225E4AFE</RequestId>
 <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
get Failed!
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOhue*****
*Frogv -k OmVwFJO3qcT0******FhOYpg3p0KnA
100% Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '564A94E
5B1119B445B9F8C3A'), ('date', 'Tue, 17 Nov 2015 02:45:57 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
 <Code>AccessDenied</Code>
 <Message>AccessDenied</Message>
 <RequestId>564A94E5B1119B445B9F8C3A</RequestId>
 <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
put Failed!
```

アクセス権限がないと、RAM ユーザーアカウント ram_test_app を使用したアクセス試行は失敗します。

一時的な権限を使用したダウンロード

これより、STS を使用してファイルをダウンロードします。 理解しやすくするため、入力するポリシーとロールポリシーは同一にしてあります。 有効期限は 3,600 秒に設定されており、ここでのアプリユーザー

は usr001です。 手順は次のとおりです。

1. STS を使用して一時的な認証情報を取得します。

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
 $python ./sts.py AssumeRole RoleArn=acs:ram::1894xxxxxx722283:role/ramtestappreadonly RoleSessionName=usr001 Policy='{"Version":"1","Statement":[{"Effect":"Allow","Action":["oss:ListObjects","oss:GetObject"],"Resource":["acs:oss:*:*:ram-test-app","acs:oss:*:*:ram-test-app/*"]}]}' --id=oOhue******Frogv --secret=OmVwFJO3qcT0******FhOYpg3p0KnA
 https://sts.aliyuncs.com/?SignatureVersion=1.0&Format=JSON&Timestamp=2015-11-17T03%3A07%3A25Z&RoleArn=acs%3Aram%3A%3A1894xxxxxx722283%3Arole%2Framtestappreadonly&RoleSessionName=usr001&AccessKeyId=oOhu******3Frogv&Policy=%7B%22Version%22%3A%221%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%5B%22oss%3AListObjects%22%2C%22oss%3AGetObject%22%5D%2C%22Resource%22%3A%5B%22acs%3Aoss%3A%2A%3A%2A%3Aram-test-app%22%2C%22acs%3Aoss%3A%2A%3A%2A%3Aram-test-app%2F%2A%22%5D%7D%5D%7D&SignatureMethod=HMAC-SHA1&Version=2015-04-01&Signature=bshxPZpwRJv5ch3SjaBiXLodwq0%3D&Action=AssumeRole&SignatureNonce=53e1be9c-8cd8-11e5-9b86-008cfa5e4938
{
  "AssumedRoleUser": {
    "Arn": "acs:ram::1894xxxxxx722283:role/ramtestappreadonly/usr001",
    "AssumedRoleId": "317446347657426289:usr001"
  },
  "Credentials": {
    "AccessKeyId": "STS. 3mQEbNf******wa180Le",
    "AccessKeySecret": "B1w7rCbR4dzGwNYJ******3PiPqKZ3gjQhAxb6mB",
    "Expiration": "2015-11-17T04:07:25Z",
    "SecurityToken": "CAESvAMIARKAASQQ******7683CGlhdGsv2/di8uI+X******DxM5FTd0fp5wpPK/7UctYH2MJ///c4yMN1PUCcEHI1zppCINmpDG2XeNA3OS16JwS6ESmI50sHyWBmsYkCJW15gXnfhz/OK+mSp1bYxlfB33qfgCFe97Ijeuj8RMgqFx0Hny2BzGhhTVFMuM21RRWJOZnR5Yzl1T3dhMTgwTGUiEjMxNzQ0NjM0NzY1NzQyNjI4OSoGdXNyMDAxMJTrgJ2RKjoGUnNhTUQ1QpsBCgExGpUBCgVBbG******CgxBY3Rpb25FcXVhbHMSBkFjdGlvbhogCg9vc3M6TGlzdE9iamVjdHMKDW9zczpHZXRPYmplY3QSUgoOUmVzb3VyY2VFcXVhbHMSCFJlc291cmNlGjYKGGFjczpvc3M6KjoqOnJhbS10ZXN0LWFwcAoaYWNzOm9zczoq******FtLXRlc3QtYXBwLypKEDE4OTQxODk3Njk3MjIyODNSBTI2ODQyWg9Bc3N1bWVkUm9sZVVzZXJgAGoSMzE3NDQ2MzQ3NjU3NDI2Mjg5chJyYW10ZXN0YXBwcmVhZG9ubHk="
  },
  "RequestId": "8C009F64-F19D-4EC1-A3AD-7A718CD0B49B"
}
```

