Object Storage Service

開発者ガイド

開発者ガイド

OSSのリソース

次の表は、OSSを完全に活用するために役立つ「マニュアル」の一覧です。

リソース	説明
Alibaba Cloud OSS開発者ガイド	OSSの基本的な概念、機能、操作手順、およびAPIとSDKの使用方法の例について説明します。
Alibaba Cloud OSSコンソールユーザガイド	OSSコンソールでサポートされるすべての操作について説明します。
Alibaba Cloud OSSベストプラクティス	OSSのアプリケーションシナリオと設定方法について説明します。
Alibaba Cloud OSS APIマニュアル	OSSでサポートされているRESTful API操作について説明し、関連する例を示します。
Alibaba Cloud OSS SDKマニュアル	主要な言語に基づくSDKの開発および関連するパラメータについて説明します。
Alibaba Cloud OSS画像処理ガイド	イメージサービスが提供するさまざまな機能について説明します。
Alibaba Cloud OSS移行ツール	移行ツールは、ローカルまたはサードパーティのクラウドに保存されたファイルをOSSに同期させることができます。

OSS の概要

ここでは、OSS プロダクトの理解を深めるために、その基本概念を説明します。

オブジェクト

オブジェクト（ファイルとも呼ばれます）は、個別のデータ単位です。

オブジェクトは：

- メタデータ (オブジェクトメタ)は、最終変更時刻やサイズなどのオブジェクトの属性とユーザー定義の情報を表すキー値の組み合わせです。
- ユーザーデータ (データ)
- ユニークなオブジェクト名 (キー)

オブジェクトサイズは、アップロード方法によって異なります。マルチパートアップロードでは、48.8 TB までのオブジェクトがサポートされます。それ以外のアップロード方法でサポートされる最大サイズは、5 GB です。

オブジェクトのライフサイクルは、オブジェクトが正しくアップロードされたときに始まり、削除されたときに終了します。ライフサイクルの途中でオブジェクト情報を変更することはできません。同じ名前のオブジェクトを複数回アップロードすると、既存のオブジェクトは上書きされます。したがって、ファイルシステムとは異なり、OSS ではオブジェクト/ファイルを部分的に変更することはできません。

OSS の Append アップロード機能を使用すると、オブジェクトの末尾にデータをつなげて付加することができます。

オブジェクト命名規則:

- UTF-8 エンコーディングを使用します。
- 長さは 1 ～ 1023 バイトでなければなりません。
- “/ “ または “\” で開始することはできません。

注意: オブジェクト名では、大文字小文字が区別されます。

バケット

バケットは、ファイルシステムとは異なり、オブジェクトをフラット構造で管理するオブジェクトストアの仮想部分です。

バケットのプロパティは次のとおりです。:

- すべてのオブジェクトはバケットに属していなければならず、オブジェクトのライフサイクル中は、対応するバケットに直接所属しています。
- ユーザーは複数のバケットを持つことができ、各バケットには無制限の数のオブジェクトを含めることができます。
- リージョンとオブジェクトのアクセス制御とオブジェクトライフサイクル管理するためにバケットの属性を設定および変更できます。 これらの属性は、バケット内のすべてのオブジェクトに適用されます。
- 複数のバケットを作成して、それぞれ異なる管理機能を実行することができます。

バケット命名規則:

- バケット名には、小文字のアルファベット、数字、ハイフン (-) のみを使用できます。
- 先頭の文字は小文字のアルファベットまたは数字である必要があります。

- 長さは 3 ~ 63 バイトでなければなりません。
- OSS内でグローバルに唯一必要があります。

一度バケット名が作成されると、変更することはできません。

リージョン

リージョンは、OSS データセンターが物理的に配置される地域です。

データストレージのリージョンを料金、リクエストソースなどの要素に基づいて選ぶことができます。一般的に、リージョンに近いほどアクセス速度は向上します。詳細については、OSS のリージョンとエンドポイントを参照してください。

リージョンはバケットの作成時に指定され、その後は変更できません。1 つのバケット内のすべてのオブジェクトは、対応するデータセンターに格納されます。現在、オブジェクト単位でのリージョンの設定はサポートされていません。

エンドポイント (アクセスドメイン名)

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 を通じて外部にサービスを提供しています。リージョンが異なると、使用されるドメイン名も異なります。イントラネットまたはインターネットを通じて同じリージョンにアクセスするためには、別のエンドポイントが使用されます。たとえば、杭州リージョンに関して:

- インターネットエンドポイントが oss-cn-hangzhou-internal.aliyuncs.com です。
- イントラネットエンドポイントが oss-cn-hangzhou.aliyuncs.comです。

詳細については、OSS のリージョンとエンドポイントを参照してください。

AccessKey

AccessKey (AK) は、アクセス時の ID 検証に使用される AccessKeyId と AccessKeySecret のペアを意味します。

OSS は、AccessKeyId と AccessKeySecret の対称暗号化方式を使用してリクエストの送信者の ID を検証します。AccessKeyId はユーザーを識別する情報です。AccessKeySecret は、ユーザーによる署名文字列の暗号化と、OSS による署名文字列の AccessKey の検証に使用されます。AccessKeySecret は、機密として扱う必要があります。OSS では、AccessKeys は次のソースから取得されます。

- バケットのオーナーによって適用された AccessKey
- 許可されたサードパーティのリクエスト送信者に対して、RAM を通じてバケットのオーナーによって付与された AccessKey
- 許可されたサードパーティのリクエスト送信者に対して、STS を通じてバケットのオーナーによって付与された AccessKey

AccessKeysの詳細については、RAMを参照してください。

強力な整合性

OSS では、オブジェクト操作はアトミックです。操作に途中のステータスは存在せず、成功か失敗のいずれかです。ユーザーがオブジェクトをアップロードすると、OSS によって確実に完了されます。OSS からは、部分的なオブジェクトについてアップロード成功の応答は返されません。

OSS でのオブジェクト操作も、同様に強力な整合性を備えています。ユーザーがアップロード (PUT) の成功応答を受け取った時点で、このオブジェクトは直ちに読み取ることができ、データは 3 重の書き込みが完了しています。中間的なアップロードステータスは存在せず、リードアフターライトが行われます。それまでデータは読み取ることができません。削除操作でも同様です。ユーザーがオブジェクトを削除すると、そのオブジェクトは存在しません。

この強力な整合性機能は、ユーザーがアーキテクチャを設計する際に役立ちます。OSS のロジックは従来型のストレージデバイスのロジックと同じです。つまり、変更は直ちに反映され、最終的な整合性の問題を考慮する必要はありません。

OSS とファイルシステムの比較

OSS は、キーと値のペアという形式を使用した分散型オブジェクトストレージサービスです。オブジェクトのコンテンツを取得するには、一意のオブジェクト名 (キー) を使用します。test1/test.jpg のような名前を使用することもできますが、これはオブジェクトが test1 という名前のディレクトリに保存されることを意味しません。OSS では、test1/test.jpg は単なる文字列であり、a.jpg と本質的な違いはありません。したがって、どちらの名前のオブジェクトにアクセスするときも、消費されるリソースの量に違いはありません。

ファイルシステムは、一般的なツリー上のインデックス構造を使用しています。test1/test.jpg という名前のオブジェクトにアクセスするには、クライアントはまず test1 ディレクトリにアクセスしてから、そのディレクトリ内で test.jpg というファイルを探します。これにより、ディレクトリの名前変更、ディレクトリの削除、ディレクトリの移動などのフォルダー操作がディレクトリノードだけの操作になり、ファイルシステムにとって対応しやすくなります。この構造では、アクセスするディレクトリのレベルが増えると消費されるリソースの量が増え、多数のファイルが存在するディレクトリに関係する操作に時間がかかります。

OSS にはほぼ同じ機能をシミュレートできる操作がいくつかありますが、料金が発生します。たとえば、test1 ディレクトリの名前を test2 に変更する場合、OSS での実際の操作は、test1/ で始まるすべてのオブジェクトを test2/ で始まるコピーに置き換えることです。この操作には、大量のリソースが消費されます。したがって、OSS を使用しているときは、できるだけこのような操作を避けることが必要です。

OSS では、保存されているオブジェクトの変更はサポートされません (Append Object 操作の場合は、特定の API を呼び出す必要があり、生成されるオブジェクトは通常のアップロードされたオブジェクトとはタイプが異なります)。たとえ 1 バイトでも変更するには、オブジェクト全体を再度アップロードする必要があります。ファイルシステムではファイルを変更することができます。たとえば、特定の位置のコンテンツを変更したり、オブジェクトの末尾部分を切り捨てたりすることができます。これらの機能により、ファイルシステムには幅広い応用範囲が生まれます。一方、OSS は同時に大量のアクセスに対応することができますが、ファイルシステムは単一のデバイスのパフォーマンスによる制約を受けます。

したがって、OSS をオブジェクトシステムにマップすることは非常に効率が悪く、お勧めしません。オブジェクトシステムをマウントすることが必要な場合は、操作を新しいファイルの書き込み、ファイルの削除、ファイルの読み取りに限定するように注意します。OSS を使用する際は、その利点を全面的に利用します。具体的には、大量のデータ処理能力により、イメージ、ビデオ、ドキュメントなどの構造化されていない大量のデータを保管します。

OSS とファイルシステムの概念の比較

OSS	ファイルシステム
オブジェクト	ファイル
バケット	メインディレクトリ
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	メインディレクトリのリストを取得
GetBucket	ファイルのリストを取得
PutObject	オブジェクトを書き込み
AppendObject	オブジェクトを付加書き込み
GetObject	オブジェクトを読み取り
DeleteObject	オブジェクトを削除
N/A	ファイルの内容を変更
CopyObject (ターゲットとソースが同じ)	ファイル属性を変更
CopyObject	オブジェクトをコピー
該当なし	オブジェクトの名前を変更

OSS 用語集

用語	定義
バケット	ストレージ容量
オブジェクト	オブジェクトまたはファイル
エンドポイント	OSS アクセスのためのドメイン名
リージョン	地域またはデータセンター
AccessKey	AccessKeyId と AccessKeySecret のペアの別名
Put Object	簡易アップロード
Post Object	フォームアップロード

Multipart Upload	複数パートでのアップロード
Append Object	データを付加するアップロード
Get Object	簡易ダウンロード
Callback	アップロードコールバック
オブジェクトメタ	長さやタイプなど、ファイルについて記述するファイルのメタデータ
データ	ファイルのデータ
キー	ファイル名
ACL (アクセス制御リスト)	バケットまたはファイルに対する権限

ストレージクラス

ストレージクラスの概要

OSS には、標準、低頻度アクセス (IA) 、およびアーカイブという 3 つのストレージクラスがあり、さまざまなホットデータとコールドデータストレージのシナリオに適用されます。

- 標準クラスは、共通のオブジェクトストレージサービスを提供します。これは、オーディオやビデオ、写真、およびWiFi ホットスポットを介して頻繁にアクセスされるウェブサイトの静的リソースの保管に適しています。高スループットのコンピューティングシナリオをサポートし、コンピューティングリソースのストレージに適しています。
- IA クラスは、長期間保存され、頻繁にアクセスされず、モバイルアプリケーション、スマートデバイス、エンタープライズデータのバックアップに適用されるデータに適しています。リアルタイムデータアクセスをサポートしています。
- アーカイブクラスは、3 つのストレージクラスの中で最も低い単価です。アーカイブデータ、医用画像、科学データ、ビデオ素材を長期間保管し、長期保管コストを効果的に最適化するのに適しています。アーカイブクラスに格納されているデータを読み取り可能な状態に復元するには 1 分かかります。

標準

標準クラスは、高い信頼性、可用性、パフォーマンスを備えた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 の高スループットと低遅延のサービス応答機能は、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。標準クラスは、ソーシャル、画像共有、オーディオおよびビデオアプリケーション、大規模サイト、および大規模なデータ分析シナリオに適した選択肢です。

主な機能：

- 99.99999999% までのデータ信頼性
- 最大 99.95% のサービス可用性
- 高スループットと低遅延アクセス性能
- HTTPS 暗号化伝送
- 画像処理

IA

IA クラスは、頻繁にアクセスされることがない長期間保存されるデータには適しています。その単価は標準クラスの単価よりも低いです。モバイルアプリケーション、スマートデバイス、エンタープライズデータの長期的なバックアップにも適用できます。IA クラスのオブジェクトには最小の保存期間があります。保存期間が 30 日未満のファイルを削除または上書きすると、コストがかかります。IA クラスのオブジェクトには最小の保存領域があります。サイズが 128 KB 未満のオブジェクトの場合、最小保存領域は 128 KB です。データの取得にはコストがかかります。

主な機能：

- 99.99999999% までのデータ信頼性
- 最大 99.9% のサービス可用性
- リアルタイムアクセス
- HTTPS 暗号化伝送
- 画像処理
- 最小保存期間と最小保存領域

アーカイブ

アーカイブクラスの単価は、3 つのストレージクラスの中で最も安価です。アーカイブデータは長期保存（半年以上を推奨）し、保存中にアクセスすることはほとんどないと想定しています。読み取り可能な状態へのデータの復元には 1 分かかります。アーカイブデータ、医用画像、科学データ、ビデオ素材を長期間保存するのに適しています。アーカイブクラスのオブジェクトには最小の保存期間があります。保存期間が 60 日未満のファイルを削除または上書きすると、コストがかかります。アーカイブクラスのオブジェクトには最小の保存領域があります。サイズが 128 KB 未満のオブジェクトの場合、最小保存領域は 128 KB です。データの取得にはコストがかかります。

主な機能：

- 99.99999999% までのデータ信頼性
- 最大 99.9% のサービス可用性
- 保管されたデータをアーカイブ済みの状態から読み取り可能な状態に復元するのに 1 分かかる

- HTTPS 暗号化伝送
- 画像処理はサポートしない
- 最小保存期間と最小保存領域

ストレージクラスの比較

比較指標	標準	IA	アーカイブ
データの信頼性	99.99999999%	99.99999999%	99.99999999%
サービス可用性	99.95%	99.9%	99.9%
最小保存領域	オブジェクトの実際サイズ	128 KB	128 KB
最小保存期間	なし	30日間	60日間
データ取得コスト	なし	取得データ量（GB単位）に基づく	取得データ量（GB単位）に基づく
データアクセスの待ち時間	ミリ秒	ミリ秒	データ復元に 1 分必要
画像処理	サポート	サポート	サポートしない

注意： “データ取得コスト”のデータは、基礎となる分散ストレージシステムから読み取られたデータの量を指します。パブリックネットワーク上で転送されるデータ量は、アウトバウンドトラフィックの請求に含まれます。

アーカイブバケットの作成と使用

アーカイブバケットの作成と使用

OSS は 3 つのストレージタイプ を提供します。このドキュメントでは、アーカイブストレージクラスのバケットの作成および使用方法について説明します。

アーカイブバケットの作成

アーカイブバケットは、コンソール、API/SDK、コマンドラインツールから作成できます。

コンソールから作成

コンソールにアーカイブバケットを作成するには、次の図に示すように **ストレージクラス** として **アーカイブストレージ** を選択します。

API/SDK から作成

Java SDK の例：

```
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.
createBucketRequest.setCannedACL(CannedAccessControlList.PublicRead);
// Set the bucket type as Archive. The default type is Standard.
createBucketRequest.setStorageClass(StorageClass.Archive);
ossClient.createBucket(createBucketRequest);
```

createBucketRequest.setStorageClass(StorageClass.Archive); はバケットのストレージクラスをArchiveに設定するために使われます。

OSS コマンドラインツールから作成

OSSUtil の例：

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

[bucket name] はバケット名を示します。
パラメーター --storage-class を Archive に指定してアーカイブバケットを作成します。

アーカイブバケットの使用

データのアップロード

アーカイブバケットは PUT Object/Multipart 2 つのアップロードモードをサポートしますが、APPEND 書き込みはサポートしていません。

PUT Object/Multipart に基づいて開発されたアップロード用アプリケーションは、アーカイブバケットを直接使用できます。

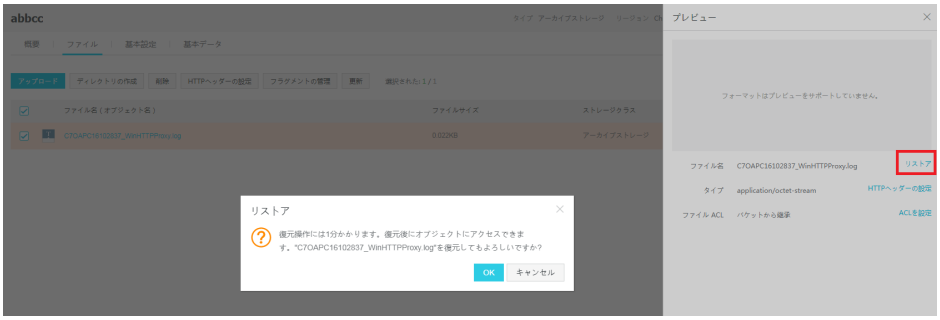
データのダウンロード

アーカイブバケットのデータ読み込みは、標準バケットと低頻度アクセスバケットとはいくつかの違いがあります。すべてのアーカイブされたデータは、読み取りの前に復元操作を介して読み取り可能な状態に復元する必要があります。復元操作には 1 分かかります。

アーカイブオブジェクトの復元操作プロセスは次のとおりです。

1. アーカイブオブジェクトは最初はフリーズ状態です。
2. 復元操作をリクエスト後、サーバーは操作を実行し、オブジェクトは復元状態になります。
3. オブジェクトは、復元後に読み取ることができます。
4. 復元された状態は、デフォルトで 1 日続きます。最大 7 日間延期することができます。この期間が終了すると、オブジェクトはフリーズ状態に戻ります。

コンソールから復元



読み込みたいオブジェクトに対して復元操作を実行します。復元操作には 1 分かかります。操作中に、オブジェクトが復元中状態になります。

API/SDK から復元

たとえば、Java SDK を使用します。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から復元

OSSUtil の例：

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

[Bucket name] と [Object name] は復元されるバケットとオブジェクトの名前です。

アクセスと制御

エンドポイント

ドメイン名の構成ルール

GetService API以外のOSSのすべてのネットワーク要求に関して、ドメイン名は指定されたバケット情報を持つ第3レベルのドメイン名です。

ドメイン名の構成はバケット名とエンドポイントの組み合わせですBucketName.Endpoint。エンドポイントは、外部から提供されるOSSのアクセスドメイン名です。OSSは、HTTP RESTful APIを介して外部にサービスを提供します。異なる地域では異なるドメイン名が使用されます。エンドポイントには、イントラネットおよびインターネットアクセスドメイン名が含まれます。たとえば、China East 1のインターネットエンドポイントは oss-cn-hangzhou.aliyuncs.comであり、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」を参照してください。

インターネット経由でOSSにアクセスする方法

インバウンドトラフィック（書き込み）は無料で、インターネットアクセスによって生成されたアウトバウンドトラフィック（読み取り）が課金されます。詳細については、「価格設定」を参照してください。

インターネット経由でOSSにアクセスするには、次の2つの方法があります。

方法1 ：アクセス中、OSSリソースはURL形式で表されます。OSS URLの構築は以下：

```
 <Schema>://<Bucket>.<Internet Endpoint>/<Object> Third-level domain name access method
Schema: Value of HTTP or HTTPS
Bucket: Your OSS storage space
Endpoint: The access domain name for a bucket's data center. Enter the Internet endpoint here.
Object: A file uploaded to the OSS.
```

例えば、中国東1で、バケットの名前はABCで、オブジェクトはmyfile/aaa.txtです。インターネットアクセスアドレスは：

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

HTMLのオブジェクトURLリンクを直接使用することができます：

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

方法2 ：OSS SDKを介してインターネットアクセスドメイン名を設定します。

OSS SDKは、各アクションのアクセスドメイン名を継承しています。ただし、異なるリージョンのバケットを操作する場合は、異なるエンドポイントを設定する必要があります。クラスターのインスタンス内にエンドポイントを設定してから、China East 1ノードでバケットを構成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);
```

イントラネットからOSSにアクセスする方法

イントラネットは、Alibaba Cloud製品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にアクセスしたり、OSSイントラネットアクセスドメイン名を設定してAlibaba Cloud CDN経由でオリジン検索を実行したりします。イントラネットで生成されるインバウンド/アウトバウンドトラフィックは無料です。詳細については、「価格設定」を参照してください。

イントラネット経由でOSSにアクセスするには、次の2つの方法があります。

方法1 ：アクセス中、OSSリソースはURL形式で表されます。OSS URLの構築は以下：