2. 一時的な認証情報を使用してファイルをダウンロードします。 ここでの sts_token は STS によって返された SecurityToken です。

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
 $./osscmd get oss://ram-test-app/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i STS.
3mQEbNf******wa180Le -k B1w7rCbR4dzGwNYJ******3PiPqKZ3gjQhAxb6mB --sts_token=CAESvAMIAR
KAASQQ******7683CGlhdGsv2/di8uI+X******DxM5FTd0fp5wpPK/7UctYH2MJ///c4yMN1PUCcEHI1zppCI
NmpDG2XeNA3OS16JwS6ESmI50sHyWBmsYkCJW15gXnfhz/OK+mSp1bYxlfB33qfgCFe97Ijeuj8RMgqFx
0Hny2BzGhhTVFMuM21RRWJOZnR5Yzl1T3dhMTgwTGUiEjMxNzQ0NjM0NzY1NzQyNjI4OSoGdXNyMDA
xMJTrgJ2RKjoGUnNhTUQ1QpsBCgExGpUBCgVBbG******CgxBY3Rpb25FcXVhbHMSBkFjdGlvbhogCg9vc3
M6TGlzdE9iamVjdHMKDW9zczpHZXRPYmplY3QSUgoOUmVzb3VyY2VFcXVhbHMSCFJlc291cmNlGjYKGG
Fjczpvc3M6KjoqOnJhbS10ZXN0LWFwcAoaYWNzOm9zczoq******FtLXRlc3QtYXBwLypKEDE4OTQxODk3
Njk3MjIyODNSBTI2ODQyWg9Bc3N1bWVkUm9sZVVzZXJgAGoSMzE3NDQ2MzQ3NjU3NDI2Mjg5chJyYW10
ZXN0YXBwcmVhZG9ubHk=
 100% The object test.txt is downloaded to test.txt, please check.
 0.061(s) elapsed
```

3. ご覧のとおり、一時的な認証情報を使用してファイルをダウンロードします。 次に、一時的な認証情報を使用してファイルをアップロードできるかどうかをテストします。

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
 $./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn-hangzhou.aliyuncs.com -i STS.
3mQEbNf******wa180Le -k B1w7rCbR4dzGwNYJ******3PiPqKZ3gjQhAxb6mB --sts_token=CAESvAMIAR
KAASQQ******7683CGlhdGsv2/di8uI+X******DxM5FTd0fp5wpPK/7UctYH2MJ///c4yMN1PUCcEHI1zppCI
NmpDG2XeNA3OS16JwS6ESml50sHyWBmsYkCJW15gXnfhz/OK+mSp1bYxlfB33qfgCFe97Ijeuj8RMgqFx
0Hny2BzGhhTVFMuM21RRWJOZnR5Yzl1T3dhMTgwTGUiEjMxNzQ0NjM0NzY1NzQyNjI4OSoGdXNyMDA
xMJTrgJ2RKjoGUnNhTUQ1QpsBCgExGpUBCgVBbG******CgxBY3Rpb25FcXVhbHMSBkFjdGlvbhogCg9vc3
M6TGlzdE9iamVjdHMKDW9zczpHZXRPYmplY3QSUgoOUmVzb3VyY2VFcXVhbHMSCFJlc291cmNlGjYKGG
Fjczpvc3M6KjoqOnJhbS10ZXN0LWFwcAoaYWNzOm9zczoq******FtLXRlc3QtYXBwLypKEDE4OTQxODk3
Njk3MjIyODNSBTI2ODQyWg9Bc3N1bWVkUm9sZVVzZXJgAGoSMzE3NDQ2MzQ3NjU3NDI2Mjg5chJyYW10
ZXN0YXBwcmVhZG9ubHk=
 100% Error Headers:
 [('content-length', '254'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '564
A9A2A1790CF0F53C15C82'), ('date', 'Tue, 17 Nov 2015 03:08:26 GMT'), ('content-type', 'application/x
ml')]
 Error Body:
 <? xml version="1.0" encoding="UTF-8"? >
 <Error>
  <Code>AccessDenied</Code>
  <Message>Access denied by authorizer's policy.</Message>
  <RequestId>564A9A2A1790CF0F53C15C82</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
 </Error>
 Error Status:
 403
 put Failed!
```

ファイルのアップロードは失敗ししています。 原因は引き受けたロールにダウンロード権限しかないためです。

一時的な権限を使用したアップロード

これより、STS を使用してファイルをアップロードします。 手順は次のとおりです。

1. STS の一時的な認証情報を取得します。 アプリユーザーは usr001 です。

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
 $python ./sts.py AssumeRole RoleArn=acs:ram::1894xxxxxx722283:role/ramtestappwrite RoleSes
sionName=usr001 Policy='{"Version":"1","Statement":[{"Effect":"Allow","Action":["oss:PutObject"],"
Resource":["acs:oss:*:*:ram-test-app/usr001/*"]}]}' --id=oOhue******Frogv --secret=OmVwFJO3qcT
0******FhOYpg3p0KnA
 https://sts.aliyuncs.com/?SignatureVersion=1.0&Format=JSON&Timestamp=2015-11-17T03%3A16
%3A10Z&RoleArn=acs%3Aram%3A%3A1894xxxxxx722283%3Arole%2Framtestappwrite&RoleSessio
nName=usr001&AccessKeyId=oOhu******3Frogv&Policy=%7B%22Version%22%3A%221%22%2C%22St
atement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%5B%22oss%3APut
Object%22%5D%2C%22Resource%22%3A%5B%22acs%3Aoss%3A%2A%3A%2A%3Aram-test-app%2F
usr001%2F%2A%22%5D%7D%5D%7D&SignatureMethod=HMAC-SHA1&Version=2015-04-01&Signatur
e=Y0OPUoL1PrCqX4X6A3%2FJvgXuS6c%3D&Action=AssumeRole&SignatureNonce=8d0798a8-8cd9-1
1e5-9f49-008cfa5e4938
{
  "AssumedRoleUser": {
    "Arn": "acs:ram::1894xxxxxx722283:role/ramtestappwrite/usr001",
    "AssumedRoleId": "355407847660029428:usr001"
  },
  "Credentials": {
    "AccessKeyId": "STS.rtfx13******NllJlS4U",
    "AccessKeySecret": "2fsaM8E2maB2dn******wpsKTyK4ajo7TxFr0zIM",
    "Expiration": "2015-11-17T04:16:10Z",
    "SecurityToken": "CAESkwMIARKAAUh3/Uzcg13******y0IZjGewMpg31ITxCleBFU1eO/3Sgpudid+GV
s+Olvu1vXJn******a8azKJKtzV0oKSy+mwUrxSvUSRVDntrs78CsNfWoOJUMJKjLIxdWnGi1pgxJCBzNZ2YV
/6ycTaZySSE1V6kqQ7A+GPwY******LpdGhhTVFMucnRmeDEzRFlNVWJjTmxJSmxTNFUiEjM1NTQwNzg0
NzY2MDAyOTQyOCoGdXNyMDAxMOPzoJ2RKjoGUnNhTUQ1QnYKATEacQoFQWxsb3cSJwoMQWN0aW9
uRXF1YWxzEgZBY3Rpb24aDwoNb3NzOlB1dE9iamVjdBI/Cg5SZXNvdXJjZUVxdWFscxIlUmVzb3VyY2UaI
wohYWNzOm9zczoqOio6cmFtLXRlc3Qt******VzcjAwMS8qShAxODk0MTg5NzY5NzIyMjgzUgUyNjg0Mlo
PQXNzdW1lZFJvbGVVc2VyYABqEjM1NTQwNzg0NzY2MDAyOTQyOHIPcmFtdGVzdGFwcHdyaXRl  100%"
  },
  "RequestId": "19407707-54B2-41AD-AAF0-FE87E8870B0D"
}
```

2. アップロードとダウンロードに認証情報を使用できるかどうかをテストします。

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