```
 <Schema>://<Bucket>.<Intranet Endpoint>/<Object> Third-level domain name access method
Schema: Value of HTTP or HTTPS
Bucket: Your OSS storage space
Endpoint: The access domain name for a bucket's data center. Enter the intranet endpoint here.
Object: A file uploaded to the OSS.
```

例えば、中国東1で、バケットの名前はABCで、オブジェクトはmyfile/aaa.txtです。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上でOSS SDKを使用してイントラネットアクセスドメイン名を設定します。

たとえば、ECSクラウドサーバー上のJava SDKでイントラネットエンドポイントは次のように設定します。

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

注意：ECSとOSSの間のイントラネットが同じリージョンではイントラネットが相互接続されるのに対し、ECSとOSSの間のイントラネットが異なるリージョンでは相互接続されません。たとえば、ECS of China North 2 (北京) を購入し、そのOSSには2つのバケットが含まれています。

> バケツの1つはbeijingresであり、領域は中国北2です。beijingres.oss-cn-beijing-internal.aliyuncs.com リージョンは、中国北2のECSのbeijingresリソースにアクセスするために使用できます。
>
> 他にはバケット qingdaoresで、リージョン中国北1、イントラネットアドレスは、qingdaores.oss-cn-qingdao-internal.aliyuncs.comは、中国北部2のECSにOSSにアクセスするために使用することはできません。OSS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 'が必要です。

OSS へのアクセス

OSSアクセスセキュリティ

OSSに送信されるHTTP要求は、ID認証情報を含む要求とID認証情報のない匿名要求の2種類に分類されます。要求のID認証情報は、次の2つの方法で構成できます。

- 許可は、要求ヘッダーにOSS + AccessKeyId +署名文字列の形式で含まれています。
- OSS AccessKeyIdおよびSignatureフィールドは、要求URLに含まれています。

OSSアクセス確認プロセス

匿名リクエストへのアクセスプロセス

ユーザーの要求は、OSSのHTTPサーバーに送信されます。

OSSはURLに基 づいてバケットとオブジェクトを解析します。

OSSは、オブジェクトがACLで設定されているかどうかをチェックします。

- オブジェクトにACLが設定されていない場合、プロセスはステップ**4**に進みます。
- オブジェクトにACLが設定されている場合、OSSはオブジェクトのACLが匿名アクセスを許可するかどうかをチェックします。
  - ACLが匿名アクセスを許可する場合、プロセスはステップ**5**に進みます。
  - そうでない場合、要求は拒否され、プロセスは終了する。

OSSはバケットのACLが匿名アクセスを許可するかどうかをチェックします。

- ACLが匿名アクセスを許可する場合、プロセスはステップ**5**に進みます。
- そうでない場合、要求は拒否され、プロセスは終了する。

要求は許可の検証に合格し、オブジェクトの内容はユーザーに返されます。

ID認証情報を持つ要求のアクセスプロセス

ユーザーの要求は、OSSのHTTPサーバーに送信されます。

OSSはURLに基 づいてバケットとオブジェクトを解析します。

要求のOSS AccessKeyIdに基づいて、OSSはリクエスタの識別情報を取得して認証を実行します。

- 情報が取得されない場合、要求は拒否され、プロセスは終了します。
- 情報が取得されたが、リクエスターがこのリソースにアクセスすることを許可されていない場合、要求は拒否され、プロセスは終了します。
- 情報が取得されたが、要求のHTTPパラメータに基づいて計算された署名が送信された署名と一致しない場合、要求は拒否され、プロセスは終了します。
- 認証が成功すると、プロセスはステップ**4**に進みます。

OSSは、オブジェクトがACLで設定されているかどうかをチェックします。

- オブジェクトにACLが設定されていない場合、プロセスはステップ**5**に進みます。
- オブジェクトにACLが設定されている場合、OSSはオブジェクトのACLが匿名アクセスを許可するかどうかをチェックします。
  - ACLが匿名アクセスを許可する場合、プロセスはステップ**6**に進みます。
  - そうでない場合、要求は拒否され、プロセスは終了します。

OSSはバケットのACLが匿名アクセスを許可するかどうかをチェックします。

- ACLが匿名アクセスを許可する場合、プロセスはステップ**6**に進みます。
- そうでない場合、要求は拒否され、プロセスは終了します。

要求は許可の検証に合格し、オブジェクトの内容はユーザーに返されます。

ID認証情報を使用したOSSアクセスの3つの方法

コンソールのOSSへのアクセス: ID認証プロセスはコンソールのユーザーから隠されています。ユーザーがコンソールからOSSにアクセスする際には、このプロセスの詳細を意識する必要はありません。

SDKを使用したOSSへのアクセス: OSSは、複数の開発言語用のSDKを提供しています。署名アルゴリズムは、ユーザがAK情報をパラメータとして入力するだけでよいSDKに実装されています。

APIを使用したOSSへのアクセス: 独自のコードを記述してRESTful APIへの呼び出しをパッケージ化する場合は、署名アルゴリズムを実装して署名を計算する必要があります。署名アルゴリズムの詳細については、APIマニュアルのヘッダーへの署名の追加およびURL への署名の追加を参照してください。

AccessKeysの説明とID認証操作の詳細については、Resource Access Management (RAM)を参照してください。

カスタムアドオンドメイン/バーチャルホスティ

ング (CNAME)

カスタムドメイン名（CNAME）をバケットにバインドできます。この操作は、OSSコンソールユーザーがAlibaba Cloudからバインドされたドメイン名とアクセス権を取得するためにICPライセンスを申請した場合にのみ利用可能です。

CNAME機能が有効になると、OSSは自動的にそのドメイン名に対するアクセス要求の処理を開始します。

CNAMEアプリケーションの例

ユーザーAには、http://img.abc.com/logo.pngというリンクを持つページを含むabc.comというWebサイトがあります。イメージimg.abc.comはOSSに移行する必要があります。

OSS Consoleを介して、ユーザAはユーザ定義ドメイン名img.abc.comをabc-imgにバインドするアプリケーションを送信し、CNAME機能承認に関連する資料を提供します。

Alibaba Cloudがアプリケーションを承認すると、OSSのバックグラウンドではimg.abc.com がabc-imgにマップされます（この間に許可の検証が実行されます）。

ユーザAは自分自身のドメインネームサーバを使用し、img.abc.com onto abc-img.oss-cn-hangzhou.aliyuncs.com. This means all access traffic to the user’s img.abc.comをabc-img.oss-cn-hangzhou.aliyuncs.comにマッピングするCNAMEルールを追加します。つまり、ユーザーの img.abc.comドメイン名へのアクセストラフィックはすべて、OSS上でabc-img.oss-cn-hangzhou.aliyuncs.comに転送されます。

http：// img.abc.com / logo.pngのリクエストがOSSに達すると、img.abc.comとabc-imgのマッピングを探し、そのリクエストをabc-imgバケットのアクセス要求に変換します。ユーザーがhttp：// img.abc.com / logo.pngにアクセスしようとすると、OSSを通過した後、アクセスされたWebサイトはhttp://abc-img.oss-cn-hangzhou.aliyuncs.com/ logo.png。

CNAMEプロセスの比較

バインドされたCNAMEなし：

http://img.abc.com/logo.pngへのアクセス要求を受け取ります。

DNSはユーザーのサーバーIPに解決されます。

ユーザのサーバ上のlogo.png へのアクセスが達成される。

バインドされたCNAMEの場合：

http://img.abc.com/logo.pngへのアクセス要求を受け取ります。

DNSはabc-img.oss-cn-hangzhou.aliyuncs.comに解決されます。

OSS バケットabc-imgのlogo.pngへのアクセスが実現しました。

参照

コンソールの操作手順については、 ドメイン名管理を参照してください。

Resource Access Management (RAM)

OSS アクセスリクエストの送信

OSS 開発者は、OSS の RESTful API を呼び出すか、API がカプセル化された SDK を使用して、OSS に直接アクセスすることができます。OSS へのアクセスのリクエストごとに、現在のバケットの権限および操作に基づいて、直接匿名アクセスまたは ID の検証が必要になります。

OSS リソースへのアクセスは、オーナーによるアクセスとサードパーティによるアクセスに分けられます。オーナーとは、バケットのオーナーを意味します。” 開発者” と呼ばれることもあります。サードパーティのユーザーとは、バケット内のリソースにアクセスするユーザーです。

アクセス方法には、匿名アクセスと署名ベースのアクセスの 2 つがあります。OSS では、識別情報を含まないリクエストは匿名アクセスと見なされます。OSS API ドキュメントの規則では、署名ベースのアクセスは、リクエストヘッダーまたは URL に署名情報を含まないリクエストを意味します。

AccessKey のタイプ

現在、OSS へのアクセスに使用できる AccessKey (AccessKey) には 3 つのタイプがあります。これらを以下に示します。

Alibaba Cloud アカウントの AccessKey

これは、バケットのオーナーの AccessKey です。各 Alibaba Cloud アカウントによって提供される AccessKey では、そのアカウント自身のリソースに対してフルアクセスすることができます。各 Alibaba Cloud アカウントには、同時に 0 ~ 5 個の AccessKey ペア (AccessKeyID と AccessKeySecret) を設定できます。コンソールにログインして、AccessKey コンソールで AccessKey ペアを追加または削除することができます。各 AccessKey ペアには、[Active] と [Inactive] の 2 つの状態があります。

- [Active] は、ユーザーの AccessKey がアクティブな状態で、ID 認証に使用できます。
- [Inactive] は、ユーザーの AccessKey が非アクティブな状態で、ID 認証に使用できません。

注意：必要な場合を除き、Alibaba Cloud アカウントの AccessKey は直接使用しないでください。

RAM アカウント AccessKey

Resource Access Management (RAM) は、Alibaba Cloud のリソースアクセス制御サービスです。RAM アカウント AK は、RAM によって付与されるアクセスキーです。この AK で行われるのは、RAM によって定義されたルールに従って、バケット内のリソースへのアクセスを許可することだけです。RAM は、ユーザー (従業員、システム、アプリケーションなど) をまとめて管理し、ユーザーがアクセスできるリソースを制御するのに役立ちます。

たとえば、ユーザーにバケットの読み取り権限のみを許可することができます。サブアカウントは、通常のアカウントの下位のアカウントで、実際のリソースを所有できません。すべてのリソースは、プライマリアカウントに属します。

STSアカウントAccessKeys

Alibaba Cloud STS (Security Token Service) は、一時的なアクセス資格情報を提供するサービスです。STSアカウントAKは、STSによって発行されたAKです。これらのAKは、STSによって定義されたルールに従ってバケットにのみアクセスできます。

ID認証の実装

現在、認証には3つの方法があります。

- AK認証
- RAM認証
- STS認証

個々のアイデンティティとしてOSSに要求を送る前に、ユーザは、OSSによって指定されたフォーマットに従って要求のための署名文字列を生成し、その後、AccessKeySecretを使用して署名文字列を暗号化して検証コードを生成する必要がある。

要求を受信した後、OSSは、AccessKeyIDに基づいて対応するAccessKeySecretを見つけ、同じ方法で署名

文字列および認証コードを取得する。取得された認証コードが提供された認証コードと同じである場合、その要求は有効であると見なされます。そうでない場合、OSSは要求を拒否し、HTTP 403エラーを返します。

ユーザーは、OSSが提供するSDKを異なるタイプのAccessKeysで直接使用して、さまざまなタイプのID認証に使用できます。

権限の制御

OSS では、格納されているオブジェクトへのアクセスに、次のさまざまな権限制御メカニズムを使用できます。

- バケットレベルの権限
- オブジェクトレベルの権限
- アカウントレベルの権限 (RAM)
- 一時的なアカウント許可 (STS)

バケットレベルの権限

バケット権限のタイプ

OSS には、権限制御のためのアクセス制御リスト (Access Control List、ACL) が用意されています。OSS の ACL は、バケットレベルのアクセス制御を提供します。現在バケットに提供されているアクセス権限は、公開読み書き、公開読み取り、非公開の 3 つです。これらを以下に示します。

権限	アクセス制限
公開読み書き	すべてのユーザー (匿名ユーザーを含む) がバケット内のオブジェクトの読み取り、書き込み、および削除を実行できます。このような操作によって発生する費用はバケットの作成者の負担になります。**この権限を使用する際には注意してください。**
公開読み取り	バケットの作成者だけがバケット内のオブジェクトの書き込みまたは削除を実行できます。すべてのユーザー (匿名ユーザーを含む) がバケット内のオブジェクトを読み取ることができます。
非公開	バケットの作成者だけがバケット内のオブジェクトの読み取り、書き込み、および削除を実行できます。その他のユーザーは権限が付与されていないと、バケット内のオブジェクトにアクセスすることはできません。

バケット権限設定と読み取り方法

機能の使用方法のリファレンス:

- API: Put BucketACL
- SDK: Java SDK-バケット ACL の設定
- コンソール: バケットの作成 権限設定
- API: Get BucketACL
- SDK: Java SDK-バケット ACL の取得

オブジェクトレベルの権限

オブジェクト権限のタイプ

OSS ACL では、オブジェクトレベルの権限によるアクセス制御も使用できます。現在オブジェクトで使用できるアクセス権限は、非公開、公開読み取り、公開読み書きおよびデフォルトの 4 つです。アクセス権限を設定するには、Put Object ACL リクエストで “x-oss-object-acl” ヘッダーを使用します。この操作を実行できるのはバケットオーナーだけです。

権限	アクセス制限
公開読み書き	オブジェクトがパブリックに読み書き可能であることを示します。つまり、すべてのユーザーがこのオブジェクトの読み取りおよび書き込みを実行できます。
公開読み取り	オブジェクトがパブリックに読み取り可能であることを示します。このオブジェクトの読み取りおよび書き込みを実行できるのは、このオブジェクトのオーナーだけです。その他のユーザーはこのオブジェクトの読み取りのみを実行できます。
非公開	オブジェクトが非公開リソースであることを示します。このオブジェクトの読み取りおよび書き込みを実行できるのは、このオブジェクトのオーナーだけです。その他のユーザーはこのオブジェクトの操作を実行できません。
デフォルト	オブジェクトがバケットの権限を継承することを示します。

考慮事項

- オブジェクトに ACL が設定されていない場合は、そのオブジェクトではデフォルトの ACL が使用されます。つまり、そのオブジェクトには、格納先のバケットと同じ ACL が設定されます。
- オブジェクトに ACL が設定されている場合は、そのオブジェクトの ACL は、バケット ACL よりも高い権限レベルになります。たとえば、公開読み取り権限が設定されたオブジェクトには、バケット権限に関係なく、認証されたユーザーと匿名ユーザーがアクセスできます。

オブジェクト権限設定と読み取り方法

機能の使用方法のリファレンス:

- API: Put Object ACL
- SDK: Java SDK-『Object ACL』の「オブジェクト ACL の設定」
- API: Get Object ACL
- SDK: Java SDK-『Object ACL』の「オブジェクト ACL の読み取り」

アカウントレベルの権限 (RAM)

シナリオ

購入したクラウドリソースを組織内の複数のユーザーが使用する必要がある場合、これらのユーザーは Alibaba Cloud アカウントの AccessKey を共有する必要があります。次の 2 つの問題があります。

多くのユーザーでキーを共有すると、漏洩のリスクが高まります。

どのリソース (バケットなど) にユーザーがアクセスできるかについて判断できません。

解決策: Alibaba Cloud アカウントでは、RAM を使用して固有の AccessKey を持つサブユーザーを作成することができます。この場合、Alibaba Cloud アカウントがプライマリアカウントとなり、作成したアカウントがサブアカウントになります。サブアカウントは、プライマリアカウントによって許可された操作とリソースに対してのみ、AccessKey を使用できます。

具体的な実装

RAM の詳細については、RAM ユーザーマニュアルを参照してください 『RAM ユーザーマニュアル』では、権限の付与方法、RAM アカウントの作成方法、およびグループ権限の管理方法を詳しく説明しています。

権限付与で必要とされるポリシーの設定方法の詳細については、この章の最後のセクションを参照してください。

一時的なアカウント許可 (STS)

シナリオ

アプリユーザー、ローカル企業アカウント、サードパーティ製のアプリなど、ローカルのIDシステムによって管理されるユーザーは、OSSリソースに直接アクセスすることもできます。これらはフェデレーションユーザーと呼ばれます。また、Alibaba Cloudのリソースにアクセスできる、作成したアプリケーションをユーザーにすることもできます。

これらの連合ユーザに関しては、アリババクラウドのセキュリティトークンサービス (STS) を介して、Alibaba Cloudアカウント (またはRAMユーザ) に短期アクセス許可管理が提供されます。Alibaba Cloudアカウント (またはRAMユーザー) の長期的なキー (ログインパスワードやAccessKeyなど) を公開する必要はありませんが、フェデレーションユーザーのために短期間のアクセス資格情報を作成するだけで

済みます。この資格情報のアクセス許可と有効性は、どちらもあなた次第です。権限の取り消しに注意する必要はありません。有効期限が切れた場合、アクセス資格は自動的に無効になります。

STSベースのアクセス資格情報には、セキュリティトークン（SecurityToken）と一時アクセスキー（AccessKeyIdとAccessKeySecret）が含まれます。AccessKeyメソッドは、Alibaba CloudアカウントまたはRAMユーザーのAccessKeyを使用する方法と同じです。さらに、各OSSアクセス要求にはセキュリティトークンが必要です。

特定の実装

STSの詳細については、RAMのユーザガイドのRoleの管理を参照してください。キーは、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することです。STS SDKを直接使用してアクセス資格情報を呼び出すこともできます。

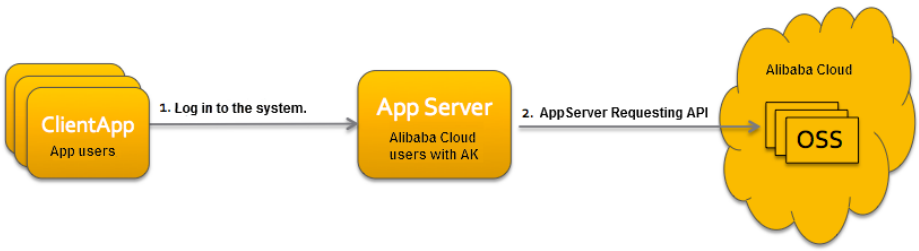
承認に必要なポリシーを設定する方法の詳細については、この章の最後のセクションを参照してください。

RAMとSTSアプリケーションシナリオの実践

異なるアプリケーションシナリオでは、アクセスIDがどのように検証されるかは異なる場合があります。以下に、典型的なアプリケーションシナリオでのアイデンティティ検証にアクセスする2つの方法について説明します。

モバイルアプリが例として使用されています。あなたはモバイルアプリの開発者であると仮定します。Alibaba Cloud OSSを使用して、アプリケーションのエンドユーザーデータを保存しようとしました。また、アプリユーザーが他のアプリユーザーのデータを取得できないように、アプリユーザー間でデータが分離されていることを確認する必要があります。

モード1：データ転送とデータ分離にAppServerを使用する

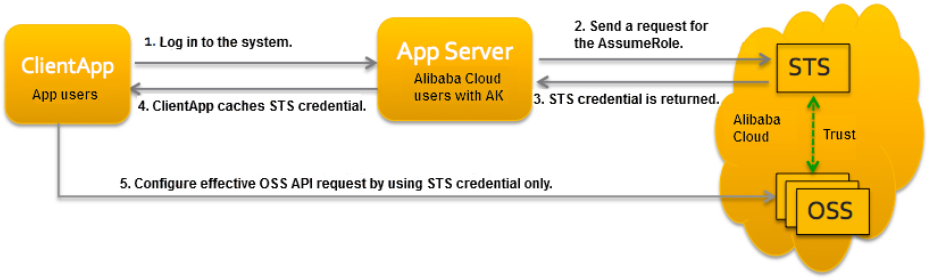


上記の図に示すように、AppServerを開発する必要があります。AppServerのみがECSにアクセスできます。ClientAppは、AppServerを介してのみデータを読み書きできます。AppServerは、異なるユーザーデータへの独立したアクセスを保証します。

この方法では、Alibaba CloudアカウントまたはRAMアカウントで提供された鍵を署名検証に使用することができます。セキュリティ上の問題が発生した場合は、Alibaba Cloudアカウントのキー（rootアカウント）を直接使用してOSSにアクセスしないでください。

モード2：STSを使用してOSSに直接アクセスする

STSソリューションを以下に示します。



手順

アプリユーザーとしてログオンします。アプリユーザーはAlibaba Cloudアカウントとは無関係ですが、アプリのエンドユーザーです。AppServerは、アプリケーションユーザーがログオンできるようにします。有効な各アプリケーションユーザに対して、AppServerはそれらのユーザに対して最小アクセス許可を定義する必要があります。

AppServerは、セキュリティトークン（SecurityToken）をSTSに要求します。STSを呼び出す前に、AppServerは、アプリユーザーの最小アクセス許可（ポリシー構文で説明）と承認の有効期限を判断する必要があります。次に、AppServerはAssumeRoleを使用してロールを示すセキュリティトークンを取得します。役割の管理と使用の詳細については、「RAMユーザーガイド」の**役割の管理**を参照してください。

STSは、有効なアクセス資格情報をAppServerに返します。アクセス資格情報には、セキュリティトークン、一時アクセスキー（AccessKeyIDとAccessKeySecret）、および有効期限が含まれます。

AppServerはアクセスクレデンシャルをClientAppに返します。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ると、ClientAppはAppServerから新しい有効なアクセス資格情報を要求する必要があります。たとえば、アクセスクレデンシャルが1時間有効である場合、ClientAppは30分ごとにアクセスクレデンシャルを更新するようにAppServerに要求できます。

ClientAppはローカルにキャッシュされたアクセス資格情報を使用してAlibaba Cloud Service APIを要求します。ECSはSTSアクセス認証情報を認識し、STSを使用して認証情報を検証し、ユーザーの要求に正しく応答します。

RAMおよびSTS認可ポリシーの設定

RAMまたはSTS認可時のポリシーの使用に関する詳細なルールは次のとおりです。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"oss:GetBucketAcl",
"oss:ListObjects"
],
"Resource": [
"acs:oss:*:1775305056529849:mybucket"
],
"Effect": "Allow",
"Condition": {
"StringEquals": {
"acs:UserAgent": "java-sdk",
"oss:Prefix": "foo"
},
"IpAddress": {
"acs:SourceIp": "192.168.0.1"
}
}
},
{
"Action": [
"oss:PutObject",
"oss:GetObject",
"oss:DeleteObject"
],
"Resource": [
"acs:oss:*:1775305056529849:mybucket/file*"
],
"Effect": "Allow",
"Condition": {
"IpAddress": {
"acs:SourceIp": "192.168.0.1"
}
}
}
]
}
```

これは認可ポリシーです。このポリシーを使用して、RAMまたはSTS経由でユーザーにアクセス許可を与えることができます。ポリシーにはStatement（1つのポリシーに複数のStatementを含めることができます）があります。Statementには、Action、Resource、Effect、およびConditionが指定されています。

このポリシーは、mybucket/file*と mybucket/file*リソースを対応するユーザーに認可し、GetBucketAcl、GetBucket、PutObject、GetObject、および DeleteObject アクションをサポートします。「条件」は、認証が成功し、許可されたユーザーが、UserAgentがjava-sdkで、送信元IPアドレスが192.168.0.1 である場合にのみ、関連リソースにアクセスできることを示します。PreButton および Delimiter 条件は GetBucket（ListObjects）アクション中に適用されます。2つのフィールドの詳細については、OSS API Documentation を参照してください。

設定ルール

Version

ポリシーのバージョンが定義されます。このドキュメントの設定方法では、” 1” に設定されます。

Statement

Statement では、権限付与の意味を記述します。これには、ビジネスのシナリオに基づいて、複数の意味を含めることができます。それぞれの意味には、Action、Effect、Resource、および Condition の説明が含まれます。リクエスト側のシステムが、各ステートメントに一致があるかどうかを 1 つずつ確認します。正しく一致したすべてのステートメントは、Effect の設定の違いに応じて、Allow および Deny に分けられます (Deny の方が優先されます)。一致がすべて Allow となる場合は、リクエストによって認証が渡されます。一致のいずれかが Deny であるか、一致がない場合は、このリクエストからのアクセスは拒否されます。

Action

Action は、バケットレベルの操作とオブジェクトレベルの操作の 2 つのカテゴリに分けられます。バケットレベルの操作には oss:PutBucketAcl や oss:GetBucketLocation があります。操作対象はバケットで、操作名は関連するインターフェイスと 1 対 1 で対応しています。オブジェクトレベルの操作には oss:GetObject、oss:PutObject、oss:DeleteObject、oss:DeleteObject、oss:AbortMultipartUpload があります。特定のタイプのオブジェクトに対する操作を許可するには、上記の操作を 1 つ以上選択します。また、上の例のように、すべての操作名に “oss:” というプレフィックスが付いている必要があります。Action はリストで、複数の操作を設定することができます。操作と API のマッピングは次のとおりです。

サーバーレベル

API	操作
GetService (ListBuckets)	oss:ListBuckets

バケットレベル

API	操作
PutBucket	oss:PutBucket
GetBucket (ListObjects)	oss:ListObjects
PutBucketAcl	oss:PutBucketAcl
DeleteBucket	oss:DeleteBucket
GetBucketLocation	oss:GetBucketLocation
GetBucketAcl	oss:GetBucketAcl
GetBucketLogging	oss:GetBucketLogging
PutBucketLogging	oss:PutBucketLogging

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

オブジェクトレベル

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

UploadPartCopy	oss:GetObject,oss:PutObject
AppendObject	oss:PutObject
GetObjectAcl	oss:GetObjectAcl
PutObjectAcl	oss:PutObjectAcl

Resource

Resource は、OSS にある 1 つ以上の特定のリソースを表します (ワイルドカードがサポートされています)。リソース名の形式は “acs:oss:region:bucket_owner:bucket_name/object_name” です。バケットレベルの操作の場合、最後の部分の “/object_name” は不要です。
” acs:oss:region:bucket_owner:bucket_name” とするだけで済みます。Resource もリストであるため、複数のリソースを設定することができます。ここでは、リージョンフィールドは現在サポートされていないため、” *” に設定されています。

Effect

Effect は、ステートメントの権限付与結果を示します。値のオプションには、Allow と Deny の 2 つがあります。ステートメントの一致が複数ある場合は、Deny が優先されます。

たとえば、特定のディレクトリの削除を拒否し、他のファイルのすべての操作を許可します。下記の例を示します。

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

Condition

Conditionは、権限付与ポリシーの条件を示します。上の例では、acs:UserAgent と acs:SourceIp のチェック条件を設定できます。oss:Delimiter フィールドと oss:Prefix フィールドは、GetBucket 操作時にリソー