 $./osscmd get oss://ram-test-app/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i STS.r
tfx13******NllJlS4U -k 2fsaM8E2maB2dn******wpsKTyK4ajo7TxFr0zIM --sts_token=CAESkwMIARKAAU
h3/Uzcg13******y0IZjGewMpg31ITxCleBFU1eO/3Sgpudid+GVs+Olvu1vXJn******a8azKJKtzV0oKSy+mw
UrxSvUSRVDntrs78CsNfWoOJUMJKjLIxdWnGi1pgxJCBzNZ2YV/6ycTaZySSE1V6kqQ7A+GPwY******Lpd
GhhTVFMucnRmeDEzRFlNVWJjTmxJSmxTNFUiEjM1NTQwNzg0NzY2MDAyOTQyOCoGdXNyMDAxMOPzo
```

```
J2RKJoGUnNhTUQ1QnYKATEacQoFQWxsb3cSJwoMQWN0aW9uRXF1YWxzEgZBY3Rpb24aDwoNb3NzOlB1dE9iamVjdBI/Cg5SZXNvdXJjZUVxdWFscxIlUmVzb3VyY2UaIwohYWNzOm9zczoqOio6cmFtLXRlc3Qt******VzcjAwMS8qShAxODk0MTg5NzY5NzIyMjgzUgUyNjg0MloPQXNzdW1lZFJvbGVVc2VyYABqEjM1NTQwNzg0NzY2MDAyOTQyOHIPcmFtdGVzdGFwcHdyaXRl  100%
Error Headers:
[('content-length', '254'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '564A9C31FFFC811F24B6E7E3'), ('date', 'Tue, 17 Nov 2015 03:17:05 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>Access denied by authorizer's policy.</Message>
  <RequestId>564A9C31FFFC811F24B6E7E3</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
get Failed!
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn-hangzhou.aliyuncs.com -i STS.rtfx13******NllJlS4U -k 2fsaM8E2maB2dn******wpsKTyK4ajo7TxFr0zIM --sts_token=CAESkwMIARKAAUh3/Uzcg13******y0IZjGewMpg31ITxCleBFU1eO/3Sgpudid+GVs+Olvu1vXJn******a8azKJKtzV0oKSy+mwUrxSvUSRVDntrs78CsNfWoOJUMJKjLIxdWnGi1pgxJCBzNZ2YV/6ycTaZySSE1V6kqQ7A+GPwY******LpdGhhTVFMucnRmeDEzRFlNVWJjTmxJSmxTNFUiEjM1NTQwNzg0NzY2MDAyOTQyOCoGdXNyMDAxMOPzoJ2RKjoGUnNhTUQ1QnYKATEacQoFQWxsb3cSJwoMQWN0aW9uRXF1YWxzEgZBY3Rpb24aDwoNb3NzOlB1dE9iamVjdBI/Cg5SZXNvdXJjZUVxdWFscxIlUmVzb3VyY2UaIwohYWNzOm9zczoqOio6cmFtLXRlc3Qt******VzcjAwMS8qShAxODk0MTg5NzY5NzIyMjgzUgUyNjg0MloPQXNzdW1lZFJvbGVVc2VyYABqEjM1NTQwNzg0NzY2MDAyOTQyOHIPcmFtdGVzdGFwcHdyaXRl  100%
 100% Error Headers:
[('content-length', '254'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '564A9C3FB8DE437A91B16772'), ('date', 'Tue, 17 Nov 2015 03:17:19 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>Access denied by authorizer's policy.</Message>
  <RequestId>564A9C3FB8DE437A91B16772</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

```
Error Status:
403
put Failed!
```

test.txt のアップロードは失敗しています。 本ページの冒頭で説明した、入力済みのポリシーは、次のようにフォーマットされています。

```
{
   "Version": "1",
     "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "oss:PutObject"
         ],
       "Resource": [
         "acs:oss:*:*:ram-test-app/usr001/*"
         ]
     }
   ]
}
```

このポリシーは、ユーザーが "usr001/" のようなファイルを ram-test-app バケットにアップロードすることのみ許可されていることを示します。 アプリユーザーが usr002 の場合、ポリシーでは "usr002/" のようなファイルのアップロードのみを許可するように変更することができます。 各アプリユーザーにそれぞれ異なるポリシーを設定することで、アプリユーザーごとのストレージ容量を分けることができます。

3. テストを再試行し、アップロード先を ram-test-app/usr001/test.txt として指定します。

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
 $./osscmd put test.txt oss://ram-test-app/usr001/test.txt --host=oss-cn-hangzhou.aliyuncs.com
-i STS.rtfx13******NllJlS4U -k 2fsaM8E2maB2dn******wpsKTyK4ajo7TxFr0zIM --sts_token=CAESkwMIA
RKAAUh3/Uzcg13******y0IZjGewMpg31ITxCleBFU1eO/3Sgpudid+GVs+Olvu1vXJn******a8azKJKtzV0oK
Sy+mwUrxSvUSRVDntrs78CsNfWoOJUMJKjLIxdWnGi1pgxJCBzNZ2YV/6ycTaZySSE1V6kqQ7A+GPwY***
***LpdGhhTVFMucnRmeDEzRFlNVWJjTmxJSmxTNFUiEjM1NTQwNzg0NzY2MDAyOTQyOCoGdXNyMDAx
MOPzoJ2RKjoGUnNhTUQ1QnYKATEacQoFQWxsb3cSJwoMQWN0aW9uRXF1YWxzEgZBY3Rpb24aDwoN
b3NzOlB1dE9iamVjdBI/Cg5SZXNvdXJjZUVxdWFscxIlUmVzb3VyY2UaIwohYWNzOm9zczoqOio6cmFtLX
Rlc3Qt******VzcjAwMS8qShAxODk0MTg5NzY5NzIyMjgzUgUyNjg0MloPQXNzdW1lZFJvbGVVc2VyYABqE
jM1NTQwNzg0NzY2MDAyOTQyOHIPcmFtdGVzdGFwcHdyaXRl  100%
 100%
 Object URL is: http://ram-test-app.oss-cn-hangzhou.aliyuncs.com/usr001%2Ftest.txt
 Object abstract path is: oss://ram-test-app/usr001/test.txt
 ETag is "946A0A1AC8245696B9C6A6F35942690B"
 0.071(s) elapsed
```

アップロードは成功しています。

まとめ

このセクションでは、STS を使用して OSS に対する一時的なアクセス許可をユーザーに付与する方法について説明しています。 一般的なモバイル開発のシナリオでは、多数のアプリユーザーがアプリにアクセスする必要がある場合、STS を使用することで OSS にアクセスするための一時的な許可を付与することができます。 一時的な許可では、権限の漏洩によって引き起こされるリスクを大幅に減らすため、有効期限を設定することができます。 一時的な許可を取得する場合、アプリのユーザーごとに異なる権限付与ポリシーを入力することで、アクセス許可を制限することができます。 たとえば、ユーザーがアクセス可能なオブジェクトパスの制限があります。 オブジェクトパスに制限をかけることにより、アプリユーザーごとのストレージスペースを分けることができます。

11.2.5. サブアカウントの設定についてのよくあるご質問

STS の一時的なアカウントはどのように作成すればよいですか。また、STS の一時的なアカウントを使用してリソースにアクセスするにはどうすればよいですか。

「STS の一時的なアクセス許可」をご参照ください。

権限付与されたサブアカウントに対して、クライアントエラー、もしくはコンソールのログインエラーが報告されました。

次のページをご参照ください。「OSS コンソールでバケットの操作権限が付与された後、サブアカウントに "OSS コンソールのバケットに対する操作権限がありません。" というエラーが発生するのはなぜですか。」

サブアカウントに対して、単一のバケットに対する操作権限を付与するにはどうすればよいですか。

「指定したバケットに対して、完全な操作権限をサブアカウントに割り当てる方法」をご参照ください。

サブアカウントに対して、バケット内のディレクトリに対する操作権限を付与するにはどうすればよいですか。

「OSS ディレクトリの権限付与」をご参照ください。

サブアカウントに対して、バケットの読み取り専用権限を付与するにはどうすればよいですか。

「サブユーザーにバケット内のリソースの一覧表示と読み取りを許可する」をご参照ください。

OSS SDK 呼び出し時にエラー InvalidAccessKeyId が表示されました。

「STS エラーとトラブルシューティング」をご参照ください。

STS 呼び出し時に次のエラーが表示されました。"権限付与者のポリシーにより、アクセスが拒否されました。"