スを制限するために使用されます。

OSS では次の条件がサポートされています。

条件	機能	有効な値
acs:SourceIp	IP アドレスセグメントを指定	一般的な IP アドレス、ワイルドカード (*) をサポート
acs:UserAgent	http useragent ヘッダーを指定	文字列
acs:CurrentTime	有効なアクセス時刻を指定	ISO8601 形式
acs:SecureTransport	HTTPS が使用されているかどうか	“true” または “false”
oss:Prefix	ListObjects のプレフィックスとして使用	有効なオブジェクト名

ベストプラクティス

RAMおよびSTSユーザーガイド

OSS アクセスドメイン名

クラシックネットワークのリージョンとエンドポイント

クラシックネットワークの各リージョンのインターネットとイントラネットのエンドポイントは次のとおりです。

リージョン名	リージョンの表記	インターネットエンドポイント	HTTPSのサポート(インターネット)	ECS アクセスのイントラネットエンドポイント	HTTPSのサポート(イントラネット)
杭州 (中国東部 1)	oss-cn-hangzhou	oss-cn-hangzhou.aliyuncs.com	はい	oss-cn-hangzhou-internal.aliyuncs.com	はい
上海 (中国東部 2)	oss-cn-shanghai	oss-cn-shanghai.aliyuncs.com	はい	oss-cn-shanghai-internal.aliyuncs.com	はい
青島 (中国北部 1)	oss-cn-qingdao	oss-cn-qingdao.aliyuncs.com	はい	oss-cn-qingdao-internal.aliy	はい

				uncs.com	
北京 (中国北部 2)	oss-cn-beijing	oss-cn-beijing.aliyuncs.com	はい	oss-cn-beijing-internal.aliyuncs.com	はい
張家口 (中国北部 3)	oss-cn-zhangjiakou	oss-cn-zhangjiakou.aliyuncs.com	はい	oss-cn-zhangjiakou-internal.aliyuncs.com	はい
フフホト (中国北部 5)	oss-cn-huhehaote	oss-cn-huhehaote.aliyuncs.com	はい	oss-cn-huhehaote-internal.aliyuncs.com	はい
深セン (中国南部 1)	oss-cn-shenzhen	oss-cn-shenzhen.aliyuncs.com	はい	oss-cn-shenzhen-internal.aliyuncs.com	はい
香港	oss-cn-hongkong	oss-cn-hongkong.aliyuncs.com	はい	oss-cn-hongkong-internal.aliyuncs.com	はい
シリコンバレー (米国西部 1)	oss-us-west-1	oss-us-west-1.aliyuncs.com	はい	oss-us-west-1-internal.aliyuncs.com	はい
バージニア (米国東部 1)	oss-us-east-1	oss-us-east-1.aliyuncs.com	はい	oss-us-east-1-internal.aliyuncs.com	はい
シンガポール	oss-ap-southeast-1	oss-ap-southeast-1.aliyuncs.com	はい	oss-ap-southeast-1-internal.aliyuncs.com	はい
シドニー	oss-ap-southeast-2	oss-ap-southeast-2.aliyuncs.com	はい	oss-ap-southeast-2-internal.aliyuncs.com	はい
クアラルンプール (マレーシア)	oss-ap-southeast-3	oss-ap-southeast-3.aliyuncs.com	はい	oss-ap-southeast-3-internal.aliyuncs.com	はい
東京 (日本)	oss-ap-northeast-1	oss-ap-northeast-1.aliyuncs.com	はい	oss-ap-northeast-1-internal.aliyuncs.com	はい

ムンバイ (インド)	oss-ap-south-1	oss-ap-south-1.aliyuncs.com	はい	oss-ap-south-1-internal.aliyuncs.com	はい
ドバイ	oss-me-east-1	oss-me-east-1.aliyuncs.com	はい	oss-me-east-1-internal.aliyuncs.com	はい

注意:

リンクを共有したり CNAME ドメイン名をバインドしたりするときは、” バケット名” + “エンドポイント” という形式のサードレベルドメイン名を使用することをお勧めします。たとえば、上海のバケット oss-sample のサードレベルドメイン名は、oss-sample.oss-cn-shanghai.aliyuncs.com となります。

新しい SDK のバージョン (C SDK を除く) を使用する場合、初期化パラメーターに “http:// ” または “https:// ” + “エンドポイント” という形式を使用してください。上海のエンドポイントを例に取ります。この場合 http://oss-cn-shanghai.aliyuncs.com または https://oss-cn-shanghai.aliyuncs.com となります。**. サードレベルドメイン名を初期化パラメーターに使用しないでください** (http://bucket.oss-cn-shanghai.aliyuncs.com は使用しないでください)。

ただし、以前の SDK バージョン (C、PHP、および Python の SDK など) では、エンドポイントを直接使用しています (oss-cn-shanghai.aliyuncs.com など)。お使いの SDK バージョンのドキュメントやコードに関する説明を参照してください。

元のアドレスである oss.aliyuncs.com は、デフォルトで杭州ノードのインターネットアドレスに転送されます。

元のイントラネットアドレスである oss-internal.aliyuncs.com は、デフォルトで杭州ノードのイントラネットアドレスに転送されます。

VPC ネットワークのリージョンとエンドポイント

OSS にアクセスするにあたって VPC ネットワークの ECS が使用できるエンドポイントは、以下のものだけです。

リージョン名	リージョンの表記	VPC ネットワークのエンドポイント	HTTPS のサポート
杭州 (中国東部 1)	oss-cn-hangzhou	oss-cn-hangzhou-internal.aliyuncs.com	はい

上海 (中国東部 2)	oss-cn-shanghai	oss-cn-shanghai-internal.aliyuncs.com	はい
青島 (中国北部 1)	oss-cn-qingdao	vpc100-oss-cn-qingdao.aliyuncs.com	はい
北京 (中国北部 2)	oss-cn-beijing	oss-cn-beijing-internal.aliyuncs.com	はい
張家口 (中国北部 3)	oss-cn-zhangjiakou	oss-cn-zhangjiakou-internal.aliyuncs.com	はい
フフホト (中国北部 5)	oss-cn-huhehaote	oss-cn-huhehaote-internal.aliyuncs.com	はい
深セン (中国南部 1)	oss-cn-shenzhen	oss-cn-shenzhen-internal.aliyuncs.com	はい
香港	oss-cn-hongkong	oss-cn-hongkong-internal.aliyuncs.com	はい
シリコンバレー (米国西部 1)	oss-us-west-1	oss-us-west-1-internal.aliyuncs.com	はい
バージニア (米国東部 1)	oss-us-east-1	oss-us-east-1-internal.aliyuncs.com	はい
シンガポール	oss-ap-southeast-1	vpc100-oss-ap-southeast-1.aliyuncs.com	はい
シドニー	oss-ap-southeast-2	oss-ap-southeast-2-internal.aliyuncs.com	はい
クアラルンプール (マレーシア)	oss-ap-southeast-3	oss-ap-southeast-3-internal.aliyuncs.com	はい
東京 (日本)	oss-ap-northeast-1	oss-ap-northeast-1-internal.aliyuncs.com	はい
ムンバイ (インド)	oss-ap-south-1	oss-ap-south-1-internal.aliyuncs.com	はい
ドバイ	oss-me-east-1	oss-me-east-1-internal.aliyuncs.com	はい

OSS へのアクセス

OSS ベースのアプリ開発

開発アーキテクチャ

一般的な OSS ベースのアプリ開発は 4 つのコンポーネントで構成されます。

OSS: アップロード、ダウンロード、および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 (アプリまたは Web ページアプリケーション): 開発者によって提供されるサービスを経由して間接的に OSS にアクセスします。

アプリケーションサーバー: クライアントとやり取りするサーバーです。開発者のサービス用のサーバーにもなります。

Alibaba Cloud STS: 一時的な資格情報を発行します。

サービス開発プロセス

注意:クライアントは開発者のAccessKeyを格納することはできません。アプリケーションサーバーからSTS（STSアクセスキーとトークン）によって発行された署名付きURLまたは一時的な資格情報のみを取得することができます。

一時的な資格情報のアップロード権限の付与

一時的な資格情報のアップロード権限の付与には、次の手順が使用されます。

クライアントはアプリケーションサーバーにリクエストを送信し、OSS へのオブジェクトのアップロードを要求します。

アプリケーションサーバーは STS サーバーにリクエストを送信し、一時的な資格情報を取得する必要があります。

アプリケーションサーバーはクライアントに応答し、一時的な資格情報を返します。

クライアントは OSS にアップロードする権限を取得し (STS AccessKey とトークン)、OSS の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します。

クライアントは正常にデータを OSS にアップロードします。コールバックが設定されていない場

合は、プロセスが完了します。コールバック機能が設定されている場合は、OSS は関連するインターフェイスを呼び出します。

注意:

- クライアントは、アップロードごとにアプリケーションサーバーに権限をリクエストする必要はありません。一度権限付与されたら、クライアントは有効期限が切れるまで、STS によって返された一時的な資格情報をキャッシュに保存します。
- STS では、クライアントのアクセス権限をオブジェクトレベルで制限できる強力なアクセス制御を提供します。これにより、OSS にアップロードされるオブジェクトをクライアントごとに分離し、アプリケーションのセキュリティを大幅に高めます。

詳細については「権限のあるサードパーティのアップロード」を参照してください。

アップロードおよびフォームアップロード用の署名付き URL 権限付与

クライアントはアプリケーションサーバーにリクエストを送信し、OSS へのオブジェクトのアップロードを要求します。

アプリケーションサーバーはクライアントに応答し、資格情報 (署名付き URL またはフォーム) を返します。

クライアントは OSS にアップロードする権限を取得し (署名付き URL またはフォーム)、OSS の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、フォームを直接アップロードします。

クライアントは正常にデータを OSS にアップロードします。コールバックが設定されていない場合は、プロセスが完了します。コールバック機能が設定されている場合は、OSS は関連するインターフェイスを呼び出します。

詳細については「権限のあるサードパーティのアップロード」を参照してください。

一時的な資格情報のダウンロード権限の付与

このプロセスは、一時的な資格情報のアップロード権限の付与と似ています。

クライアントは OSS からオブジェクトのダウンロードするために、アプリケーションサーバーにリクエストを送信します。

アプリケーションサーバーは STS サーバーにリクエストを送信し、一時的な資格情報を取得する

必要があります。

アプリケーションサーバーはクライアントに応答し、一時的な資格情報を返します。

クライアントは OSS からダウンロードする権限を取得し (STS AccessKey とトークン)、OSS のモバイルクライアント SDK を呼び出してデータをダウンロードします。

クライアントは正常に OSS からオブジェクトをダウンロードします。

注意:

- アップロードの場合のように、クライアントは一時的な資格情報をキャッシュに保存して、アクセス速度を高めます。
- STS でも同様に、オブジェクトのダウンロード権限を細かく制御できます。これをアップロード権限の制御と組み合わせることで、モバイルクライアントごとに OSS ストレージ容量を完全に分離できます。

ダウンロード用の署名付き URL 権限付与

これは、アップロード用の署名付き URL 権限付与に似ています。

クライアントは OSS からオブジェクトのダウンロードするために、アプリケーションサーバーにリクエストを送信します。

アプリケーションサーバーはクライアントに応答し、署名付き URL を返します。

クライアントは OSS からダウンロードする権限を取得し (署名付き URL)、OSS のモバイルクライアント SDK を呼び出してデータをダウンロードします。

クライアントは正常に OSS からオブジェクトをダウンロードします。

リファレンス:

SDK: Android SDK - オブジェクトをアップロードする

SDK: iOS SDK - オブジェクトをアップロードする

OSS クイックスタート

クイックスタート

コンソールのクイックスタート:

1. OSS コンソールにログオンし、OSS を有効化します。
2. バケットを作成します。
3. ファイルをアップロード、ダウンロードします。

詳細については、『Alibaba Cloud OSS を利用する前に』ドキュメントを参照してください。

OSS アップロードとダウンロードの簡単な概要

SDK を使い始める前に、『開発者ガイド』のアップロード機能とダウンロード機能に関するセクションを参照してください。

OSS では、RESTful API を使用して操作を実行できます。またすべてのリクエストは標準の HTTP リクエストです。

- OSS には、さまざまなファイルアップロード方法が用意されています。
  - 1 つの PUT リクエストを使用した簡易アップロード。
  - Web ページフォームを使用して直接アップロードするフォームアップロード。
  - および大きなファイルのアップロードを行う再開可能なアップロードがあります。
  - ビデオモニタリングなどの用途には、追加アップロードも提供しています。
- 同様に、OSS では、簡易ダウンロードと大きなファイル向けの再開可能なダウンロードという複数のダウンロード方法を提供しています。

SDK のクイックスタート:

1. OSS を有効化した後に、コンソールから AccessKeyId と AccessKeySecret を取得します。
2. さまざまなプログラミング言語の SDK をダウンロードします。
3. SDK のドキュメントの説明に基づいて、アップロード、ダウンロード、およびその他の操作を実行します。

詳細については、 OSS SDK 参照を参照してください。

バケットの管理

バケットの作成

バケットを作成できるのは既存のリージョン内です。次の条件を満たす必要があります。

- 各ユーザーが作成できるバケットは、最大 10 個です。
- 各バケット名はグローバルに一意である必要があります。一意でない場合は作成できません。
- バケット名は命名規則に準拠している必要があります。
- バケットの作成後に、そのバケットの名前とリージョンを変更することはできません。

OSS には、権限制御のためのアクセス制御リスト (Access Control List、ACL) が用意されています。バケットを作成するときに ACL を設定できます。バケットを作成した後に変更を行うことができます。ACL が設定されていない場合、デフォルト値は Private (非公開) です。詳細については、「バケットの読み取りおよび書き込み権限 (ACL) の設定」を参照してください。

機能の使用方法のリファレンス:

- コンソール: バケットの作成
- SDK: Java SDK - Bucketの管理
- API: Put Bucket

バケットの読み取りおよび書き込み権限 (ACL) の設定

バケットを作成するときは、バケットの所有者だけが、アクセス制御リスト（ACL）を使用してバケットの読み取りと書き込みの許可を設定できます。所有者は、サービス要件に従ってそのバケットのACLを変更することもできます。現在、バケットには3つのアクセス権があります。

権限	アクセス制限
公開読み書き	すべてのユーザー (匿名ユーザーを含む) がバケット内のファイルの読み取りおよび書き込み操作を実行できます。これらの操作によって発生する費用はバケットの作成者の負担になります。この権限を使用する際には注意してください。

公開読み取り	バケットの作成者のみがバケット内のファイルの書き込み操作を実行できます。ファイルの読み取り操作については、その他のユーザー (匿名ユーザーを含む) も実行できます。
非公開	権限を付与されたユーザーだけがバケット内のファイルの読み取り、書き込み、および削除を実行できます。その他のユーザーは権限が付与されていないと、バケット内のファイルにアクセスすることはできません。

詳細については、Access controlを参照してください。

参照:

バケットの ACL の設定

- Console: ACL設定
- SDK: Java SDK - Bucketの管理
- API: Put BucketACL

バケットの ACL の取得

- コンソール: コンソールにログイン後、バケット属性で ACL を確認できます。
- SDK: Java SDK - Bucketの管理
- API: Get BucketACL

バケットのリストの表示

作成したすべてのバケットを表示するリストを表示できます。

参照

- コンソール：コンソールにログオンすると、作成したすべてのバケットのリストを直接表示できます。
- API: GetService
- SDK: Java SDK - List buckets

その他のリンク

- バケットの作成

バケット情報の取得

このリージョンは、データセンターの物理的な場所を表します。戻される [Location] フィールドは、バケットが配置されているリージョンを示します。たとえば、物理的な場所が中国東部 1（杭州）の場合、返される [Location] フィールドは oss-cn-hangzhou です。詳細については、OSSアクセスドメイン名を参照してください。

参照

- コンソール: コンソールにログインした後、ユーザーはコンソール上でバケット属性を直接表示できます。
- API: Get Bucket Location
- SDK: Java SDK - Get the bucket location

バケットの削除

作成したバケットを削除することができます。ただし、削除が発生する前にバケットをまずファイルとファイルの断片から空にする必要があります。バケット内のすべてのファイルを削除するには、ライフサイクル管理を使用することをお勧めします。

参照

- コンソール: バケットの削除
- API: バケットの削除
- SDK: Java SDK - バケットの削除

ファイルのアップロード

簡易アップロード

簡易アップロードとは、ユーザーが OSS API の Put Object メソッドを使用して 1 つのオブジェクトをアップロードする状況を意味します。これは、小さなファイルのアップロードなど、1 回の HTTP リクエスト操作でアップロードが完了するあらゆるシナリオで使用できます。

ファイルをアップロードするときにオブジェクトメタを設定する

簡易アップロードでは、Content-Type およびその他の標準 HTTP ヘッダーやユーザーが定義した情報など、オブジェクトを説明するオブジェクトメタを含めることができます。詳細については、「オブジェクトメタ」を参照してください。

アップロードの制限

- サイズ制限: このモードでの最大オブジェクトサイズは 5 GB です。
- 命名の制限
    - UTF-8 エンコーディングを使用します。
    - 長さは 1 ～ 1,023 バイトでなければなりません。
    - “/ “ または “\” で開始することはできません。

大きなファイルのアップロード

1 つの HTTP リクエストを使用して大きなオブジェクトをアップロードすると、アップロードに時間がかかりすぎる場合があります。この間にネットワークエラー (タイムアウトや切断など) が発生すると、アップロードに失敗する可能性が高くなります。この場合には、再開可能なアップロード (マルチパートアップロード) を検討してください。5 GB を超えるオブジェクトに対してのみ、再開可能なアップロード (マルチパートアップロード) を使用できます。詳細については、「再開可能なアップロード」を参照してください。

アップロードのセキュリティと権限付与

権限のないサードパーティが開発者のバケットにオブジェクトをアップロードするのを防ぐために、OSS ではバケットレベルおよびオブジェクトレベルのアクセス権限制御を使用できます。詳細については、「アクセス制御」を参照してください。バケットレベルおよびオブジェクトレベルのアクセス権限に加えて、OSS では、アカウントレベルの権限付与を使用して、サードパーティによるアップロード権限を付与することもできます。詳細については、「権限のあるサードパーティのアップロードによるアップロードのセキュリティ確保」を参照してください。

アップロード後の操作

OSS にオブジェクトがアップロードされたら、開発者はアップロードコールバックを使用して、指定したアプリケーションサーバーに対するコールバックリクエストを開始し、その後の操作を実行できます。アップロードしたイメージを処理するには、アップロードしたイメージのクラウド処理を利用できます。

関数を使用する際のリファレンス

- API: PutObject
- SDK: Java SDK-『オブジェクト』の「PutObject」
- コンソール: ファイルのアップロード

参照リンク先

- アップロードコールバック
- モバイル開発のアップロードシナリオの概要
- アップロードされたファイルのダウンロード
- アップロードしたイメージのクラウド処理
- アクセス制御を使用したアップロードのセキュリティ確保
- 権限のあるサードパーティのアップロードによるアップロードのセキュリティ確保
- アップロードしたファイルのコピー、削除、および管理

フォームアップロード

ユースケース

これは、ユーザーがOSS APIのPost Object要求を使用してオブジェクトをアップロードする状況を指します。アップロードするオブジェクトは5 GBを超えることはできません。このメソッドは、HTML Webページにフォームを埋め込み、オブジェクトをアップロードします。典型的なシナリオはウェブサイトです。ここでは、求人検索サイトを例にとります。

	フォームアップロードを使わないで	フォームアップロードを使用する
プロセス比較	1. ウェブサイトのユーザーが履歴書をアップロードします。 2. ウェブサイトサーバはアップロードページに応答します。 3. レジュームはサーバーにアップロードされます。 4. サーバーはレジュームを	1. ウェブサイトのユーザーが履歴書をアップロードします。 2.ウェブサイトサーバはアップロードページに応答します。 3. レジュームはOSSにアップロードされます。

	OSSにアップロードします。	

アップロード制限

- サイズ制限: フォームアップロードを使用する場合、オブジェクトは5 GBを超えることはできません。
- 命名の制限:
    - UTF-8エンコーディングを使用しています。
    - 長さは1〜102バイトである必要です。
    - “/“または”\”で始めることはできません

フォームアップロードの利点

- ファイル転送のステップがバイパスされます。
- フォームアップロードを使用しない伝統的な方法では、ファイルはWebサイトサーバーにまずアップロードされ、ボトルネックとなり、巨大なアップロードの場合にサイズを変更する必要があります。フォームのアップロードでは、ファイルはクライアントからOSSに直接アップロードされます。OSSは巨大なアップロードからのストレスを引き受け、サービス品質を保証します。

セキュリティと承認をアップロードする

権限のない第三者が開発者のバケットオブジェクトをアップロードするのを防ぐため、OSSはバケットおよびオブジェクトレベルのアクセス制御を提供します。詳細は、OSS Fine-grained Access Controlを参照してください。

サードパーティにアップロード許可を与えるには、PostObjectインターフェイスを使用できます。詳細はPostObjectを参照してください。

フォームアップロードの基本ステップ

投稿ポリシーを作成します。

ポストリクエストのポリシーフォームフィールドは、リクエストの有効性を確認するために使用されます。ポストリクエストのポリシーフォームフィールドは、リクエストの有効性を確認するために使用されます。詳細については、Post Policyを参照してください。

次のポリシーの例では、Webサイトユーザーによるアップロードの有効期限は2115-01-27T10：56：19Zです（テストを正常に完了するために、実際の使用では推奨されない長い有効期限を設定しています）ファイルサイズは104857600バイトです。

This example uses Python code and the policy is a string in JSON format.

```
policy="{\"expiration\":\"2115-01-27T10:56:19Z\",\"conditions\":[[\"content-length-range\", 0,
104857600]]}"
```

2. Base64を使用してポリシー文字列をエンコードします。
3. Base64でエンコードされたポリシーに署名するには、OSS AccessKeySecretを使用します。
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、アップロードするファイルを選択します。

完全なPythonコード:

```
#coding=utf8
import md5
import hashlib
import base64
import hmac
from optparse import OptionParser

def convert_base64(input):
return base64.b64encode(input)

def get_sign_policy(key, policy):
return base64.b64encode(hmac.new(key, policy, hashlib.sha1).digest())

def get_form(bucket, endpoint, access_key_id, access_key_secret, out):
#1. Construct a Post policy
policy="{\"expiration\":\"2115-01-27T10:56:19Z\",\"conditions\":[[\"content-length-range\", 0, 1048576]]}"
print("policy: %s" % policy)

#2. Encode the policy string using Base64
base64policy = convert_base64(policy)
print("base64_encode_policy: %s" % base64policy)

#3. Use the OSS AccessKeySecret to sign the Base64-encoded policy
signature = get_sign_policy(access_key_secret, base64policy)

#4. Construct an HTML page for uploads
form = '''
<html>
<meta http-equiv=content-type content="text/html; charset=UTF-8">
<head><title>OSS form upload (PostObject)</title></head>
<body>
<form action="http://%s.%s" method="post" enctype="multipart/form-data">
<input type="text" name="OSSAccessKeyId" value="%s">
<input type="text" name="policy" value="%s">
<input type="text" name="Signature" value="%s">
<input type="text" name="key" value="upload/${filename}">
<input type="text" name="success_action_redirect" value="http://oss.aliyun.com">
<input type="text" name="success_action_status" value="201">
<input name="file" type="file" id="file">
<input name="submit" value="Upload" type="submit">
</form>
```

```
</body>
</html>
''' % (bucket, endpoint, access_key_id, base64policy, signature)
f = open(out, "wb")
f.write(form)
f.close()
print("form is saved into %s" % out)

if __name__ == '__main__':
parser = OptionParser()
parser.add_option("", "--bucket", dest="bucket", help="specify ")
parser.add_option("", "--endpoint", dest="endpoint", help="specify")
parser.add_option("", "--id", dest="id", help="access_key_id")
parser.add_option("", "--key", dest="key", help="access_key_secret")
parser.add_option("", "--out", dest="out", help="out put form")
(opts, args) = parser.parse_args()
if opts.bucket and opts.endpoint and opts.id and opts.key and opts.out:
get_form(opts.bucket, opts.endpoint, opts.id, opts.key, opts.out)
else:
print "python %s --bucket=your-bucket --endpoint=oss-cn-hangzhou.aliyuncs.com --id=your-access-key-id --
key=your-access-key-secret --out=out-put-form-name" % __file__
```

このコードセグメントをpost_object.pyとして保存し、Pythonを使用して実行します。

```
Usage:
python post_object.py --bucket=Your bucket --endpoint=The bucket's OSS domain name --id=Your AccessKeyId --
key=Your AccessKeySecret --out=Output file name

Example:
python post_object.py --bucket=oss-sample --endpoint=oss-cn-hangzhou.aliyuncs.com --id=tphpxp --
key=ZQNJzf4QJRkrH4 --out=post.html
```

注意:

- 構築されたフォームでは、success_action_redirect value=http://oss.aliyun.comはアップロードが成功した後にジャンプするページを示します。自分のページで置き換えることができます。
- 構築されたフォームでは、success_action_status value=201は、成功したアップロードの後に ステータスコード201が返されることを示します。これは、必要に応じて置き換えることができます。
- 生成されたHTMLファイルがpost.htmlの場合、post.htmlを開き、アップロードするファイルを選択します。この例では、アップロードが成功した後、クライアントはOSSホームページにジャンプします。

関数の使用法のリファレンス

- API: PostObject

ベストプラクティス

- Web クライアントの直接データ転送
- CORS (Cross-Origin Resource Sharing)

リファレンスリンク

- アップロードコールバック
- OSS ベースのアプリ開発
- 簡易ダウンロード
- Cloud Processing for Uploaded Images
- Resource Access Management (RAM)
- 権限のあるサードパーティのアップロード
- オブジェクトメタ (ファイルのメタデータ)

再開可能なアップロード

再開可能なアップロード

適用されるシナリオ

簡易アップロード (PutObject) を使用して大きなファイルを OSS にアップロードすると、ネットワークエラーによってアップロードに失敗する場合があります。再試行すると、ファイルは最初からアップロードされます。この問題に対応するため、OSS では、ユーザーが中断されたアップロードを再開できるようマルチパートアップロードを提供しています。名前が表すように、マルチパートアップロードでは、アップロードするファイルを複数のデータブロック (OSS ではパート) に分割し、各パートを個別にアップロードします。アップロードが完了すると、OSS API が呼び出されて、パートが 1 つのオブジェクトに結合されます。

その他のアップロード方法と比べて、マルチパートアップロードは、次のシナリオで役立ちます。

- **繋がりにくいネットワーク環境。**携帯電話でのアップロードが失敗した場合、失敗したパートのみ再アップロードできるため、成功したパートを再アップロードする必要はありません。
- **再開可能なアップロードが必要な状況。**進行中のアップロードを一時停止した後で、一時停止されたパートから再開できます。
- **アップロードの高速化。**OSS にアップロードするファイルが非常に大きい場合、複数のパートを並行してアップロードすることで、アップロードを高速化できます。
- **ストリームアップロード。**サイズが不明なオブジェクトのアップロードを開始できます。このシナ

リオとしては、ビデオモニタリング業界が典型的です。

基本プロセス

このプロセスは次のとおりです。

- 指定したパートのサイズに従って、アップロードするファイルが分割されます。
- マルチパートアップロードタスクが初期化されます (InitiateMultipartUpload)。
- 順次または並行してパートがアップロードされます (UploadPart)。

アップロードを完了します (CompleteMultipartUpload)。

注意:

最後のパートを除き、各パートを 100 KB 未満にすることはできません。そのようにしないと、CompleteMultipartUpload の呼び出しに失敗します。

ファイルをパートに分割した後、アップロード中に、パートは指定された partNumbers に従って番号が付けられます。実際の実行では、パートは並行してアップロードされます。連続してアップロードする必要はありません。ユーザーのネットワークの状況とデバイスの負荷の両方を考慮する必要があるため、アップロードの速度は、並行してアップロードされるパートの数に合わせて上昇するとは限りません。

デフォルトでは、アップロードが完了しても CompleteMultipartUpload が呼び出されなかった場合は、パートは自動的にはリサイクルされません。そのため、アップロードを終了し、データに占有されたストレージ容量を削除するために、AbortMultipartUpload を呼び出します。アップロードされたパートを自動的にリサイクルするには、「ライフサイクル管理」を参照してください。

再開可能なアップロード

アップロードされたパートのライフサイクルは永続的であるため、再開可能なアップロード機能を実装するのは簡単です。

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に、ユーザーがアップロードを再開できるようにするには、ListMultipartUploads および ListParts API を使用して、オブジェクトに対するマルチパートアップロードタスクをすべて取得し、各タスクの完了したアップロードをリストにします。これにより、最後にアップロードされたパートからアップロードを再開できます。同じ方法は、アップロードの一時停止と再開にも適用できます。

この機能は、特にモバイルデバイスでのアップロードと大きなファイルのアップロードに役立ちます。

アップロードの制限

- サイズ制限: オブジェクトのサイズはパートのサイズによって決まります。この機能では、最大 10,000 パートをサポートします。最小パートサイズは 100 KB (最後のパートは 100 KB 未満でも可)、最大パートサイズは 5 GB です。オブジェクトのサイズは48.8 TBを超えることはできません。
- 命名の制限
    - UTF-8 エンコーディングを使用します。
    - 長さは 1 ～ 1023 バイトでなければなりません。
    - “/ ” または “\” で開始することはできません。

アップロードのセキュリティと権限付与

権限のないサードパーティが開発者のバケットにオブジェクトをアップロードするのを防ぐために、OSS ではバケットレベルおよびオブジェクトレベルのアクセス権限制御を使用できます。詳細については、「アクセス制御」を参照してください。バケットレベルおよびオブジェクトレベルのアクセス権限に加えて、OSS では、アカウントレベルの権限付与を使用して、サードパーティによるアップロード権限を付与することもできます。詳細については、「権限のあるサードパーティのアップロードによるアップロードのセキュリティ確保」を参照してください。

アップロード後の操作

OSS にオブジェクトがアップロードされたら、開発者はアップロードコールバックを使用して、指定したアプリケーションサーバーに対するコールバックリクエストを開始し、その後の操作を実行できます。

関数を使用する際のリファレンス:

- API: MultipartUpload、InitiateMultipartUpload、UploadPart、UploadPartCopy、CompleteMultipartUpload、AbortMultipartUpload、ListMultipartUploads、ListParts
- SDK: Java SDK-『MultipartUpload』の「マルチパートアップロード」

参照リンク先:

- アップロードコールバック
- モバイル開発のアップロードシナリオの概要
- アップロードされたファイルのダウンロード
- アクセス制御を使用したアップロードのセキュリティ確保
- 権限のあるサードパーティのアップロードによるアップロードのセキュリティ確保
- アップロードしたファイルのコピー、削除、および管理

追加アップロード

追加アップロード

適用されるシナリオ

簡易アップロード、フォームアップロードおよび再開可能なアップロードという方法では、アップロードの完了後にコンテンツが固定された標準タイプのオブジェクトが作成されます。これらを読み取ることはできますが、変更することはできません。これらのオブジェクトのコンテンツが変更された場合、同じ名前のオブジェクトをアップロードして、コンテンツを上書きする必要があります。この点では、OSS がファイルシステムと大きく異なります。

この機能は、ビデオモニタリングやライブでのビデオブロードキャストなどの多くのアプリケーションシナリオにとって不便です。ビデオデータがリアルタイムで継続的に生成されるからです。他のアップロード方法を使用すると、ビデオストリームをスライスして小さなデータに分割し、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。これらの方法を実際に使用すると、次の欠点が明らかです。

- ソフトウェアアーキテクチャが非常に複雑であるため、ファイルのフラグメント化などの煩雑な問題を考慮する必要があります。
- 生成されたオブジェクトのリストなど、メタデータ用のストレージ容量が必要になります。そのため、各リクエストでメタデータを読み取り、新しいオブジェクトが生成されたかどうかを判断する必要があります。これにより、サーバーへのアクセスによる負荷が増大します。さらに、各クライアントリクエストを 2 回送信する必要があるため、一定の遅延が発生します。
- 分割後のオブジェクトが小さい場合は、遅延は非常に短く済みます。ただし、ほとんどのオブジェクトで管理が複雑になります。分割後のオブジェクトが大きい場合は、遅延が増大します。

このようなシナリオでより簡単に開発して、コストを削減するために、OSS では Append Object メソッドを提供し、ユーザーがオブジェクトの後ろにコンテンツを直接追加できます。このメソッドは、追加可能オブジェクトの操作に使用できます。その他のメソッド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、すぐに読み取ることができます。

Append Object を使用すると、先ほどのシナリオが非常に簡単になります。ビデオデータが生成された場合は、Append Object メソッドを使用して、生成されたデータを同じオブジェクトに直ちに追加できます。クライアントは、定期的にオブジェクトの長さを取得して、その長さを過去の値と比較するだけで済みます。新しい読み取り可能なデータが見つかると、読み取り操作がトリガーされ、新しくアップロードされたデータセグメントが取得されます。このメソッドによって、大幅にアーキテクチャが簡素化され、アプリケーションのスケーラビリティが高まります。

ビデオの使用例以外、Append Object メソッドを使用すると、ログデータを追加することもできます。

アップロードの制限

- サイズ制限: このモードでの最大オブジェクトサイズは 5 GB です。
- 命名の制限
    - UTF-8 エンコーディングを使用します。
    - 長さは 1 ～ 1023 バイトでなければなりません。
    - “/ “ または “\” で開始することはできません。
- ファイルタイプ: Append Object を使用して作成されたファイルに対してのみ、新しいデータを追加できます。そのため、簡易アップロード、フォームアップロード、またはマルチパートアップロードを使用して作成されたファイルには新しいデータを追加できません。

アップロードのセキュリティと権限付与

権限のないサードパーティが開発者のバケットにオブジェクトをアップロードするのを防ぐために、OSS ではバケットレベルおよびオブジェクトレベルのアクセス権限制御を使用できます。詳細については、「アクセス制御」を参照してください。バケットレベルおよびオブジェクトレベルのアクセス権限に加えて、OSS では、アカウントレベルの権限付与を使用して、サードパーティに対してアップロード権限を付与することもできます。詳細については、「権限のあるサードパーティのアップロードによるアップロードのセキュリティ確保」を参照してください。

関数を使用する際のリファレンス:

- API: Append Object
- SDK: Java SDK-『Append Object の例』

注意

Append Object では、アップロードコールバックはサポートされていません。

参照リンク先:

- アップロードされたファイルのダウンロード
- アップロードしたイメージのクラウド処理
- アクセス制御を使用したアップロードのセキュリティ確保
- 権限のあるサードパーティのアップロードによるアップロードのセキュリティ確保

権限のあるサードパーティのアップロード

権限のあるサードパーティのアップロード

適用されるシナリオ

一般的なクライアント/サーバーシステムのアーキテクチャでは、サーバーがクライアントからのリクエストを受信し、処理します。OSS をバックエンドストレージサービスとして使用するシナリオでは、クライアントはアップロードするファイル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がそのファイルを OSS に転送します。このプロセスでは、データを 2 回転送する必要があります。つまり 1 回はクライアントからサーバーに、もう 1 回はサーバーから OSS に、それぞれ転送する必要が生じます。そのため、アクセス数が多いと、多数のクライアントから同時にアップロード要求が発生し、サーバーに十分な帯域幅リソースが必要です。これにより、アーキテクチャのスケーラビリティに課題が生じます。

この課題は、OSS が提供する “権限のあるサードパーティのアップロード” 機能によって解決することができます。この機能を使用すると、各クライアントはサーバーを経由することなく、OSS に直接ファイルを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、大容量のデータを処理する OSS の能力を十分に活用できます。帯域幅と同時操作の制限を気にかける必要がないため、サーバーはサービスの処理に集中することができます。

現在、アップロード権限を付与する方法には次の 2 つがあります。

URL の署名

URL の署名は、アクセス権限を付与する方法の 1 つです。この方法では、リクエスト URL に OSS の AccessKeyID フィールドと Signature フィールドを追加し、ユーザーが直接この URL を使用してアップロードできるようにします。各 URL の署名には、有効期限が含まれており、セキュリティを確保します。詳細については、「URL への署名の追加」を参照してください。

一時的なアクセス資格情報

一時的なアクセス資格情報は、Alibaba Cloud SecurityTokenService(STS) によって付与され、ユーザーにアクセス権限を与えます。一時的なアクセス資格情報の実装については、「STS Java SDK」を参照してください。

- クライアントは、権限を取得するために、サーバーに対するリクエストを開始します。サーバーは、最初にクライアントの正当性を検証します。クライアントが正当である場合は、サーバーは独自の AccessKey を使用して、STS に対して権限のリクエストを開始します。詳細については、「アクセス制御」を参照してください。
- サーバーは一時的な資格情報を取得した後、その情報をクライアントに返します。
- クライアントはこれらの一時的な資格情報を使用して、OSS に対するアップロードリクエストを開始します。リクエストの構造の詳細については、「一時的なアクセス資格情報の発行」を参照して

ください。クライアントは、これらの資格情報をキャッシュに保存し、有効期限が切れるまで後続のアップロードに使用できます。有効期限の後は、新しい資格情報をサーバーに要求する必要があります。

ベストプラクティス

- Web クライアントの直接データ転送およびアップロードコールバック

参照リンク先:

- アップロードコールバック
- モバイル開発のアップロードシナリオの概要
- アップロードされたファイルのダウンロード
- アップロードしたイメージのクラウド処理
- アクセス制御を使用したアップロードのセキュリティ確保
- フォームアップロード

アップロードコールバック

適用可能な使用例

オブジェクトのアップロードが完了すると、OSS はアプリケーションサーバにコールバックを提供できます。コールバックを実装するには、関連するコールバックパラメータをOSSに送信されたリクエストに添付するだけです。コールバックを現在サポートしている API には、PutObject、PostObject、CompleteMultipartUpload などがあります。

アップロードコールバックの典型的な使用例は、承認された第三者によるアップロードを処理することです。クライアントは、オブジェクトを OSS にアップロードするときにサーバーのコールバックを指定します。クライアントのアップロードタスクが OSS で完了した後、OSS は自動的にアプリケーションサーバーへのコールバックの HTTP リクエストを開始します。これにより、アップロードが完了したことをサーバーに速やかに通知するので、サーバーはデータベースの変更などの操作を完了できます。コールバック要求がサーバーからの応答を受け取ると、OSS はその状態をクライアントに返します。

OSS が POST コールバックリクエストをアプリケーションサーバーに送信すると、POST リクエストの本体には特定の情報を保持するいくつかのパラメータが含まれます。このようなパラメータは、システム定義のパラメータ（バケット名やオブジェクト名など）とユーザー定義のパラメータの2種類に分類されます。コールバックを含むリクエストを OSS に送信するときに、アプリケーションロジックに基づいてユーザー定義のパラメータを指定できます。ユーザー定義パラメーターを使用して、アプリケーション・ロジックに関連す

る情報（例えば、リクエストイニシエーターのユーザー ID）を使用することができます。ユーザー定義のパラメータの使用方法については、コールバックを参照してください。

アップロードコールバックを適切に使用することで、クライアントのロジックの複雑さを軽減し、ネットワークリソースの消費を減らすことができます。プロセスは次のとおりです。

注意:

- サポート対象リージョンには中国本土（中国東部3および中国東部5を除く）、香港リージョン、アジア太平洋南1、アジア太平洋SE 2、米国東部、米国西部、アジア太平洋北東1、中欧1および中東1 。
- 現在のところ単純なアップロード（PutObject）、フォームアップロード（PostObject）、マルチパートアップロード（Complete Multipart Upload）操作のみがアップロードコールバックをサポートしています。

機能の使用

- SDK: iOS アップロード後のコールバック通知

ベストプラクティス

Webクライアントでの直接データ転送とコールバックのアップロード

コールバックアプリケーションサーバーの構築方法（サンプルコードはダウンロード可能）

参照

- モバイルアプリケーションでの直接データ転送
- モバイルアプリケーションの許可管理
- モバイルデバイスの開発の紹介とユースケースの紹介
- アップロードされたオブジェクトのダウンロード
- アップロードされた画像のクラウド処理
- アップロードされたオーディオ/ビデオファイルのクラウド処理
- アップロードセキュリティのアクセス制御
- 第三者によるアップロードセキュリティの承認
- ファイルのコピーや削除などのアップロードされたファイルの管理

ファイルのダウンロード

簡易ダウンロード

簡易ダウンロード

簡易ダウンロードとは、アップロードしたファイル (オブジェクト) をユーザーがダウンロードすることです。オブジェクトのダウンロードは、HTTP GET リクエストによって行われます。

ユーザーがあるオブジェクトにアクセスする際に起こることとして、次の 2 つが挙げられます。

- アクセス先のオブジェクトには匿名読み取りアクセスが許可されていないものの、ユーザーが対応する AccessKey を持っているため、これを使用して GET リクエストに署名してオブジェクトにアクセスできる。
- アクセス先のオブジェクトには匿名読み取りアクセスが許可されているため、ユーザーが GET リクエストでオブジェクトに直接アクセスできる。

オブジェクトの URL の生成規則については、「OSS へのアクセス」を参照してください。

ユーザー定義ドメイン名を使用したオブジェクトへのアクセスについては、「ユーザー定義ドメイン名を使用した OSS へのアクセス」を参照してください。

オブジェクトおよびバケットのアクセス権限の制御についての詳細については、「アクセス制御」を参照してください。

非公開のバケット内のオブジェクトをダウンロードする権限をサードパーティのユーザーに付与するには、「権限のあるサードパーティのダウンロード」を参照してください。

再開可能なダウンロードを利用する場合は、「再開可能なダウンロード」を参照してください。

機能の使用方法のリファレンス:

- API: Get Object
- SDK: Java SDK-『オブジェクト』
- コンソール: ファイルのアクセスアドレスの取得

ベストプラクティス

- RAM and STS User Guide

参照リンク先:

- ファイルのアップロード方法
- アップロードコールバック
- モバイルクライアント開発とダウンロードシナリオの概要
- 安全なダウンロードのアクセス制御
- 権限のあるサードパーティのダウンロードによるダウンロードのセキュリティ確保
- アップロードしたファイルのコピー、削除、および管理

再開可能なダウンロード

再開可能なダウンロード

OSS は “指定した点からオブジェクトのダウンロードを開始する” 機能を備えています。これにより、大きなオブジェクトを複数のダウンロードに分割できます。ダウンロードが中断した場合、再開時には中断したところからダウンロードを続行します。

簡易アップロードと同様、オブジェクトの読み取り権限を持っている必要があります。再開可能なダウンロードがサポートされるのは、Range パラメーターが設定されている場合です。この機能はオブジェクトサイズが大きい場合に推奨されます。Range の定義については、HTTP の RFC を参照してください。リクエストヘッダーで Range パラメーターを指定した場合は、返されるメッセージにファイル全体の長さと今回返された範囲が示されます。

たとえば、Content-Range: bytes 0-9/44 は、ファイル全体の長さが 44 で今回返された範囲が 0 ~ 9 であることを示しています。範囲の要件が満たされていない場合は、ファイル全体が転送され、結果に Content-Range は含まれません。リターンコードは 206 です。

機能の使用方法のリファレンス:

- API: Get Object

参照リンク先:

- ファイルのアップロード方法
- アップロードコールバック
- モバイルクライアント開発とダウンロードシナリオの概要
- アップロードしたイメージのクラウド処理
- 安全なダウンロードのアクセス制御
- 権限のあるサードパーティのダウンロードによるダウンロードのセキュリティ確保
- アップロードしたファイルのコピー、削除、および管理

第三者へのダウンロード権限付与

第三者へのダウンロード権限付与

第三者に非公開バケット内のオブジェクトをダウンロードする権限を付与する場合、ダウンロードを実行するユーザーに直接 AccessKey を付与しないでください。次の 2 つの方法うち、いずれかの方法により行ってください。

URL の署名

OSS では署名を使ったダウンロード方法を用意しています。開発者は URL に署名を付け加えて第三者に転送することで、アクセス権限を付与できます。第三者のユーザーは、HTTP GET リクエストを使用してこの URL にアクセスすることで、オブジェクトをダウンロードできます。

実装方法

署名を含む URL の例:

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

URL の署名には、少なくとも、Signature、Expires、OSSAccessKeyID の 3 つのパラメーターが含まれている必要があります。

- OSSAccessKeyId: 開発者の AccessKeyId です。
- Expires: 開発者が定める URL の有効期限です。
- Signature: 開発者の署名の文字列です。詳細については、API ドキュメントの署名のセクションを参照してください。

注意: このリンクは URL エンコードを行う必要があります。

具体的な実装

機能の使用方法のリファレンス:

- API: Get Object
- SDK: Java SDK-『URL の署名を使用したアクセスの許可』

- コンソール: ファイルのアクセスアドレスの取得

注意：非公開読み書きに設定されているバケットがある場合は、コンソールで取得されるアクセスアドレスは、署名付きの URL になります。そうでない場合は、URL に署名は付きません。

一時的なアクセス資格情報

OSS では、STS (Security Token Service) を使用して一時的なアクセス資格情報を第三者のユーザーに提供します。リクエストヘッダーに署名を追加することで、オブジェクトにアクセスできます。この権限付与方法はモバイルでのダウンロードに適しています。一時的なアクセス資格情報の実装の詳細については、『STS Java SDK』を参照してください。

実装方法

1. 第三者のユーザーが、リクエストをアプリケーションサーバーに送信し、STS によって発行される AccessKeyID、AccessKeySecret、および STS トークンを取得します。
2. 続いて、STS の AccessKeyID、AccessKeySecret、および STS トークンを署名として使い、開発者のオブジェクトリソースをリクエストします。

関数を使用する際のリファレンス:

- API: 一時的なアクセス資格情報
- コンソール: ファイルのアクセスアドレスの取得

参照リンク先:

- ファイルのアップロード方法
- アップロードコールバック
- モバイルクライアント開発とダウンロードシナリオの概要
- アップロードしたイメージのクラウド処理
- 安全なダウンロードのアクセス制御
- 権限のある第三者のダウンロードによるダウンロードのセキュリティ確保
- アップロードしたファイルのコピー、削除、および管理

ファイルの管理

オブジェクトメタ (ファイルのメタデータ)

オブジェクトメタは、OSS にアップロードされたファイルの属性を記述します。これらの属性には、HTTP 標準属性 (HTTP ヘッダー) とユーザーメタ (カスタムメタデータ) の 2 種類があります。ファイルのメタデータは、さまざまな方法でファイルをアップロードまたはコピーするときに設定できます。

HTTP 標準属性

名前	説明
Cache-Control	オブジェクトがダウンロードされる際の Web ページのキャッシュ動作
Content-Disposition	オブジェクトがダウンロードされる際のオブジェクトの名前
Content-Encoding	オブジェクトがダウンロードされる際のコンテンツのエンコーディング形式
Content-Language	オブジェクトがダウンロードされる際のコンテンツの言語エンコーディング
Expires	有効期限
Content-Length	オブジェクトのサイズ
Content-Type	オブジェクトのファイルタイプ
Last-Modified	最終変更日時