詳細なエラー情報 "ErrorCode: AccessDenied ErrorMessage: 権限付与者のポリシーにより、アクセスが拒否されました。"

エラーの原因は以下のとおりです。

- 一時的なアカウントにアクセス権限がない
- 一時的なアカウントのロールを引き受ける指定をした許可ポリシーは、そのアカウントに対してアクセス権限を割り当てていない

その他の STS エラーとその原因については、「OSS 権限エラーとトラブルシューティング」をご参照ください。

11.3. 一時的なアクセス許可

STS の概要

OSS では、一時的なアクセス許可を Alibaba Cloud STS (Security Token Service) を使用して付与することができます。 Alibaba Cloud STS は、一時的なアクセストークンをクラウドコンピューティングユーザーに提供する Web サービスです。 STS を使用すると、 サードパーティ製アプリケーションまたはフェデレーションユーザー (ユーザー ID を管理できます) に対して、権限と有効期間をカスタマイズしたアクセス資格情報を付与することができます。 サードパーティ製アプリケーションまたはフェデレーションユーザーは、これらのアクセス資格情報を使用して、Alibaba Cloud プロダクトの API を呼び出したり、Alibaba Cloud プロダクト提供の SDK を使用してクラウドプロダクトの API にアクセスすることができます。

- サードパーティ製アプリケーションに長期キー (AccessKey) を公開する必要はありません。アクセストークンを生成してサードパーティ製アプリケーションに送信するだけで済みます。
- アクセス権限とトークンの有効期間をカスタマイズできます。
- 権限が失効する問題を気にする必要はありません。 アクセス資格情報は、有効期限が切れると自動的に無効になります。

アプリを例として、相互のプロセスを次に示します。

ソリューションの詳細を次に示します。

1. アプリユーザーとしてログインします。

   アプリユーザーの ID はカスタマーが管理します。 ID 管理システムをカスタマイズしたり、外部の Web アカウントや OpenID を使用することもできます。 AppServer で、有効なアプリユーザーごとに最小のアクセス権限を細かく定義できます。

2. AppServer から STS にセキュリティトークン (SecurityToken) をリクエストします。

   STS を呼び出す前に、アプリユーザーの最小のアクセス権限 (ポリシー構文で記述) と許可の有効期限を決定する必要があります。 次に STS の AssumeRole インターフェイスを呼び出して、セキュリティトークンを取得します。

3. STS から AppServer に有効なアクセス資格情報が返されます。このアクセス資格情報には、セキュリティトークン、一時的な AccessKey (AccessKeyId と AccessKeySecret)、有効期限が含まれます。

4. AppServer から ClientApp にアクセス資格情報が返されます。

   ClientApp は、この資格情報をキャッシュします。 資格情報が無効になった場合、AppServer に対して新しいアクセス資格情報をリクエストする必要があります。 たとえば、アクセス資格情報の有効期間が 1 時間の場合、AppServer に対して 30 分ごとにアクセス資格情報の更新をリクエストできます。

5. ClientApp は、ローカルにキャッシュしたアクセス資格情報を使用して、Alibaba Cloud サービスの API をリクエストします。 クラウドサービスは STS アクセス資格情報を認識し、STS を使用して資格情報を検証し、ユーザーのリクエストに適切にレスポンスします。

STS セキュリティトークンの詳細は、RAM ユーザーガイドの「ロール管理」をご参照ください。

STS インターフェイスの AssumeRole を呼び出して、有効なアクセス資格情報を取得します。 STS SDK を使用して、このメソッドを呼び出すこともできます。

STS 資格情報を使用した署名付きリクエストの作成

ユーザーのクライアントは、STS 一時資格情報を取得した後、 その資格情報のセキュリティ トークン (SecurityToken) と一時的な AccessKey (AccessKeyId と AccessKeySecret) を使用して署名を作成します。 アクセス許可の署名を作成する方法は、 ルートアカウントの AccessKey を使用して 署名を ヘッダーに追加する方法と基本的に同じです。 次の 2 つの点に注意してください。