ユーザーメタ

この属性は、ユーザがオブジェクトの説明を改善したい場合があると考慮して設計されています。OSS では、x-oss-meta-location のように “x-oss-meta- ” で始まるすべてのパラメーターはユーザーメタと見なされます。1 つのオブジェクトに対して同様のパラメーターを複数設定することはできますが、すべてのユーザーメタの合計サイズは 8 KB 以下にする必要があります。ユーザーメタ情報は、GetObject または HeadObject 操作の際に HTTP ヘッダーで返されます。

オブジェクトをアップロードするときにオブジェクトメタを設定する

オブジェクトをアップロードするときに、オブジェクトメタを設定することができます。

関数を使用する際のリファレンス:

- API: Put Object
- SDK: Java SDK - Set the HTTP Headers and User-defined meta information

マルチパートアップロード (再開可能なデータ転送) を使用する際にオブジェクトメタを設定できます。

関数を使用する際のリファレンス:

- API: InitiateMultipartUpload

- SDK: Java SDK - オブジェクトのアップロード

オブジェクトをアップロードした後でオブジェクトメタを変更する

実際のデータは変更せずにオブジェクトメタを変更するには、Copy Object インターフェイスを使用します。そのために必要な操作は、新しいメタデータ (情報が完備している必要があります) を HTTP ヘッダーに配置し、コピー元とコピー先のアドレスをオブジェクトの現在のアドレスに設定することだけです。

関数を使用する際のリファレンス:

- API: オブジェクトのコピー
- SDK: Java SDK -オブジェクトの管理

オブジェクトメタの取得

この機能は、ユーザーがオブジェクトメタを取得する必要があり、オブジェクトデータは不要な場合に使用します。

関数を使用する際のリファレンス:

- API: Head Object
- SDK: Java SDK - オブジェクトの管理

オブジェクトの一覧表示

オブジェクトの一覧表示

この機能は、ユーザーによってバケットにアップロードされたファイル (オブジェクト) をリストします。OSS インターフェイスを呼び出すことにより、特定のバケットから最大 1,000 個のオブジェクトのリストを一度に取得できます。次の 4 つのパラメーターによって幅広く活用できます。

名前	機能
Delimiter	オブジェクト名の文字をグループ化するために使用されます。指定したプレフィックスから最初の Delimiter までの名前を持つオブジェクトはすべて要素のグループ (CommonPrefixes) として機能します。
Marker	アルファベット順で marker 後の最初のエントリから結果を返すように設定します。
MaxKeys	1 件のリクエストに対して返されるオブジェクト

	の最大数を制限します。指定されていない場合のデフォルト値は 100 です。MaxKeys の値は 1,000 以下にする必要があります。
Prefix	指定されたプレフィックスが Keys に含まれているオブジェクトだけが返されることを示します。プレフィックスを使用するクエリから返されるキーにはそのプレフィックスも含まれることに注意してください。

フォルダーのシミュレーション

OSS サービスでは、フォルダーを使用しません。すべての要素はオブジェクトとして格納されます。シミュレーションフォルダーを作成することは、サイズが 0 のオブジェクトを作成することを意味します。このオブジェクトはアップロードおよびダウンロードできます。コンソールには、” / “ で終わるオブジェクトがフォルダーとして表示されるため、 この方法でシミュレーションフォルダーを作成できます。

Delimiter と Prefix を組み合わせて、フォルダー機能をシミュレートできます。Delimiter と Prefix の組み合わせは、次の目的で使用されます。

- フォルダーの名前を Prefix として設定すると、そのプレフィックスで始まるファイルが列挙されて、そのフォルダーのすべてのファイルとサブフォルダー (ディレクトリ) が再帰的に返されます。ファイル名は Contents として示されます。
- Delimiter を “/ “ に設定すると、返される値にはフォルダー内のファイルが列挙され、サブフォルダー (ディレクトリ) は CommonPrefixes セクションで返されます。サブフォルダー内のファイルとフォルダーは再帰的に表示されません。

例:
この例では、oss-sample という OSS バケットに以下のオブジェクトが含まれています。

File D
Directory A/File C
Directory A/File D
Directory A/Directory B/File B
Directory A/Directory B/Directory C/File A
Directory A/Directory C/File A
Directory A/Directory D/File B
Directory B/File A

1. 第 1 レベルのディレクトリおよびファイルをリストします。
API リクエストの規則に基づいて、Prefix を "" に、Delimiter を "/" に設定します。
返される結果は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
<Name>oss-sample</Name>
<Prefix></Prefix>
<Marker></Marker>
<MaxKeys>1000</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>false</IsTruncated>
```

```
<Contents>
<Key>File D</Key>
<LastModified>2015-11-06T10:07:11.000Z</LastModified>
<ETag>"8110930DA5E04B1ED5D84D6CC4DC9080"</ETag>
<Type>Normal</Type>
<Size>3340</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>oss</ID>
<DisplayName>oss</DisplayName>
</Owner>
</Contents>
<CommonPrefixes>
<Prefix>Directory A/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Directory B/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

次のことがわかります。
Contents は第 1 レベルのファイル "File D" を返します。
CommonPrefixes は第 1 レベルのディレクトリ "Directory A/" と "Directory B/" を返しますが、その中のファイルは示されません。

2. Directory A 内の第 2 レベルのディレクトリとファイルをリストします。
API リクエストの規則に基づいて、Prefix を "Directory A" に、Delimiter を "/" に設定します。
返される結果は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
<Name>oss-sample</Name>
<Prefix>Directory A/</Prefix>
<Marker> </Marker>
<MaxKeys>1000</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>false</IsTruncated>
<Contents>
<Key>Directory A/File C</Key>
<LastModified>2015-11-06T09:36:00.000Z</LastModified>
<ETag>"B026324C6904B2A9CB4B88D6D61C81D1"</ETag>
<Type>Normal</Type>
<Size>2</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>oss</ID>
<DisplayName>oss</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>Directory A/File D</Key>
<LastModified>2015-11-06T09:36:00.000Z</LastModified>
<ETag>"B026324C6904B2A9CB4B88D6D61C81D1"</ETag>
<Type>Normal</Type>
<Size>2</Size>
<StorageClass>Standard</StorageClass>
```

```
<Owner>
<ID>oss</ID>
<DisplayName>oss</DisplayName>
</Owner>
</Contents>
<CommonPrefixes>
<Prefix>Directory A/Directory B/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Directory A/Directory C/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Directory A/Directory D/</Prefix>
</CommonPrefixes>
</ListBucketResult>

次のことがわかります。
Contents は、第 2 レベルのファイル "Directory A/File C" と "Directory A/File D" を返します。
CommonPrefixes は第 2 レベルのディレクトリ "Directory A/Directory B/"、"Directory A/Directory C/"、"Directory A/Directory D/" を返します。これらのディレクトリ内のファイルの名前は示されません。
```

関数を使用する際のリファレンス:

- API: Get Bucket
- SDK: Java SDK - List objects in a bucket

オブジェクトのコピー

ファイルをバケットからコピーすることができます。オブジェクトを (内容を変更せずに) 別のバケットにコピーすることが必要な場面では、通常、オブジェクトをダウンロードしてからコピー先のバケットにアップロードします。ただ、この方法だとデータが変更されていないため、帯域だけが無駄に使用されます。OSS には CopyObject 関数が用意されており OSS 内でオブジェクトをコピーすることができるため、ユーザーと OSS の間で大量のデータを送信する必要はありません。

また、OSS では名前の変更がサポートされていないため、オブジェクトの名前を変更する際は OSS CopyObject インターフェイスを呼び出すことをお勧めします。まず、元のデータを新しい名前のオブジェクトにコピーし、その後、元のファイルを削除します。オブジェクトのオブジェクトメタだけを変更する場合も、元のアドレスと変更後のアドレスを同じ値に設定して CopyObject インターフェイスを呼び出します。これにより、オブジェクトメタだけが更新されます。オブジェクトメタの詳細については、「オブジェクトメタ」を参照してください。

この操作を実行する際は、次の点に注意する必要があります。

- ソースオブジェクトを操作する権限を持っている必要があります。そうでない場合、操作は失敗します。
- この操作では、リージョンをまたいでデータをコピーすることはできません。たとえば、杭州のバ

ケット内のオブジェクトを青島にコピーすることはできません。
- この操作でサポートされるオブジェクトのサイズは 1 GB までです。

関数を使用する際のリファレンス:

- API: 『オブジェクトのコピー』
- SDK: Java SDK - オブジェクトのコピー

大きいオブジェクトのコピー

OSS では、再開可能なデータ転送に似た、大きいファイルをコピーするための関数がサポートされています。

基本的な操作は、「再開可能なデータ転送」で説明されているものと同じです。1 つ異なる点は、UploadPart を UploadPartCopy に置き換えるということです。UploadPartCopy の構文は、基本的に UploadPart と同じです。ただし、データは HTTP リクエストから直接アップロードされるのではなく、ソースオブジェクトから取得されます。

関数を使用する際のリファレンス:

- API: UploadPartCopy
- SDK: Java SDK - Copy an object

オブジェクトの削除

オブジェクトの削除

OSS バケットにアップロード済みのファイル (オブジェクト) を削除することができます。OSS では、次のようにしてオブジェクトを削除できます。

- 単独のオブジェクトの削除: 指定したオブジェクトを削除します。
- バッチ削除: 一度に最大 1,000 個のオブジェクトを削除します。
- 自動削除: この機能は、特定のルールに従って大量のオブジェクトを削除する必要がある場合に使用します。たとえば、一定日数前に作成されたオブジェクトを定期的に削除するため、またはバケット全体を定期的に空にするためなどです。そのためには、ライフサイクル管理を行うことをお勧めします。ルールを指定すると、OSS はそのルールを使用して期限切れのオブジェクトを再利用するため、ユーザーが削除を要求する数が大幅に減少し、削除の速度が向上します。

関数を使用する際のリファレンス:

- API: 「Delete Object」および「Delete Multiple Objects」
- SDK: Java - Delete Files
- Console: Delete Files

ライフサイクル管理

オブジェクトのライフサイクル管理

OSS では、オブジェクト (ファイル) のライフサイクル管理によってオブジェクトを管理できます。バケットのライフサイクルを設定して、バケットのオブジェクトのさまざまなルールを定義できます。現在、ユーザーはルールを使用して、一致するオブジェクトを削除できます。各ルールは、次の要素があります。

- オブジェクト名のプレフィックス: このルールは、一致したプレフィックスが付加されたオブジェクトのみに適用されます。
- 操作: 一致したオブジェクトに対してユーザーが実行する操作です。
- 日付または日数: 指定した日付またはオブジェクトの最終変更日時から指定の日数後にオブジェクトで操作を実行します。

オブジェクト名のプレフィックスがルールのプレフィックスに一致すると、ルールがオブジェクトに適用されます。たとえば、バケットに次のオブジェクトがあったとします。

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

- プレフィックスが logs/ であれば、プレフィックス logs/ が付加された最初の 3 つのオブジェクトにこのルールが適用されます。
- プレフィックスが doc/readme.txt であれば、doc/readme.txt のみがルールに適用されます。

現在、ルールでは、” 期限切れによる削除” が許可されています。たとえば、「プレフィックス logs/ が付加されたオブジェクトの最終更新日時が 30 日前の場合は、オブジェクトを削除する」というルールを設定できます。doc/readme.txt を削除する日付も指定できます。

オブジェクトが期限切れルールに一致すると、OSS は、GET Object リクエストまたは HEAD Object リクエストへの応答に x-oss-expiration ヘッダーを含めます。このヘッダーには、キーと値の 2 つのペアが含まれます。expiry-date はオブジェクトの有効期限を示し、rule-id は一致したルール ID を示します。

OSS のオープンインターフェイスを介してバケットのライフサイクル構成を設定できます。ライフサイクル設定は、XML 形式で提供されます。次に具体的な例を示します。