- ユーザーが使用する署名キーは、STS が提供する一時的な AccessKey (AccessKeyId と AccessKeySecret) です。
- セキュリティトークン (SecurityToken) をリクエストヘッダー内に含めるか、 リクエストパラメーターとして URI に含める必要があります。 この 2 つの方法を 同時に使用することはできません。 両方の方法を同時に選択すると、InvalidArgument エラーが返されます。
  - ヘッダー `x-oss-security-token: SecurityToken` を リクエストヘッダーに含めます。 署名の CanonicalizedOSSHeaders を計算するとき、x-oss-security-token 考慮されます。
  - パラメーター `security-token=SecurityToken` を URL に含めます。 署名の CanonicalizedResource を計算するとき、security-token が考慮され、 サブリソースと見なされます。

12.OSS へのアクセス

12.1. クイックスタート

本ページでは、バケットの作成、オブジェクトのアップロードとダウンロードなどの基本的な OSS の操作方法について説明します。

1. OSS コンソールにログインします。
2. バケットを作成します。
3. ファイルのアップロードとダウンロードを実行します。

詳細は、「Alibaba Cloud OSS 利用ガイド (Get started with Alibaba Cloud OSS)」をご参照ください。

OSS のアップロードとダウンロードに関する基礎知識

OSS SDK を使用する前に、OSS のアップロードとダウンロード方法について基礎知識を身に着けることを推奨します。

OSS では RESTful API を使用して操作を実行します。また、全てのリクエストは標準の HTTP 形式のリクエストになります。

OSS は、多様な要件を満たすために、さまざまなアップロード方法を提供しています。 サポート内容は以下のとおりです。

- Put Object メソッドを使用して、5 GB 未満の単一ファイルを OSS にアップロードします。 詳細は、「簡易アップロード (Simple upload)」をご参照ください。
- ブラウザから OSS に 5 GB 未満のファイルをアップロードするには、Post Object メソッド (HTTP 形式) を使用します。 詳細は、「フォームアップロード (Form upload)」をご参照ください。
- 5 GB を超えるファイルをアップロードするには、マルチパートアップロード方式を使用します。 詳細は、「マルチパートアップロード (Multipart upload)」をご参照ください。
- オブジェクトの最後にコンテンツを直接追加するには、 Append Object メソッドを使用します。 この方法は、ビデオのモニタリングや ビデオのライブ配信に特に適しています。 詳細は、「オブジェクトの追加 (Append object)」をご参照ください。

OSS は、さまざまなダウンロード方法も提供します。 詳細は、「簡易ダウンロード (Simple download) 」および「マルチパートダウンロード (Multipart download)」をご参照ください。

OSS SDK を使用する場合の一般的なプロセス

1. コンソールから AccessKeyId と AccessKeySecret を取得します。
2. GitHub から好みのプログラミング言語で OSS SDK をダウンロードします。
3. アップロードやダウンロードなどの操作を実行します。

各種プログラミング向けの OSS SDK の使用方法についての詳細は、「OSS SDKリファレンス (OSS SDK Reference)」をご参照ください 。

12.2. OSS ベースのアプリ開発

開発シーケンス図

一般的な OSS ベースのアプリ開発には、次の 4 つの要素があります。

- OSS: アップロード、ダウンロード、アップロードコールバックなどの機能を提供します。

- 開発者のモバイルクライアント (クライアントとも呼ばれるモバイルアプリケーションや Web アプリケーション): 開発者が提供するサービスを使用して OSS にアクセスします。
- アプリケーションサーバー: クライアントと対話します。 このサーバーは開発者のサービスに使用されます。
- Alibaba Cloud STS: 一時的な認証情報を発行します。

ベストプラクティス

- モバイルアプリの直接データ転送を設定
- モバイルアプリのデータコールバックを設定
- 権限管理

サービス開発プロセス

- 一時的な認証情報を使用したデータのアップロード

  以下は、一時的な認証情報を使用したデータのアップロードプロセスを示しています。

  プロセスの説明は次のとおりです。

  i. クライアントからアプリケーションサーバに対して、OSS へのデータアップロードリクエストが送信されます。
  ii. アプリケーションサーバから STS にリクエストが送信されます。
  iii. 一時的な認証情報 (STS AccessKey とトークン) が、 STS からアプリケーションサーバーに返されます。
  iv. クライアントで認証情報 (STS AccessKey とトークン) を取得すると、モバイルクライアント SDK を呼び出して、データを OSS にアップロードします。
  v. クライアントでデータが OSS に正常にアップロードされます。 コールバックが設定されていなければ、プロセスは完了です。 コールバックが設定されている場合、OSS で関連するインタフェースが呼び出されます。