```
<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>2014-12-31T00:00:00.000Z</CreatedBeforeDate>
</Expiration>
</Rule>
</LifecycleConfiguration>
```

上記の例のすべての要素の説明は、次のとおりです。

- **ID**: 各ルールの一意の ID。
- **Status**: Enabled または Disabled。OSS では、Enabled のルールのみが使用されます。
- **Prefix**: プレフィックスです。
- **Expiration**: 操作の有効期限。下位要素 **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指定します。
    - **CreatedBeforeDate** は、最終変更日時が 2014-12-31T00:00:00.000Z よりも前のファイルが削除されることを示します。この日時よりも後に変更されたオブジェクトは削除されません。
    - **Days** は、最後の更新が 10 日以上前のファイルが削除されることを示します。

最初のルールでは、OSS は、プレフィックス logs/ が付加されている 10 日前に更新されたオブジェクトを削除します。2 番目のルールは、プレフィックスが doc/ が付加され最終変更日が 2014 年 12 月 31 日よりも前のオブジェクトを削除すると指定していますが、ステータスが Disabled になっているため、このルールは有効になりません。

詳細分析:

1. 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。
2. プレフィックスが空の場合、ルールはバケットのすべてのオブジェクトに適用されます。
3. ルールの各プレフィックスは一意である必要があります。たとえば、バケットにプレフィックスがそれぞれ logs/ と logs/program である 2 つのルールがある場合、OSS はエラーを返します。
4. 特定の日付にオブジェクトを削除するようにルールを設定する場合、その日付は UTC 時刻の午前 00:00 時であり、ISO8601 形式に準拠している必要があります。たとえば、2014-01-01T00:00:00.000Z のようにします。上記の例では、2014 年 1 月 1 日午前 00:00 時になると、一致したオブジェクトが削除されます。
5. ルールでオブジェクトを削除するまでの日数が指定されている場合、OSS は指定された日数を最

終更新日時 ([Last-Modified]) に加え、得られた日時を次の午前 00:00 時 (UTC 時刻) に丸めます。たとえば、オブジェクトの最終更新日時が 2014 年 4 月 12 日午前 01:00 時で、一致したルールで指定されている日数が 3 の場合、有効期限は 2014 年 4 月 16 日午前 00:00 時になります。

6. OSS は、ルールに一致したオブジェクトを指定された日時に削除します。通常、オブジェクトは指定された日時の直後に削除されることに注意してください。通常、オブジェクトの最終更新日時はその作成日時と同じであることが多いです。オブジェクトが複数回 PUT 操作された場合、最終更新日時はオブジェクトが最後に PUT 操作された日時になります。オブジェクトが自身にコピーされた場合、最終更新日時はオブジェクトが最後にコピーされた日時になります。

リファレンス:

- API: Put Bucket Lifecycle

リージョン間レプリケーション

バケットのリージョン間レプリケーションでは、バケット内のオブジェクトを自動的かつ非同期的に別の OSS データセンターにコピーします。ソースバケット内のオブジェクトへの変更 (作成、上書き、削除など) が、ターゲットバケットに同期されます。この機能は、リージョン間でディザスタリカバリの方法として最適であり、また、ユーザーがデータをコピーする際に使用できます。ターゲットバケットのオブジェクトはソースバケットのオブジェクトの正確なコピーであり、オブジェクト名、メタデータ、内容 (作成日時、オーナー、ユーザー定義メタデータ、オブジェクト ACL、オブジェクトコンテンツなど) は同じです。

アプリケーションシナリオ

バケットのリージョン間レプリケーションの設定が必要なケース:

- コンプライアンス要件: OSS では、格納されている各オブジェクトについて物理ディスクに複数のコピーが作成されますが、定められた要件を満たすために、コピーは相互に一定の距離のある場所に格納する必要があります。リージョン間同期を使用すると、遠く離れた OSS データセンター間でデータをコピーして、このようなコンプライアンス要件を満たすことができます。
- レイテンシの最小化: お客様が地理的に離れた 2 つの場所で活動している場合があります。オブジェクトへのアクセスのレイテンシを最小化するために、ユーザーに近い OSS データセンターにオブジェクトのコピーを保持しておくことが考えられます。
- データバックアップとディザスタリカバリ: 非常に高いデータセキュリティおよび可用性が必要とされており、自然災害から保護するために、作成されたすべてのデータのコピーを第 2 のデータセンターに明示的に保持したい場合があります。地震、津波などによって OSS データセンターが損傷したときは、別の OSS データセンターでバックアップデータを使用することができます。
- データのコピー: 業務上の理由から、データを 1 つの OSS データセンターから別のデータセンターに移行する必要がある場合があります。

- 操作上の理由: 複数のデータセンターにコンピューティングクラスターが存在し、それらを使用して 1 つのオブジェクトグループを分析することがあります。この場合は、オブジェクトのコピーをこれらのリージョンに配置することが考えられます。

使用方法

現在、名前の異なるバケットのリージョン間レプリケーションもサポートされています。異なるリージョンに存在する 2 つのバケットに対して同期機能を有効にすることで、ソースバケットのデータをターゲットバケットにリアルタイムで同期することができます。下記のような特徴があります。

1. リアルタイムのデータ同期: この機能では、データの追加、削除、変更がリアルタイムでモニターされ、変更がターゲットリージョンのバケットに同期されます。2 MB 以上のファイルは、同期に数分かかる可能性があります。同期により、データは両方の側で完全に一致します。
2. 履歴データの移行: この機能では、ソースバケット内の履歴データも同期して、2 つの完全に同じデータコピーを作成することができます。
3. リアルタイム同期の進捗の取得: リアルタイムのデータ同期については、最新の同期日時ノードが示されます。履歴データの移行については、移行されたデータの割合が示されます。
4. 簡単な設定: OSS コンソールは使いやすい管理インターフェイスを備えています。

制約

1. 同期状態の 2 つのバケットは、同時に操作することができます。しかし、ソースバケットからコピーされたオブジェクトによってターゲットバケット内の同じ名前のオブジェクトが上書きされる可能性があります。注意してください。
2. バケットのレプリケーションでは非同期のコピー方法が使用されるため、データがターゲットブックにコピーされるまでに多少時間がかかる可能性があります。具体的には、データ量に応じて数分から数時間かかり場合があります。
3. リージョン間同期は、同期される 2 つのバケットのデータが、この 2 つ以外のバケットとの間では同期が許可されない場合にのみ適用されます。たとえば、バケット A からバケット B への同期が有効化されているとき、バケット A からバケット C への同期は、バケット A とバケット B の間の同期設定を削除しない限り有効にすることはできません。同様に、バケット A からバケット B への同期が有効なとき、バケット C からバケット B への同期を有効にすることはできません。
4. データ同期に参加する 2 つのバケットは、異なるリージョンに属している必要があります。同じリージョンのバケット間でデータ同期を実行することはできません。
5. 現在、リージョン間同期は北京リージョンと上海リージョンの間で利用可能です。将来的に、他のリーションも適用されます。

関数を使用する際のリファレンス:

- コンソール: リージョン間レプリケーション

Back-To-Source 設定

Back-to-source設定では、複数のback-to-source読み取りメソッドが適用され、ホットデータ移行と特定の要求リダイレクト要件を満たします。

この設定により、各OSS Get要求のURLを一致させることができます。これにより、バックツーソースメソッドが指定されます。最大5つのルールを設定できます。有効な規則に一致するまで、要求は設定された順序で規則と比較されます。指定された方法は、ミラーリングまたはリダイレクトのいずれかになります。

ミラーリング

ミラーリングライトバックは、データをOSSにシームレスに移行するように設計されています。これは、ユーザーが設定したサイトまたは別のクラウド製品で既に実行されているサービスを、サービスの中断なしにOSSに移行できることを意味します。

プロセスは次のとおりです。

1. クライアントはオブジェクトのデータを要求します。
2. OSSはオブジェクトが存在しないと判断し、その要求をソースURLに転送します。
3. ソースURLはOSSを介してオブジェクトを返します.OSSはクライアントに送信されます。OSSは、将来の要求を処理するためにオブジェクトデータを同時に書き込みます。

シナリオの例

詳細なシナリオは次のとおりです。

起点サイトは新しいホットデータを生成しており、レガシーコールドデータも保存されています。

まず、移行ツールossimport2 を使用して、コールドデータをOSSに移行できます。移行中に、ユーザーはミラーリングの書き戻しを構成し、起点サイトのURLをOSSに設定できます。ドメイン名がOSSに切り替えられ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、ユーザはOSSを介してアクセスすることができ、ファイルに初めてアクセスした後にOSSに保存されます。

新しいデータを生成しないオリジンサイトのドメイン名を変更すると、サイトがスキャンされ、移行されていないすべてのデータがOSSにインポートされます。この状況では、ユーザーがミラーリングの書き戻しを無効にすることがあります。

構成された起点サイトがIPアドレスである場合、ドメイン名がOSSに移行された後も、起点サイトにデータ

をミラーリングできます。ただし、ドメイン名であれば、ドメイン名がOSSまたはCDNに解決されるため、ミラーリングは作成できません。この状況では、ユーザーは、起点サイトをミラーリングする別のドメイン名を申請することができます。このドメイン名とインサービスドメイン名は、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の移行時にオリジンサイトイメージングを続行できます。

使用規則

- OSSは、GetObject() が404コードを返すときにオリジンサイトからオブジェクトを要求するためにミラーリングライトバックのみを実行します。

  起点サイトから要求されたURLはMirrorURL + objectであり、OSSに書き戻されたファイルの名前はobjectです。たとえば、次のように仮定します。

  - バケット例-という名前でバケット 。
  - ミラーリングライトバックが設定されています。
  - MirrorURLは http://www.example.com/です。
  - ファイルobject.jpgは、この中には存在しないバケット 。
    ファイルをダウンロードするために、OSSは http://www.example.com/object.jpgへのGet要求を開始し、結果を記録し、それをユーザに返すこのファイルは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.com/dir1/のようなパス情報を運ぶ場合、プロセスは前の例と同じですが、OSSのback-to-source URLは http://www.example.com/dir1/object.jpg と表示されますが、OSSに書き込まれるオブジェクトはobject.jpgのままです。このプロセスは、オブジェクトを元のサイトディレクトリからOSSに移行するのと同じです。

  OSSに送信されるヘッダーおよびクエリーストリング情報は、元のサイトに送信されません。

  起点サイトがデータをチャンクで返す場合、OSSは同様にデータをチャンクでユーザに返します。

  OSSは、元のサイトから次のヘッダー情報を戻して保存します。

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

  “MIRROR” + space + urldecode(back-to-source URL)という値を持つ、ライトバックファイルにx-oss-tagレスポンスヘッダーが追加されます。上記の例では、これはx-oss-tag:MIRROR http%3a%2f%2fwww.example-domain.com%2fdir1%2fimage%2fexampleobject.jpg。ファイ

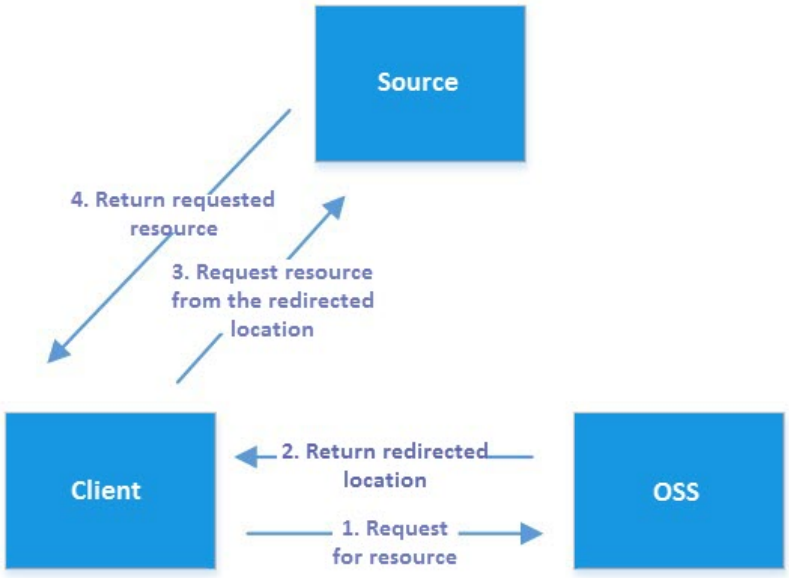
ルがOSSに書き戻された後、再度上書きされない限り、このヘッダーはミラーリングから取得されたことを示すようにダウンロードされるたびに追加されます。

元のサイト上の対応するファイルが変更された場合、ファイルがミラーリングライトバックによってOSSにすでに書き込まれていると仮定すると、OSSはOSS上に存在するファイルを更新しません。 OSSが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ソースに存在しない場合、返される結果はHTTPステータス404になり、OSSを介してユーザに転送されます。ミラーリング元が200以外のステータスコードを返す場合(ネットワーク関連の原因によるファイルの取得に失敗した場合を含む)、OSSは424をユーザーに返します。エラーコードは’ MirrorFailed’ です。

リダイレクション

URLリダイレクト機能は、ユーザー定義の条件と対応するホップ構成に基づいて、ユーザーに3xxホップを返します。 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、この操作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。

1. クライアントはオブジェクトのデータを要求します。
2. OSSは、オブジェクトが存在しないと判断し、リダイレクト・ロケーション・ソースをクライアントに返します。
3. クライアントは、要求をOSSが提供するリダイレクションの場所に直接送信します。

4. ソースはオブジェクトをクライアントに直接返します。

アプリケーションのシナリオ

データソースをOSSに移行する
ユーザーはデータをOSSに非同期で移行できます。このようにして、移行されていないデータに対する要求では、URL書き換えメソッドを使用して302リダイレクト要求がユーザーに返されます。ユーザーのクライアントは、302リダイレクト要求の場所に基づいて、ユーザーのデータソースからデータを返します。

ページリダイレクト機能の設定
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、カスタマイズされたページを訪問者に表示することができます。

404または500のエラーが発生したときにリダイレクトされたページを設定する
404または500の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エラーがユーザによって検出されないことが保証される。

参照

- コンソール: Back-To-Source ルールの設定

セキュリティマネジメント

サーバーアクセスログ

サーバーアクセスログ

OSS では、サーバーアクセスログの自動保存機能が用意されています。バケットのオーナーは、OSS コンソールにログインし、すべてのオーナーのバケットに対してサーバーアクセスログ機能を有効にすることができます。バケット (ソースバケット) でアクセスログが有効化されると、OSS は、そのバケットのすべてのアクセスリクエストログを格納するオブジェクトを (1 時間単位で) 生成し、一定の命名規則に従って、ユーザー指定のバケット (ターゲットバケット) にオブジェクトを書き込みます。

アクセスログのオブジェクト命名規則

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

この命名規則では、TargetPrefix はユーザーが指定します。YYYY、mm、DD、HH、MM、SS には、作成日時の年、月、日、時、分、秒がアラビア数字で設定されます (桁数に注意してください)。UniqueString は、OSS システムによって生成される文字列です。OSS アクセスログの格納に使用されるオブジェクトの名前の実例を以下に示します。

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

上の例では、” MyLog-“ はユーザーが指定したオブジェクトプレフィックス、” oss-example” はオリジンバケットの名前、” 2012-09-10-04-00-00” はオブジェクトの作成日時 (北京現地時間)、” 0000” は OSS システムによって生成された文字列です。

ログファイルの形式

(左から右にスペースで区切られます):

名前	例	説明
Remote IP	119.140.142.11	リクエストが開始された IP アドレス (このフィールドは、プロキシやユーザーファイアウォールによってブロックされる場合があります)
Reserved	-	予約フィールド
Reserved	-	予約フィールド
Time	[02/May/2012:00:00:04 +0800]	OSS がリクエストを受信した日時
Request-URI	“GET /aliyun-logo.png HTTP/1.1”	ユーザーがリクエストした URI (クエリ文字列を含む)
HTTP Status	200	OSS から返された HTTP ステータスコード
SentBytes	5576	ユーザーが OSS からダウンロードしたトラフィック
RequestTime (ms)	71	このリクエストが完了するまでにかかった時間 (単位はミリ秒)
Referer	http://www.aliyun.com/product/oss	リクエストされた HTTP リファラー
User-Agent	curl/7.15.5	HTTP User-Agent ヘッダー
HostName	oss-example.oss-cn-hangzhou.aliyuncs.com	アクセスリクエストのドメイン名

Request ID	505B01695037C2AF032593A4	このリクエストを一意に識別するために使用される UUID
LoggingFlag	true	アクセスログ機能が有効かどうか
Reserved	-	予約フィールド
Requester Alibaba Cloud ID	1657136103983691	リクエスト送信者の Alibaba Cloud ID (匿名アクセスの場合は “-“)
Operation	GetObject	リクエストタイプ
Bucket	oss-example	アクセスをリクエストされたバケットの名前
Key	/aliyun-logo.png	ユーザーがリクエストしたキー
ObjectSize	5576	オブジェクトサイズ
Server Cost Time (ms)	17	OSS サーバーがこのリクエストの処理に費やした時間 (単位はミリ秒)
Error Code	NoSuchBucket	OSS から返されたエラーコード
Request Length	302	ユーザーリクエストの長さ (バイト)
UserID	1657136103983691	バケットオーナーの ID
Delta DataSize	280	バケットサイズの変動 (変動がない場合は “-“)
Sync Request	-	CND からの Back-To-Source リクエストかどうかを示す。そうでない場合は “-“
Reserved	-	予約フィールド

詳細分析

1. ソースバケットとターゲットバケットは同じユーザーに属している必要があります。
2. TargetPrefix は、アクセスログの格納に使用するオブジェクトの名前のプレフィックスを示します。このフィールドは空白のままにできます。
3. ソースバケットとターゲットバケットは、同じバケットでも別のバケットでもかまいません。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保存できます (この場合は、TargetPrefix に別の値に割り当てるすることをお勧めします)。
4. バケットアクセスログファイルは 1 時間ごとに生成されますが、 その時間内のすべてのリクエストがログファイルに記録されるとは限りません。前のログファイルや次のログファイルに記録される場合もあります。
5. OSS によって生成されるログファイルの命名規則の “UniqueString” は、OSS がオブジェクトを一意に識別するために生成する UUID です。

6. バケットアクセスログファイルが生成されるたびに、この操作は PUT 操作として見なされ、占有スペースが記録されますが、生成されたトラフィックは記録されません。生成されたログファイルは、通常のオブジェクトとして操作できます。
7. “x-“というプレフィックスが付いたクエリ文字列パラメーターは、OSS ではすべて無視されますが、アクセスログには記録されます。大量のアクセスログで特定のリクエストにマークを付けるには、”x-“というプレフィックスが付いたクエリ文字列パラメーターを URL に追加できます。例:http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.pnghttp://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png?x-user=admin上の 2 つのリクエストは、OSS の処理結果は同じですが、 アクセスログで “x-user=admin” を検索すると、マークを付けたリクエストをすばやく見つけることができます。
8. OSS ログのフィールドの値が “-“になっている場合があります。これは、データが不明であるか、その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。将来的に、OSS ログファイルの末尾に必要に応じてフィールドが追加されます。ログ処理ツールを開発する際には、互換性の問題を考慮に入れることをお勧めします。

機能の使用方法のリファレンス:

- コンソール: サーバーアクセスログ

OSS Anti-leech 保護

Anti-leechの設定

OSS では、使用状況に基づいてサービス料金が徴収されます。OSS のユーザーデータを盗難から守るために、OSS では、HTTP の referrer ヘッダーフィールドに基づいた Anti-leech 策を提供しています。ユーザーは OSS コンソールにログインするか API を使用して、バケットのリファラーホワイトリストと、リファラーが空白となっているリクエストによるアクセスを許可するかどうかを設定できます。たとえば、oss-example という名前のバケットの場合は、リファラーホワイトリストを "https://jp.aliyun.com" に設定します。このように設定すると、"https://jp.aliyun.com" というリファラーを使ったリクエストのみがバケット内のオブジェクトにアクセスできるようになります。

詳細分析

Anti-leech 検証は、ユーザーが URL 署名を介して、または匿名でオブジェクトにアクセスする場合にのみ実行されます。リクエストヘッダーに “Authorization” フィールドが含まれる場合、Anti-leech 検証は実行されません。

バケットでは、コンマ “,” で区切られた複数のリファラーフィールドがサポートされています。

リファラーフィールドでは、ワイルドカード “* ” および “?” がサポートされています。

空のリファラーフィールドを使ったアクセスリクエストを許可するかどうかを設定できます。

ホワイトリストが空の場合、リファラーフィールドが null かどうかがチェックされません (そうでない場合、すべてのリクエストが拒否されます)。

ホワイトリストが空ではなく、ルールでリファラーフィールドが null の状態が許可されていない場合は、ホワイトリストにリファラーがあるリクエストだけが許可されます。その他のリクエスト (リファラーが null のリクエストを含む) は拒否されます。

ホワイトリストが空ではなく、ルールが空のリファラーフィールドを許可する場合は、リファラーが空のリクエストおよびホワイトリストにリファラーがあるリクエストが許可されます。その他のリクエストは拒否されます。

3 つのバケット権限 (非公開、公開読み取り、および公開読み書き) によって、すべてのリファラーフィールドがチェックされます。

ワイルドカードの詳細:

アスタリスク “”: アスタリスクを使用して、0 個または複数の文字を表すことができます。プレフィックス AEW が付加されたオブジェクト名を探しているものの、オブジェクト名の残りの部分を忘れてしまった場合、「AEW*」と入力すると、AEWT.txt、AEWU.EXE、AEWI.dll など、AEW で始まるすべてのタイプのファイルを検索することができます。検索範囲を狭くする場合は、「AEW*.txt」と入力すると、AEWIP.txt や AEWDF.txt など、AEW で始まるすべての .txt ファイルを検索することができます。

疑問符 “?”: 疑問符を使用して、1 文字を表すことができます。「love?」と入力すると、lovey や lovei など、love で始まり 1 文字で終わるすべてのタイプのファイルが表示されます。検索範囲を狭くする場合は、「love?.doc」と入力すると、lovey.doc や loveh.doc など、love で始まり 1 文字で終わるすべての .doc ファイルを検索することができます。

機能の使用方法のリファレンス:

- API: PutBucketReferer
- コンソール: Anti-leech 設定

クロスオリジンアクセス設定

クロスオリジンアクセス、または JavaScript のクロスオリジンは、セキュリティのためのブラウザー制限セット、つまり同一オリジンポリシーです。Web サイト A がその Web ページで JavaScript コードを使用して Web サイト B にアクセスしようとすると、A と B は別のオリジンの 2 つの Web サイトであるため、この操作はブラウザーによって拒否されます。

クロスオリジンアクセスのニーズは、ユーザーの Web サイト www.a.com のバックエンドで OSS が使用される場合など、実際の場面で頻繁に生じます。JavaScript で実装されたアップロード機能が Web ページで提供されている場合、 リクエストは Web ページで www.a.com にのみ送信でき、他の Web サイトに送信されるすべてのリクエストはブラウザーによって拒否されます。したがって、ユーザーがアップロードしたデータは www.a.com 経由で他のサイトにリレーする必要があります。クロスオリジンアクセスが設定されている場合、www.a.com 経由でリレーするのではなく OSS に直接アップロードできます。

CORS (Cross-origin Resource Sharing) は、HTML5 により提供される標準のクロスオリジンソリューションです。現在、CORS 標準は、OSS によってクロスオリジンアクセスの目的でサポートされています。特定の CORS ルールの詳細については、「W3C CORS の規範」を参照してください。

要約すると、CORS では HTTP リクエストのオリジンを含むヘッダーを使用して、リクエスト元のオリジンを示します。前の例では、オリジンヘッダーは www.a.com です。リクエストを受信した後、サーバーは特定のルールに基づいてリクエストが受け入れ可能かどうかを判断します。可能な場合、サーバーが応答に Access-Control-Allow-Origin ヘッダーをアタッチします。ヘッダーには、クロスオリジンアクセスが許可されることを示す www.a.com が含まれます。サーバーがすべてのクロスオリジンリクエストを受け入れる場合は、Access-Control-Allow-Origin ヘッダーを * に設定します。

ブラウザーでは、対応するヘッダーが返されるかどうかに基づいて、クロスオリジンリクエストが成功するかどうかが決まります。対応するヘッダーがアタッチされていない場合、ブラウザーはリクエストをブロックします。リクエストが単純ではない場合、ブラウザーでは、最初に OPTIONS リクエストを送信し、サーバーの CORS 設定を取得します。サーバーが後続の操作をサポートしていない場合、ブラウザーでも後続の操作がブロックされます。

OSS では、対応するクロスオリジンリクエストの受け入れや拒否を必要に応じて実行する CORS ルールを設定できます。このルールは、バケットレベルで設定します。詳細については、「PutBucketCORS」を参照してください。

キーポイント

関連する CORS ヘッダーのアタッチとその他の操作は、ブラウザーによって自動的に実行され、ユーザーが追加で操作する必要はありません。ブラウザー環境でのみ、CORS の操作が意味を持ちます。

CORS リクエストを受け入れるかどうかは、OSS 認証とその他の認証方法とはまったく関係ありま

せん。つまり、OSS CORS ルールは、関連する CORS ヘッダーをアタッチするかどうかを決定する目的でのみ使用されます。リクエストをブロックする必要性は、ブラウザーだけで決定されるようにする必要があります。

クロスオリジンリクエストを使用する場合は、ブラウザーのキャッシュ機能が有効であることを確認します。たとえば、同じブラウザーで実行されている 2 つの Web ページが、それぞれ同じクロスオリジンリソースを同時にリクエストした場合を考えてみます (リクエスト元は www.a.com と www.b.com)。www.a.com のリクエストが最初にサーバーによって受信されると、サーバーはユーザーに Access-Control-Allow-Origin ヘッダー “www.a.com” を含めてリソースを返します。www.b.com からリクエストが開始されると、ブラウザーは以前キャッシュされたリクエストをユーザーに返します。ヘッダーコンテンツが CORS リクエストと一致しないため、後のリクエストは失敗します。

機能の使用方法のリファレンス

- API: Cross-origin Resource Sharing
- SDK: Java SDK-『Cross-origin Resource Sharing』
- コンソール: Cross-origin Resource Sharing

サーバー側の暗号化

OSS では、ユーザーがアップロードしたデータのサーバー側の暗号化がサポートされています。ユーザーがデータをアップロードすると、OSS は受信したユーザーデータを暗号化し、暗号化されたデータを永続的に格納します。ユーザーがデータをダウンロードするとき、OSS は暗号化されたデータを自動的に暗号化解除して、元のデータをユーザーに返し、データがサーバー側で暗号化されたことを返された HTTP リクエストヘッダーで宣言します。つまり、OSS はコーデック処理全体を管理するため、暗号化されたオブジェクトのダウンロードと一般的なオブジェクトのダウンロードは差が大きくありません。現在、OSS によるサーバー側の暗号化は、オブジェクトの属性です。ユーザーがオブジェクトを作成するときは、HTTP ヘッダー “x-oss-server-side-encryption” を Put Object リクエストに追加し、その値を “AES256” と指定するだけで済みます。その後、オブジェクトを格納する前に、サーバー側でオブジェクトを暗号化することができます。現在、OSS によるサーバー側の暗号化は、次の操作でサポートされています。

- Put Object
- Copy Object
- Initiate Multipart Upload

詳細分析

1. OSS が受信したリクエスト (Put Object、Copy Object、および Initiate Multipart Upload リク

エスト以外) に “x-oss-server-side-encryption” ヘッダーが含まれる場合、OSS は、メッセージ本文にエラーコード InvalidArgument を入れて、HTTP ステータスコード 400 を直接返します。
2. 現在、OSS では、AES256 暗号化アルゴリズムのみがサポートされています。ユーザーが “x-oss-server-side-encryption” ヘッダーにその他の値を指定した場合、OSS は、メッセージ本文にエラーコード “InvalidEncryptionAlgorithmError” を入れて、HTTP ステータスコード 400 を直接返します。
3. サーバー側で暗号化してから格納されたオブジェクトの場合、OSS は、値をエントロピ暗号化アルゴリズムにして、以下の API リクエストで x-oss-server-side-encryption ヘッダーを返します。
    - Put Object
    - Copy Object
    - Initiate Multipart Upload
    - Upload Part
    - Complete Multipart Upload
    - Get Object
    - Head Object

具体的な実装

- API: Append Object
- API: Put Object
- API: Copy Object
- API: Post Object

静的 Web サイトホスティング

静的 Web サイトホスティング

OSS では、静的 Web サイトホスティングがサポートされています。ユーザーは OSS コンソールにて、ストレージ容量を静的 Web サイトホスティングモードで機能するよう設定できます。例えば、設定が有効になると、杭州リージョンのバケットでホストされている静的 Web サイトのドメイン名は、次のようになります。

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

ユーザーが OSS にホストされている静的 Web サイトをより簡単に管理できるようにするために、OSS では

、次の 2 つの機能が提供されます。

- 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名に直接アクセスしたときに OSS によって返されるデフォルトのインデックスドキュメント (Web サイトの index.html と同じです) を指します。バケットに静的 Web サイトホスティングモードを設定した場合は、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 (最も一般的なエラーは 404 “NOT FOUND”) が発生した場合、OSS によってユーザーに返されるエラーページを指します。エラーページを指定することにより、ユーザーに適切なエラープロンプトを提供できるようになります。

たとえば、ユーザーが次のように設定します。

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

したがって、

- http://oss-sample.oss-cn-hangzhou.aliyuncs.com/ および http://oss-sample.oss-cn-hangzhou.aliyuncs.com/directory/ にアクセスする場合は、http://oss-sample.oss-cn-hangzhou.aliyuncs.com/index.html にアクセスするのと同じになります。
- http://oss-sample.oss-cn-hangzhou.aliyuncs.com/object にアクセスした際に、オブジェクトが存在しなかった場合は、OSS は http://oss-sample.oss-cn-hangzhou.aliyuncs.com/error.html を返します。

詳細分析

- 静的 Web サイトとは、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) で構成されている Web サイトです。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) はサポートされていません。
- ユーザーが定義したドメイン名を使用して、バケットベースの静的 Web サイトにアクセスするには、CNAME ドメイン名を使用できます。
- OSS では、バケットドメイン名によるアクセスが制限されるため、ブラウザーでユーザーのファイルを直接表示することはできません。ユーザーには、CNAME の使用が推奨されます。
- 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と、静的 Web サイトのルートドメイン名への匿名アクセスに対してはインデックスページが返され、静的 Web サイトのルートドメイン名への署名付きアクセスに対しては Get Bucket の結果が返されるようになります。
- バケットに静的 Web サイトホスティングモードが設定されていて、ユーザーが静的 Web サイトまたは存在しないオブジェクトのルートドメイン名にアクセスする場合、OSS は指定されたオブジェクトをユーザーに返し、返されたトラフィックおよびリクエストに応じて課金します。

関数を使用する際のリファレンス:

- API: - API: Put Bucket Website
- コンソール: 静的 Web サイトホスティング

イメージ処理

Image Processing (IMG) は、大量のボリューム処理、セキュリティ、低コスト、高信頼性を特長とする Alibaba Cloud OSS によって外部ユーザーに提供されるサービスです。OSS にソースイメージをアップロードして保存することで、簡単な RESTful インターフェイスを通じていつでも、どこでも、どんなインターネットデバイスでもイメージを処理できます。IMG は、画像処理、ウォーターマーク、パイプライン、画像スタイル操作を提供します。画像処理サービスは、画像処理インターフェースを提供します。イメージアップロードの場合は、OSS アップロードインターフェイスを使用します。画像処理サービスに基づいて画像関連サービスを設定することができます。

Image Processing Service には次の機能があります。

- 画像のスケーリング、トリミング、回転
- 画像、テキストの追加、およびテキストと画像のウォーターマークを画像への追加
- 画像フォーマットの変換
- イメージ処理スタイルのカスタマイズ
- 複数の画像処理機能をパイプラインを介して設定された順序で呼び出す
- 画像情報の取得

多くの機能と詳細な紹介については、Image Processing Service Documentationを参照してください。

モニタリングサービス

モニタリングサービスの概要

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ステータス、パフォーマンス、計測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ご利用いただけば、リクエストの追跡、使

用率の分析、ビジネストレンドに関する統計の収集、システム障害を迅速に発見および診断できます。

OSS メトリックの指標は、基本サービス指標、パフォーマンス指標、計測指標などの指標グループに分類されます。詳細については、「モニタリング指標リファレンス」を参照してください。

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、実際の影響を表示して、ビジネスシナリオの分析と評価に関する知見を提供できます。OSS リアルタイムメトリック指標 (計測指標を除く) を使用して、1 分未満の出力遅延でメトリックデータを分レベルで収集および集計できます。

計測指標の説明

計測指標では、次の機能を使用します。

- 計測エントリが時間レベルで収集、集計、出力されます。ただし、出力は最大 30 分遅延する可能性があります。
- 計測時刻は、関連する統計期間の開始時刻を表します。
- 計測データ取得終了時刻は、当月最後の計測データ統計期間の終了時刻です。当月に計測データが生成されない場合、計測データ取得終了時刻は、当月初日の午前 0 時になります。
- 計測エントリの最大値はプッシュされて表示されます。正確な計測データについては、Billing Management に移動し、[**使用状況レコード**] をクリックします。

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できます。

アラームルールは、他の計測指標に設定して、アラームモニタリングに追加することができます。また、単一のメトリックの指標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」を参照してください。
- 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」を参照してください。
- OSS メトリック指標の詳細については、「モニタリング指標リファレンス」を参照してください。

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、有効期限になると自動的にクリアされます。メトリックデータをオフラインで分析したり、履歴メトリックデータを 31 日より長く格納したりするには、適切なツールまたは入力コードを使用して、Cloud Monitor のデータストレージを読み取ります。詳細については、「OpenAPI を通じたメトリックデータのアクセス」を参照してください。

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。詳細については、「OpenAPI を通じたメトリックデータのアクセス」を参照してください。

OpenAPI を通じたメトリックデータのアクセス

Cloud Monitor の OpenAPI では、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、以下を参照してください。

*Cloud Monitor OpenAPI ユーザーマニュアル

- 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 の実行ステータス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。

追跡と診断
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と、追跡および診断用に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
典型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。

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、削除されたバケットと同じ名前で別のバケットを作成した場合、削除したバケットに設定されたモニタリングおよびアラームルールが新しいバケットに適用されます。

モニタリングサービスユーザーガイド

モニタリングサービスユーザーガイド

Cloud Monitor コンソール

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 コンソールで利用できます。OSS モニタリングサービスは、以下のいずれかの方法でアクセスできます。

Alibaba Cloud アカウントにログインしてから、ホームページで左側のナビゲーションバーの [**Object Storage Service**] に移動します。OSS の [概要] ページで、右側の [**モニターパネル**] をクリックします (下図を参照)。

Alibaba Cloud アカウントにログインしてから、ホームページで左側のナビゲーションバーの [Cloud Monitor] に移動します。Cloud Monitor の [概要] ページで、[**クラウドサービスモニターリング**] > [**OSS**] をクリックします (下図を参照)。

[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] ページは自動更新されません。最新データを表示するには、右上の [**更新**] 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、次の情報が表示されます。

- ユーザーモニタリング情報
- 最新の月間統計
- ユーザーレベルのメトリック指標

ユーザーモニタリング情報

このモジュールには、バケットおよび関連アラームルールの総数が表示されます。

パラメーターは次のとおりです。

- [**バケット数**] の横の数字をクリックすると、作成したすべてのバケットが表示されます。
- [**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、詳細の情報が表示されます。
  - [**アラームルール数**] は、設定したアラームルールの総数を参照します。
  - [**アラーム中**] は、アラーム状態のアラームを参照します。
  - [**禁止数**] は、現在無効になっているアラームを参照します。
  - [**アラーム済み**] は、最近アラーム状態に変更されたアラームを参照します。

最新の月間統計

このモジュールは、当月の初日の午前 0 時から計測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。次の指標が表示されます。

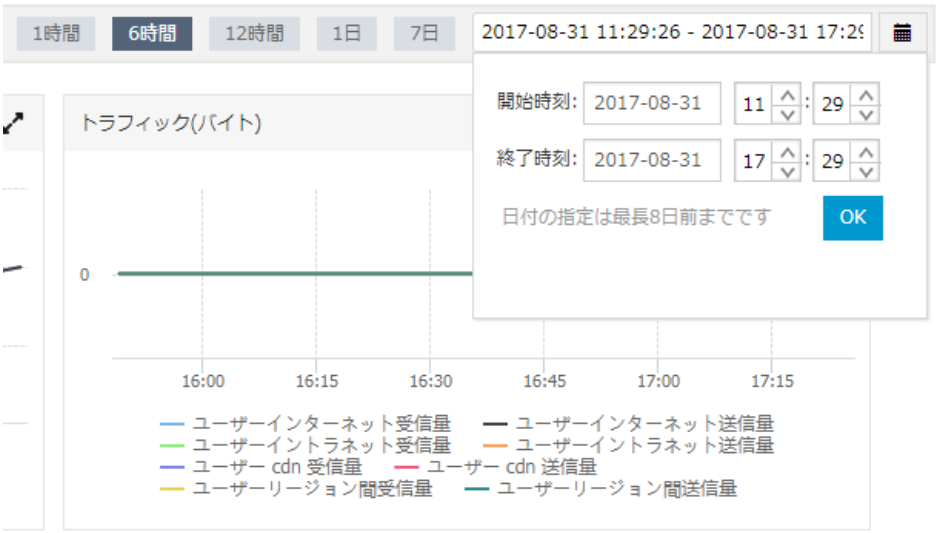
- ストレージ使用率
- インターネットトラフィック
- Put リクエスト
- Get リクエスト

ユーザーレベルのメトリック指標

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を表示し、[**サービス監視概要**] と [**クエリ状態詳細**] から構成されます。

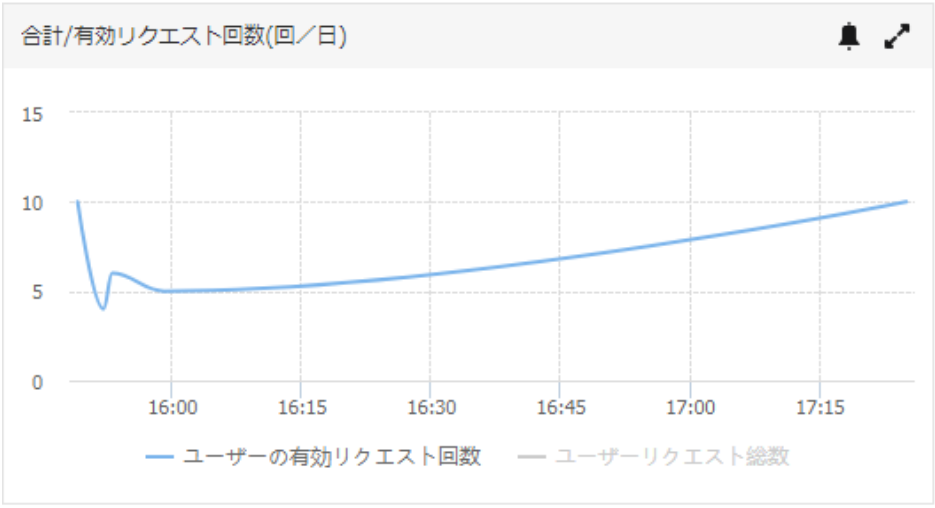
事前に定められた時間範囲を選択するか、カスタム時間ボックスで時間範囲を定義して、対応するメトリックチャートまたは表を表示できます。

- 1 時間、6 時間、12 時間、1 日、7 日の時間範囲オプションを使用できます。デフォルトのオプションは、1 時間です。
- カスタム時間ボックスでは、開始時刻と終了時刻を分レベルで定義できます。

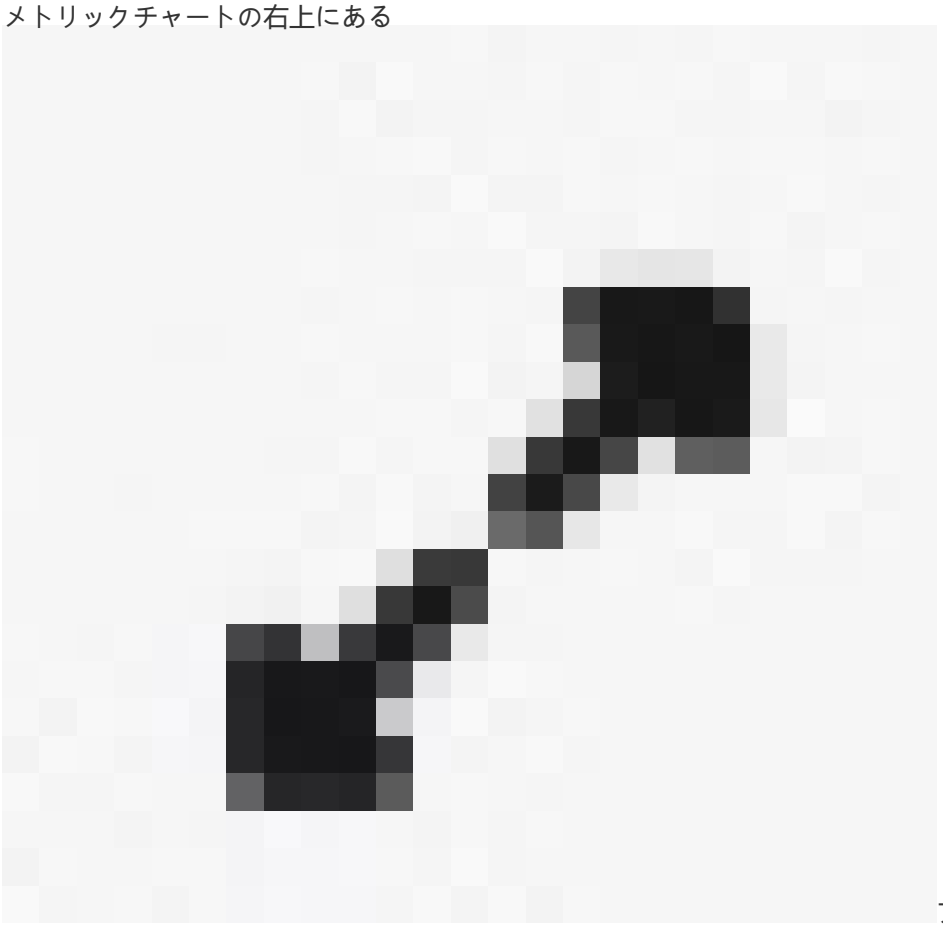


メトリックチャート/表では、次の表示モードを使用できます。

- 凡例の非表示: 凡例をクリックして、対応する指標曲線を非表示にできます (下図を参照)

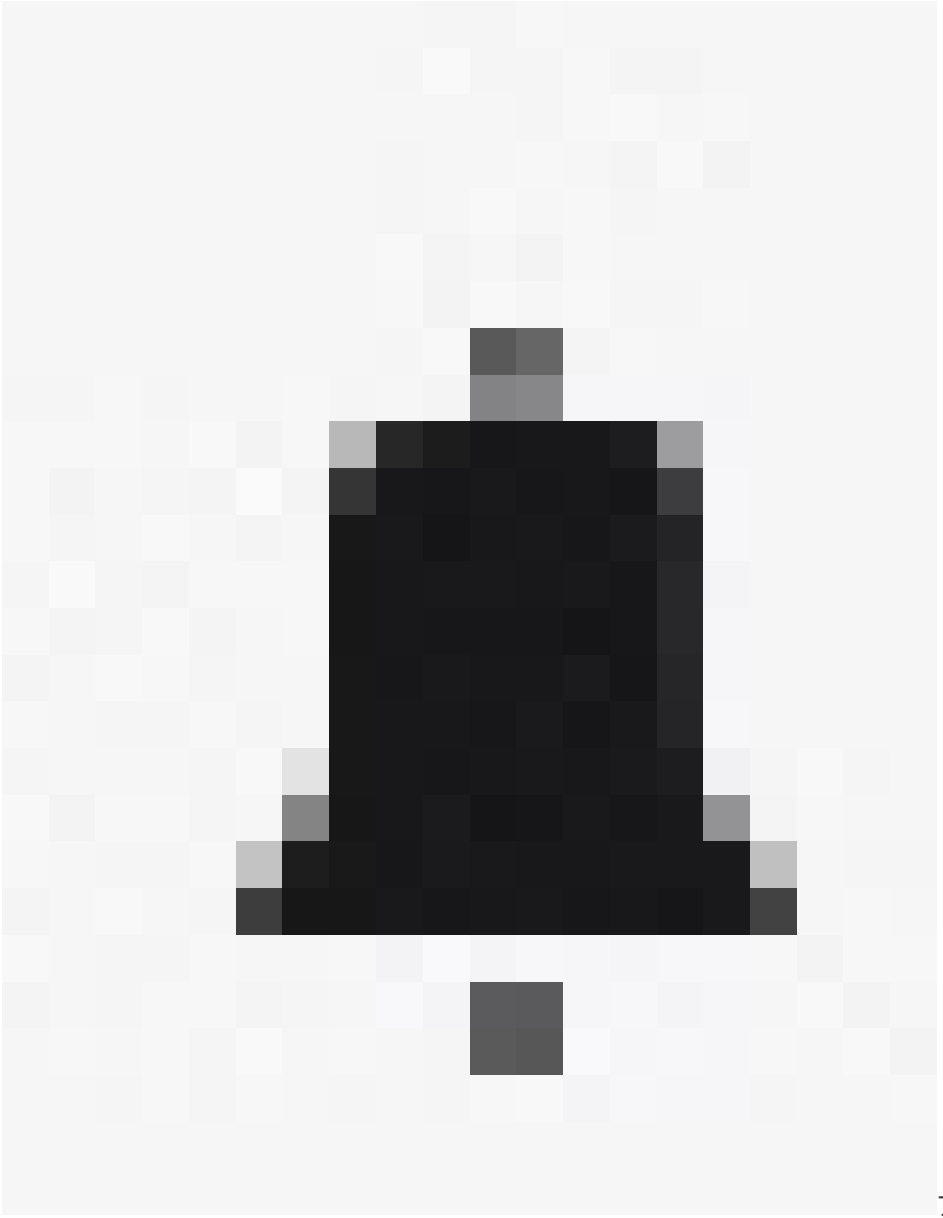


- メトリックチャートの右上にある



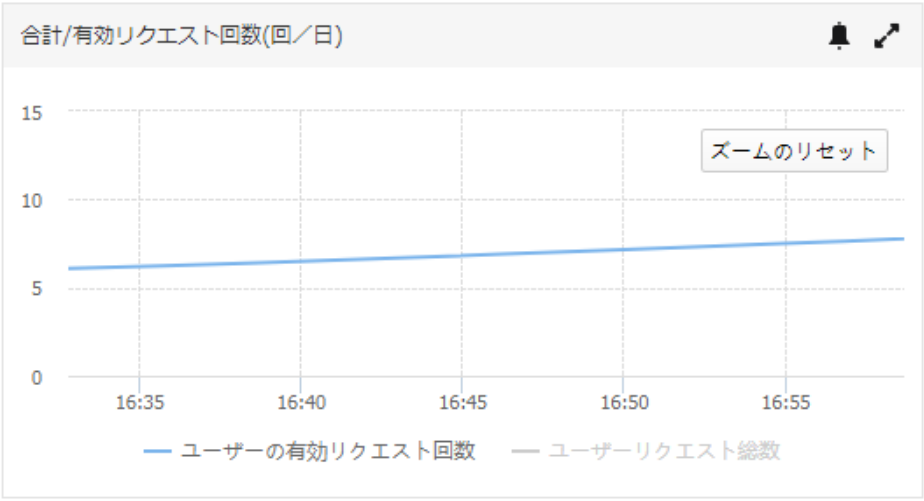
アイコンをクリックすると、チャートが拡大されます。注意: 表は拡大できません。

- 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指標のアラームルールを設定できます。詳細については、「アラームサービスユーザーガイド」を参照してください。

注意: 表および計測参照指標のアラームルールは設定できません。

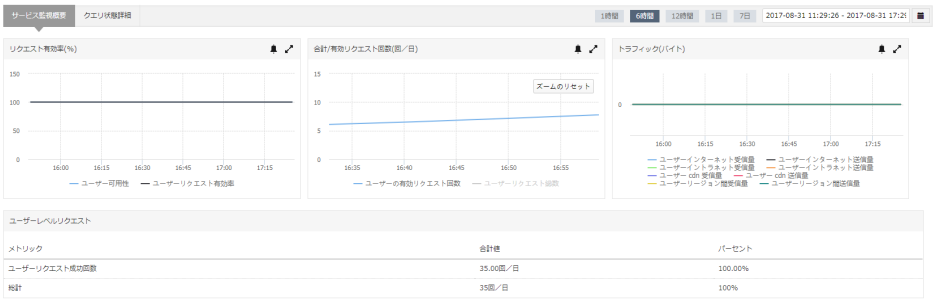
- カーソルをチャートの曲線領域の内部に置き、マウスの左ボタンを押しながらマウスをドラッグすると、時間範囲が拡張されます。[**ズームのリセット**] をクリックすると、元の時間範囲に戻ります。



サービス監視概要

[**サービス監視概要**] ページには、次の主要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/有効リクエスト率。これには、可用性と有効なリクエストの割合の 2 つのメトリック指標が含まれます。
- ユーザーレベルのリクエスト/有効リクエスト。これには、リクエストの総数と有効なリクエストの数の 2 つのメトリック指標が含まれます。
- ユーザーレベルのトラフィック。これには、インターネットアウトバウンドトラフィック、インターネットインバウンドトラフィック、イントラネットアウトバウンドトラフィック、イントラネットインバウンドトラフィック、CDN アウトバウンドトラフィック、CDN インバウンドトラフィック、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、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 8 つのメトリック指標が含まれます。
- ユーザーレベルのリクエスト状態分布。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表です。

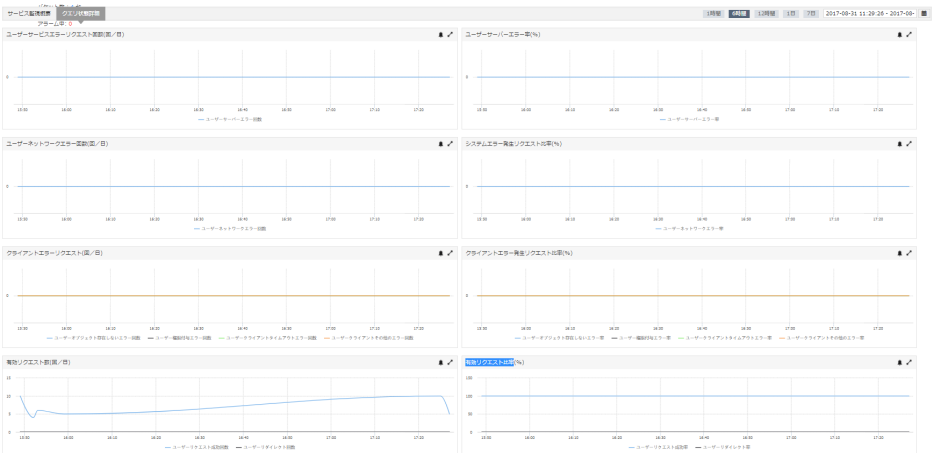


クエリ状態詳細

[**クエリ状態詳細**] ページでは、リクエスト状態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。

- ユーザーサービスエラーリクエスト回数
- ユーザーサーバーエラー率
- ユーザーネットワークエラー回数

- システムエラー発生リクエスト比率
- クライアントエラーリクエスト。これには、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 つのメトリック指標が含まれます。
- クライアントエラー発生リクエスト比率。これには、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 つのメトリック指標が含まれます。
- 有効リクエスト数。これには、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 つのメトリック指標が含まれます。
- 有効リクエスト比率。これには、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 つのメトリック指標が含まれます。



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、当月の計測統計、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。

- 当月の計測統計には、各バケットのストレージサイズ、インターネットアウトバウンドトラフィック、Put リクエスト数、Get リクエスト数が表示されます。
- [モニターリングチャット] 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の [アラームルール] をクリックするか [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 (ファジーマッチを利用できます)。
- 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て、[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。詳細については、「アラームサービスユーザーガイド」を参照してください。

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 [**モニターリングチャット**] をクリックすると、バケットモニタリングビューに移動します。

バケットモニタリングビューには、次の 6 つの指標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
- クエリ状態詳細
- 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

測定参照以外の指標は、60 秒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デフォルト時間範囲は過去 6 時間ですが、ユーザーレベルのメトリックチャートでは過去 1 時間です。左上にある [**バケットリストに戻る**] をクリックすると、[**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

この指標グループは、ユーザーレベルのサービスモニタリング概要と同様ですが、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。

- 有効リクエスト可用性。これには、可用性と有効なリクエストの割合の 2 つのメトリック指標が含まれます。
- 合計/有効リクエスト。これには、リクエストの総数と有効なリクエストの数の 2 つのメトリック指標が含まれます。
- オーバーフロー。これには、インターネットアウトバウンドトラフィック、インターネットインバウンドトラフィック、イントラネットアウトバウンドトラフィック、イントラネットインバウンドトラフィック、CDN アウトバウンドトラフィック、CDN インバウンドトラフィック、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、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 8 つのメトリック指標が含まれます。
- 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表です。

クエリ状態詳細

この指標グループは、ユーザーレベルのリクエスト状態詳細と同様ですが、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率
- クライアントエラーリクエスト回数。これには、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 つのメトリック指標が含まれます。
- クライアントエラーリクエスト率。これには、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 つのメトリック指標が含まれます。
- リダイレクトリクエスト回数。これには、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 つのメトリック指標が含まれます。
- 成功リダイレクト率。これには、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 つのメトリック指標が含まれます。

測定参照

計測参照グループには、計測指標が 1 時間単位の収集および表示粒度で表示されます (下図を参照)。

計測メトリックチャートは次のとおりです。

- 割当サイズ
- フローの計測
- Billing requests。これには、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、収集されたデータが 30 分以内に表示されます。

平均レイテンシ

この指標グループには、API モニタリングの平均レイテンシ指標が含まれます。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。下図のように設定します。

最大レイテンシ

この指標グループには、API モニタリングの最大レイテンシ指標が含まれます。メトリックチャートは次のとおりです。

- 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

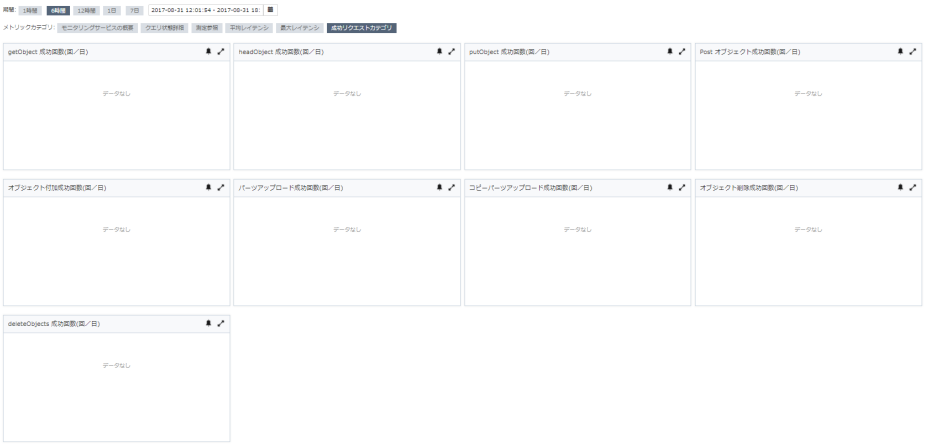
各メトリックチャートには、対応する最大 E2E レイテンシと最大サーバーレイテンシが表示されます。下図のように設定します。

成功リクエストカテゴリ

この指標グループには、API モニタリングの正常なリクエスト数の指標が含まれます。メトリックチャートは次のとおりです。

- 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

下図のように設定します。



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 (下図を参照)。

[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」を参照してください。

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」の関連する章を参照してください。

メトリック項目リファレンス

メトリック項目リファレンス

ここでは、OSS モニタリングサービスのメトリックデータにアクセスするために OpenAPI または Cloud Monitor 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メトリックデータは、同じプロジェクト名 acs_oss を使用します。

Java SDK で記述されたサンプルコードを次に示します。

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

StartTime と EndTime

Cloud Monitor の時間パラメーター値範囲は、[StartTime, EndTime] という形式です。StartTime に発生したデータは収集されませんが、EndTime に発生したデータにはアクセスできます。

Cloud Monitor の保持ポリシーでは、データを 31 日間保持するように指定されています。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できません。

他の時間パラメーターの詳細については、Cloud Monitor API の説明 を参照してください。

Java SDK で記述されたサンプルコードを次に示します。

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

ディメンション

OSS メトリック項目は、アプリケーションのシナリオに基づいてユーザーレベルまたはバケットレベルに分類されます。ディメンションの値は、これらの異なるレベルでのメトリックデータのアクセスに関して異なります。

- ユーザーレベルのメトリックデータにアクセスする場合、ディメンションを設定する必要はありません。

  バケットレベルのメトリックデータへのディメンションアクセスは次のように設定します。