  注記:

  - クライアントからアップロードを試行するたびに、アプリケーションサーバーに承認をリクエストする必要はありません。 認証が取得されるたびに、クライアントではSTS から返された一時的な認証情報は、期限切れになるまでキャッシュに保存されます。
  - STS はアップロード向けのきめ細かいアクセス制御機能が備わっているため、オブジェクトレベルでクライアントのアクセス許可を制限できます。 これにより、異なるクライアントによって OSS にアップロードされたオブジェクトが完全に分離されるため、アプリケーションのセキュリティが大幅に向上します。

  詳細は、「認証済みの第三者によるアップロード (Authorized third-party upload)」をご参照ください。

- 署名付き URL またはフォーム認証を使用したデータアップロード

  下図は、署名付き URL またはフォーム認証によるデータアップロードのプロセスを示しています。

  プロセスの説明は次のとおりです。

  i. クライアントからアプリケーションサーバに対して、OSS へのデータアップロードリクエストが送信されます。
  ii. 認証情報 (署名付き URL またはフォーム) がアプリケーションサーバーからクライアントに返されます。

iii. クライアント側で承認 (署名付き URL またはフォーム) を取得するると、モバイルクライアント SDK を呼び出してデータをアップロードするか、フォームアップロードを使用してデータを OSS に直接アップロードします。

iv. クライアントでデータが OSS に正常にアップロードされます。 コールバックが設定されていなければ、プロセスは完了です。 コールバックが設定されている場合、OSS で関連するインタフェースが呼び出されます。

詳細は、「認証済みの第三者によるアップロード (Authorized third-party upload)」をご参照ください。

- 一時的な認証情報を使用したデータのダウンロード

  一時的な認証情報によるデータダウンロードのプロセスは、一時的な認証情報によるデータダウンロードのプロセスと似ています。

  i. クライアントからアプリケーションサーバに対して、OSS へのデータダウンロードリクエストが送信されます。

  ii. アプリケーションサーバーから STS にリクエストが送信され、一時的な認証情報 (STS AccessKey とトークン) が取得されます。

  iii. 一時的な認証情報 (STS AccessKey とトークン) が、アプリケーションサーバーからクライアントに返されます。

  iv. クライアントで認証情報 (STS AccessKey とトークン) を取得すると、モバイルクライアント SDK を呼び出して、データを OSS からダウンロードします。

  v. クライアントでデータが OSS に正常にダウンロードされます。

  注記:

  - ダウンロードでは、一時的な認証情報のキャッシュがクライアントに保存されるため、アクセス速度が向上します。
  - STS はダウンロード向けのきめ細かいアクセス制御機能を備えています。 アップロードとダウンロードに対するアクセス制御機能は、各モバイルクライアントの OSS ストレージスペースを分離するのに役立ちます。

- 署名付き URL を使用したデータのダウンロード

  署名付き URL 認証によるダウンロードプロセスは、署名付き URL 認証によるアップロードプロセスと似ています。

  i. クライアントからアプリケーションサーバに対して、OSS へのデータのダウンロードリクエストが送信されます。

  ii. 署名付き URL が、アプリケーションサーバーからクライアントに返されます。

  iii. クライアント側で認証（署名付き URL）を取得すると、モバイルクライアント SDK を呼び出して OSS からデータをダウンロードします。

  iv. クライアントでデータが OSS に正常にダウンロードされます。

  > 説明 クライアント側で開発者の AccessKey を保存することはできません。 アプリケーションサーバーによって署名された URL、または STS で発行された一時的な認証情報 STS とトークンの AccessKey) のみを取得できます。

ベストプラクティス

- RAM と STS の概要

参照情報

- Android SDK：オブジェクトのアップロード (Upload objects)
- iOS SDK：オブジェクトのアップロード (Upload objects)