  ```
  {"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。

注意: ディメンションは JSON 文字列であり、OSS メトリック指標に対するキーと値のペアを 1 つだけ含みます。

Java SDK で記述されたサンプルコードを次に示します。

```
```

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

期間

計測指標を除き、すべての OSS メトリック指標のデフォルトの集計粒度は 60 秒です。計測指標のデフォルトの集計粒度は 3,600 秒です。

Java SDK で記述されたサンプルコードを次に示します。

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

メトリック

「モニタリング指標リファレンス」では次のメトリック項目が説明されています。

メトリック	メトリック項目名	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
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	バイト	ユーザーレベル

	ョンのアウトバウンドトラフィック		
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
UserClientTimeoutErrorRate	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
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
ClientTimeoutErrorRate	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
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	ミリ秒	バケットレベル

	テンシ		
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
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
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

次の表では、集計粒度が 3,600 秒である計測指標のメトリック項目の一覧を示します。

メトリック	メトリック項目名	単位	レベル
MeteringStorageUtilization	ストレージのサイズ	バイト	返されるメトリックデータは、ディメンションが設定されている場合はバケットレベルに属し、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。

MeteringGetRequest	Get リクエスト数	回	返されるメトリックデータは、ディメンションが設定されている場合はバケットレベルに属し、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。
MeteringPutRequest	Put リクエスト数	回	返されるメトリックデータは、ディメンションが設定されている場合はバケットレベルに属し、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	返されるメトリックデータは、ディメンションが設定されている場合はバケットレベルに属し、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	返されるメトリックデータは、ディメンションが設定されている場合はバケットレベルに属し、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	返されるメトリックデータは、ディメンションが設定されている場合はバケットレベルに属し、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 で記述されたサンプルコードを次に示します。

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

モニタリング指標リファレンス

OSS 指標は、アプリケーションのシナリオに基づいてユーザーレベルまたはバケットレベルでモニターできます。

一般的な時系列メトリック指標に加えて、ユーザーがメトリックデータおよび課金ポリシーの一致を簡単に確認できるように、システムは既存のメトリック指標を分析して統計情報を収集します。1 か月間のリクエストステータス分布と計測統計など、指定された期間についての統計指標が提供されます。このリファレンスガイドでは、指標について詳細に説明します。

計測指標と統計指標を除くすべての指標は、分レベル (1 分単位) で収集されます。計測指標は時間レベル (1 時間単位) で収集されます。

ユーザーレベルの指標

ユーザーレベルの指標は、3 つのモニタリング指標詳細、つまり当月の計測統計、サービスモニタリング概要、リクエスト状態詳細で構成されます。

当月の計測統計

当月の計測統計は、月の初日の午前 0 時から、同じ月について指定されている計測終了時刻まで収集されます。

現在使用できる計測指標の詳細は次のとおりです。

指標	単位	説明
ストレージサイズ	バイト	計測統計収集期限の前に指定されたユーザーの全バケットによって占有されていた合計ストレージのサイズ
インターネットアウトバウンドトラフィック	バイト	当月の初日の 00:00 から計測統計収集期限までの、ユーザーの合計インターネットアウトバウンドトラフィック。
Put リクエスト	回	当月の初日の 00:00 から計測統計収集期限までの、ユーザーの Put リクエストの合計回数。
Get リクエスト	回	当月の初日の 00:00 から計測統計収集期限までの、ユーザーの Get リクエストの合計回数。

サービスモニタリング概要

サービスモニタリング概要の指標は、基本的なサービスの指標です。サービスモニタリング概要の指標の詳

細は次のとおりです。

指標	単位	説明
可用性	%	ストレージサービス使用のシステムの可用性を示す指標。次の式で取得されます: 可用性 = 1 - 全リクエストのうちサーバーエンドエラー (リターンコードが 5xx) のリクエストの割合。
有効リクエスト率	%	全リクエストに対する有効なリクエストの割合。有効なリクエストの詳細については、後の説明を参照してください。
リクエスト数	回	OSS サーバーによって受信されて処理されたリクエストの合計数
有効リクエスト数	回	リターンコードが 2xx または 3xx であるリクエストの合計数。
インターネットアウトバウンドトラフィック	バイト	ダウンストリームインターネットトラフィック
インターネットインバウンドトラフィック	バイト	アップストリームインターネットトラフィック
イントラネットアウトバウンドトラフィック	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
イントラネットインバウンドトラフィック	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 アウトバウンドトラフィック	バイト	CDN 高速化サービスが有効になっているときのダウンストリーム CDN トラフィック、つまり Back-To-Source トラフィック
CDN インバウンドトラフィック	バイト	CDN 高速化サービスが有効になっているときのアップストリーム CDN トラフィック
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	リージョン間レプリケーション機能が有効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るダウンストリームトラフィック
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	リージョン間レプリケーション機能が有効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るアップストリームトラフィック

リクエストの状態の詳細

リクエストの状態の詳細指標は、異なる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く要求されたモニタリング情報です。リクエストの状態の詳細指標の詳細は次のとおりです。

指標	単位	説明
サーバーサイトエラーリクエスト数	回	リターンコード 5xx によって示されるシステムレベルエラーになったリクエストの合計数
サーバーサイトエラーリクエスト率	%	全リクエストのうちサーバーエンドエラーになったリクエストの割合
ネットワークエラーリクエスト数	回	HTTP ステータスコードが 499 であるリクエストの合計数
ネットワークエラーリクエスト率	%	全リクエストのうちネットワークエラーになったリクエストの割合
クライアントエンド権限付与エラーリクエスト数	回	リターンコードが 403 であるリクエストの合計数
クライアントエンド権限付与エラーリクエスト率	%	全リクエストのうちクライアントエンド権限付与エラーになったリクエストの割合
リソース不明のクライアントエンドエラーリクエスト数	回	リターンコードが 404 であるリクエストの合計数
リソース不明のクライアントエンドエラーリクエスト率	%	全リクエストのうちリソース不明のクライアントエンドエラーになったリクエストの割合
クライアントエンドタイムアウトエラーリクエスト数	回	リターンステータスコードが 408 であるリクエストまたはリターン OSS エラーコードが RequestTimeout であるリクエストの合計数
クライアントエンドタイムアウトエラーリクエスト率	%	全リクエストのうちクライアントエンドタイムアウトエラーになったリクエストの割合
他のクライアントエンドエラーリクエスト数	回	リターンコード 4xx によって示される他のクライアントエンドエラーになったリクエストの合計数
他のクライアントエンドエラーリクエスト率	%	全リクエストのうち他のクライアントエンドエラーになったリクエストの割合
正常リクエスト数	回	リターンコードが 2xx であるリクエストの合計数。
正常リクエスト率	%	全リクエストに対する正常なリ

		クエストの割合
リダイレクトリクエスト数	回	リターンコードが 3xx であるリクエストの合計数。
リダイレクトリクエスト率	%	全リクエストに対するリダイレクトリクエストの割合

バケットレベルの指標

バケットレベルの指標は、特定のバケットの OSS 操作をモニターするために使用され、ビジネスシナリオに使用できます。アカウントレベルでモニターできるサービスモニタリング概要やリクエスト状態詳細などの当月計測統計および基本サービスの指標項目と同様に、バケットレベルの指標にも、計測参照、レイテンシ、正常リクエスト操作カテゴリなどの計測指標とパフォーマンス指標が含まれます。

サービスモニタリング概要

ユーザーレベルの説明と同様に、サービスモニタリング概要の指標は基本的な指標ですが、バケットレベルで表示されるメトリックデータを使用します。

リクエストの状態の詳細

ユーザーレベルの説明と同様に、リクエスト状態詳細の指標は、バケットレベルで表示されるメトリックデータを使用します。

当月の計測統計

統計方法はユーザーレベルでの当月計測統計で示されているものと似ていますが、前者はバケットレベルでリソース使用率の統計を収集します。バケットレベルでの当月計測統計の詳細は次のとおりです。

指標	単位	説明
ストレージサイズ	バイト	計測統計収集期限の前に指定されたバケットによって占有されていたストレージのサイズ
インターネットアウトバウンドトラフィック	バイト	当月の初日の 00:00 から計測統計収集期限までの、指定されたバケットの合計インターネットアウトバウンドトラフィック。
Put リクエスト	回	当月の初日の 00:00 から計測統計収集期限までの、指定されたバケットの Put リクエストの合計数。
Get リクエスト	回	当月の初日の 00:00 から計測統計収集期限までの、指定されたバケットの Get リクエストの合計数。

計測指標

計測指標は時系列順にモニターされ、時間レベルで収集および集計されます。計測指標の詳細は次のとおりです。

指標	単位	説明
ストレージサイズ	バイト	指定されたバケットによって使用されたストレージの 1 時間の平均サイズ
インターネットアウトバウンドトラフィック	バイト	指定されたバケットの 1 時間の合計インターネットアウトバウンドトラフィック。
Put リクエスト数	回	指定されたバケットの 1 時間の Put リクエストの合計回数。
Get リクエスト数	回	指定されたバケットの 1 時間の Get リクエストの合計回数。

レイテンシ

リクエストレイテンシは、システムのパフォーマンスを直接反映し、平均レイテンシと最大レイテンシの 2 つの指標を使ってモニターされます。指標は分レベルで収集および集計されます。

さらに、OSS API のリクエスト操作タイプに基づいて指標を分類し、異なる操作に応答するシステムのパフォーマンスをより明確に反映させることができます。現在モニターされるのは、バケット関連操作 (メタ操作を除きます) でのデータ操作を含む API だけです。

さらに、パフォーマンスのホットスポットと環境の問題の分析を容易にするため、E2E とサーバーの 2 つの異なるリンクからレイテンシモニタリング指標が収集されます。

- E2E レイテンシは、OSS に正常なリクエストを送信するときの E2E レイテンシのことであり、リクエストの読み取り、応答の送信、応答確認メッセージの受信のために OSS で必要な処理時間を含みます。
- サーバーレイテンシは、正常なリクエストを OSS が処理するときのレイテンシであり、E2E レイテンシに含まれるネットワーク遅延を除きます。

パフォーマンス指標は正常なリクエスト (リターンステータスコードが 2xx) のモニターに使用されることに注意してください。

次の表は具体的なメトリック指標項目の一覧です。

指標	単位	説明
GetObject リクエストの平均 E2E レイテンシ	ミリ秒	リクエスト API が GetObject である正常なリクエストの平均 E2E レイテンシ
GetObject リクエストの平均	ミリ秒	リクエスト API が GetObject

サーバーレイテンシ		である正常なリクエストの平均サーバーレイテンシ
GetObject リクエストの最大 E2E レイテンシ	ミリ秒	リクエスト API が GetObject である正常なリクエストの最大 E2E レイテンシ
GetObject リクエストの最大サーバーレイテンシ	ミリ秒	リクエスト API が GetObject である正常なリクエストの最大サーバーレイテンシ
HeadObject リクエストの平均 E2E レイテンシ	ミリ秒	リクエスト API が HeadObject である正常なリクエストの平均 E2E レイテンシ
HeadObject リクエストの平均サーバーレイテンシ	ミリ秒	リクエスト API が HeadObject である正常なリクエストの平均サーバーレイテンシ
HeadObject リクエストの最大 E2E レイテンシ	ミリ秒	リクエスト API が HeadObject である正常なリクエストの最大 E2E レイテンシ
HeadObject リクエストの最大サーバーレイテンシ	ミリ秒	リクエスト API が HeadObject である正常なリクエストの最大サーバーレイテンシ
PutObject リクエストの平均 E2E レイテンシ	ミリ秒	リクエスト API が PutObject である正常なリクエストの平均 E2E レイテンシ
PutObject リクエストの平均サーバーレイテンシ	ミリ秒	リクエスト API が PutObject である正常なリクエストの平均サーバーレイテンシ
PutObject リクエストの最大 E2E レイテンシ	ミリ秒	リクエスト API が PutObject である正常なリクエストの最大 E2E レイテンシ
PutObject リクエストの最大サーバーレイテンシ	ミリ秒	リクエスト API が PutObject である正常なリクエストの最大サーバーレイテンシ
PostObject リクエストの平均 E2E レイテンシ	ミリ秒	リクエスト API が PostObject である正常なリクエストの平均 E2E レイテンシ
PostObject リクエストの平均サーバーレイテンシ	ミリ秒	リクエスト API が PostObject である正常なリクエストの平均サーバーレイテンシ
PostObject リクエストの最大 E2E レイテンシ	ミリ秒	リクエスト API が PostObject である正常なリクエストの最大 E2E レイテンシ
PostObject リクエストの最大サーバーレイテンシ	ミリ秒	リクエスト API が PostObject である正常なリクエストの最大サーバーレイテンシ
AppendObject リクエストの	ミリ秒	リクエスト API が

平均 E2E レイテンシ		AppendObject である正常なリクエストの平均 E2E レイテンシ
AppendObject リクエストの平均サーバーレイテンシ	ミリ秒	リクエスト API が AppendObject である正常なリクエストの平均サーバーレイテンシ
AppendObject リクエストの最大 E2E レイテンシ	ミリ秒	リクエスト API が AppendObject である正常なリクエストの最大 E2E レイテンシ
AppendObject リクエストの最大サーバーレイテンシ	ミリ秒	リクエスト API が AppendObject である正常なリクエストの最大サーバーレイテンシ
UploadPart リクエストの平均 E2E レイテンシ	ミリ秒	リクエスト API が UploadPart である正常なリクエストの平均 E2E レイテンシ
UploadPart リクエストの平均サーバーレイテンシ	ミリ秒	リクエスト API が UploadPart である正常なリクエストの平均サーバーレイテンシ
UploadPart リクエストの最大 E2E レイテンシ	ミリ秒	リクエスト API が UploadPart である正常なリクエストの最大 E2E レイテンシ
UploadPart リクエストの最大サーバーレイテンシ	ミリ秒	リクエスト API が UploadPart である正常なリクエストの最大サーバーレイテンシ
UploadPartCopy リクエストの平均 E2E レイテンシ	ミリ秒	リクエスト API が UploadPartCopy である正常なリクエストの平均 E2E レイテンシ
UploadPartCopy リクエストの平均サーバーレイテンシ	ミリ秒	リクエスト API が UploadPartCopy である正常なリクエストの平均サーバーレイテンシ
UploadPartCopy リクエストの最大 E2E レイテンシ	ミリ秒	リクエスト API が UploadPartCopy である正常なリクエストの最大 E2E レイテンシ
UploadPartCopy リクエストの最大サーバーレイテンシ	ミリ秒	リクエスト API が UploadPartCopy である正常なリクエストの最大サーバーレイテンシ

正常なリクエストの操作カテゴリ

レイテンシのモニタリングと併せて、正常なリクエストのモニタリングは、アクセスリクエストの処理のシ

ステム機能をある程度反映します。同様に、現在モニターされるのは、バケット関連操作でのデータ操作を含む API だけです。

具体的な指標項目の一覧を次に示します。

指標	単位	説明
正常な GetObject リクエスト数	回	リクエスト API が GetObject である正常なリクエストの数
正常な HeadObject リクエスト数	回	リクエスト API が HeadObject である正常なリクエストの数
正常な PutObject リクエスト数	回	リクエスト API が PutObject である正常なリクエストの数
正常な PostObject リクエスト数	回	リクエスト API が PostObject である正常なリクエストの数
正常な AppendObject リクエスト数	回	リクエスト API が AppendObject である正常なリクエストの数
正常な UploadPart リクエスト数	回	リクエスト API が UploadPart である正常なリクエストの数
正常な UploadPartCopy リクエスト数	回	リクエスト API が UploadPartCopy である正常なリクエストの数
正常な DeleteObject リクエスト数	回	リクエスト API が DeleteObject である正常なリクエストの数
正常な DeleteObjects リクエスト数	回	リクエスト API が DeleteObjects である正常なリクエストの数

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すると、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に関するユーザーのコストは少なくて済みますが、クラウドアプリケーションのモニタリング、診断、トラブルシューティングは複雑です。

OSS ストレージサービスでは広範なモニタリングとログの情報が提供され、プログラムの動作を完全に把握し、問題をすばやく発見して特定するのに役立ちます。

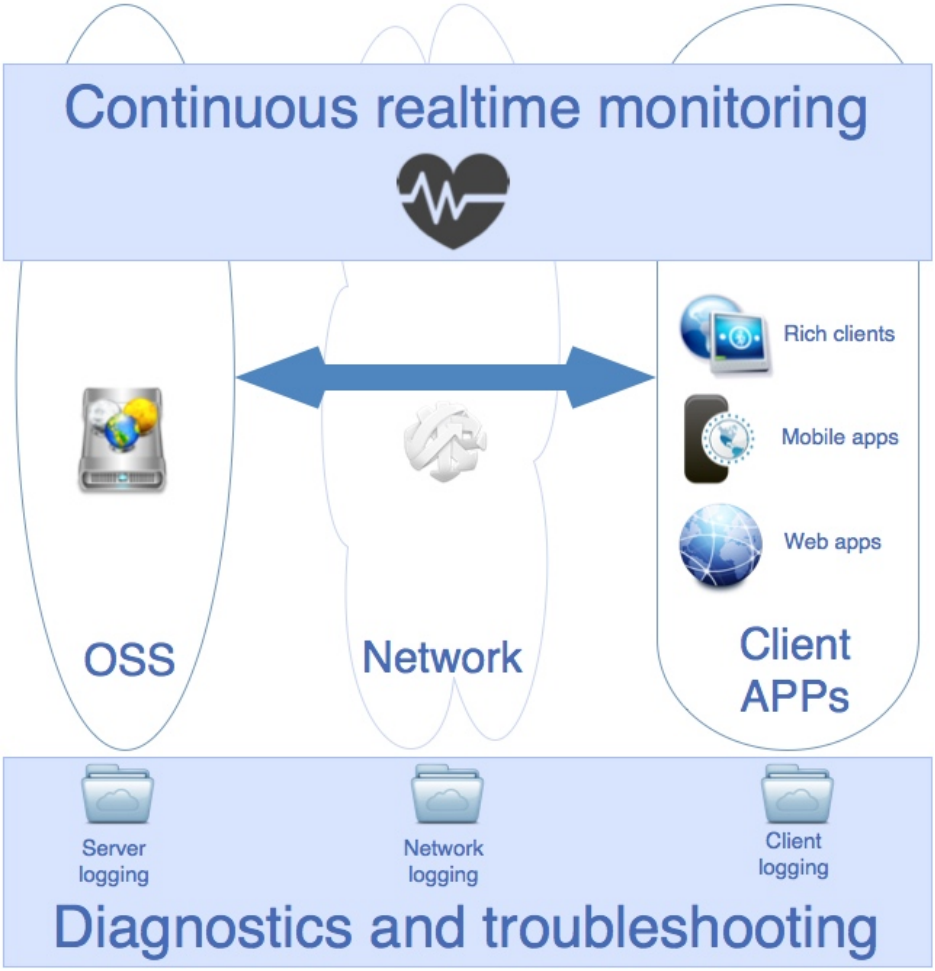
概要

ここでは、OSS のモニタリングサービス、ログ、他のサードパーティ製ツールを使用して、OSS の問題をモニター、診断、トラブルシューティングし、以下の目標を達成する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、迅速に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。
- OSS 関連の一般的な問題をすばやく解決するのに役立つ方法を提供します。

この章の内容は次のとおりです。

- OSS の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。
- 追跡と診断: 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と、追跡および診断用に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: 典型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。

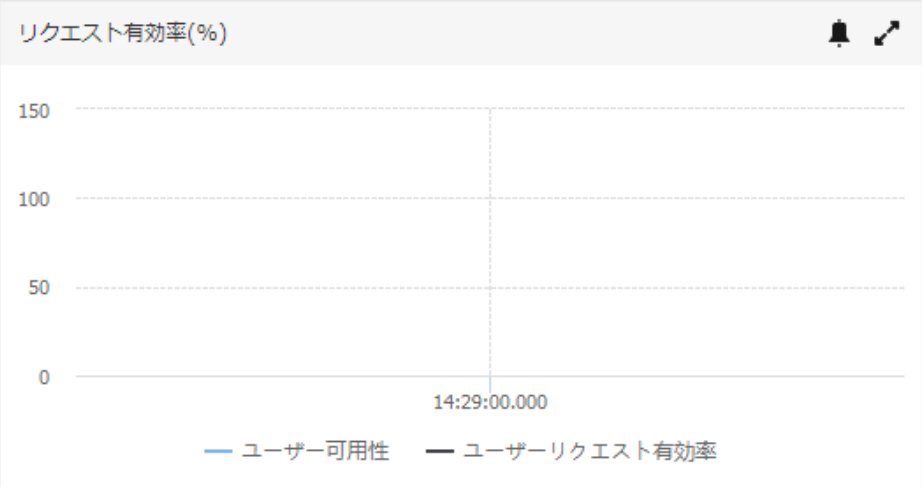


OSS モニタリング

全体的な動作条件

可用性と有効なリクエストの割合

これは、システムの安定性と、ユーザーがシステムを正常に使用できることに関する、重要な指標です。100% より低い値はすべて、一部のリクエストが失敗したことを示します。



ロードバランシングのためのパーティション移行など、システム最適化の要因のために、可用性が一時的に 100% より低下することがあります。このような場合、OSS SDK は関連する再試行メカニズムを提供し、この種の断続的障害を処理して、サービスエンドに認識させないようにできます。

また、有効なリクエストの割合が 100% より低下したときは、独自の使用方法に基づいて問題を分析する必要があります。リクエスト分布統計またはリクエストステータスの詳細を使用して、リクエストエラーの実際の種類を特定できます。その後、追跡と診断を使用して、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。ビジネスのシナリオによっては、有効なリクエストの割合が 100% より低下します。たとえば、最初にオブジェクトが存在することを確認した後、オブジェクトの存在に基づいて特定の操作を実行することが必要な場合があります。このとき、オブジェクトが存在しない場合、オブジェクトの存在を確認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、有効なリクエストの割合は 100% より下がります。

システムの高い可用性が必要なビジネスでは、指標が期待されるしきい値より低下するとトリガーされるアラームルールを設定できます。

リクエストの総数と有効なリクエストの数

この指標は、総トラフィック量の観点からシステムの動作ステータスを反映します。有効なリクエストの数がリクエストの総数と等しくない場合は、一部のリクエストが失敗したことを示します。

リクエストの総数と有効なリクエストの数の変動を監視できます (特に急激な増加または低下があった場合)。このような場合は、フォローアップアクションが必要です。アラームルールを設定して、すぐに通知を受け取ることができます。定期的なビジネスの場合は、定期的なアラームルールを設定できます (定期的なアラームは近々利用できるようになります)。詳細については、「アラームサービスユーザーガイド」を参照してください。

リクエストステータス分布の統計

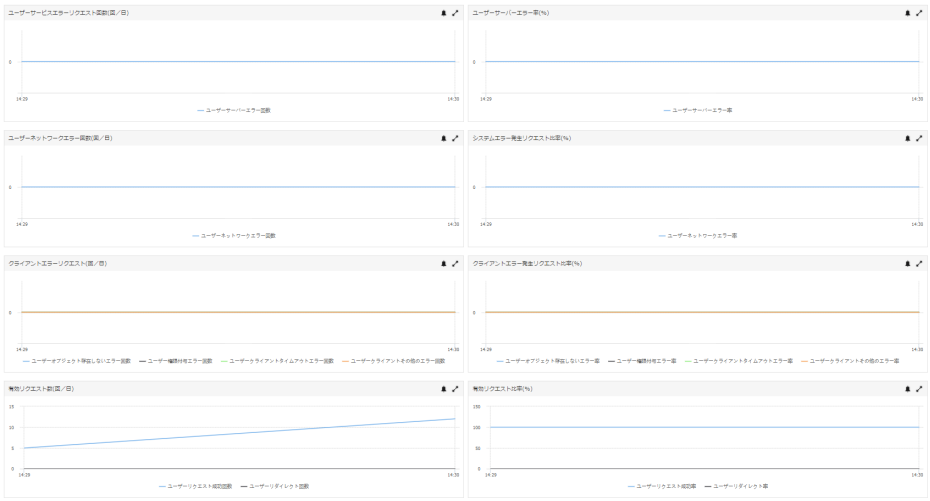
可用性または有効なリクエストの割合が 100% より低下した場合 (または、有効なリクエストの数がリクエストの総数と等しくない場合)、リクエストステータス分布の統計を調べて、リクエストエラーの種類を簡単に特定できます。このメトリック指標の詳細については、「OSS メトリックの指標のリファレンスマニュアル」を参照してください。

ユーザーレベルリクエスト

メトリック	合計値	パーセント
ユーザーリクエスト成功回数	5.00回／日	100.00%
総計	5回／日	100%

リクエストステータス詳細モニタリング

リクエストステータスの詳細では、リクエストステータス分布統計を基にしてリクエストモニタリングのステータスに関する詳細がわかります。特定の種類のリクエストを詳細にモニターできます。



パフォーマンスモニタリング

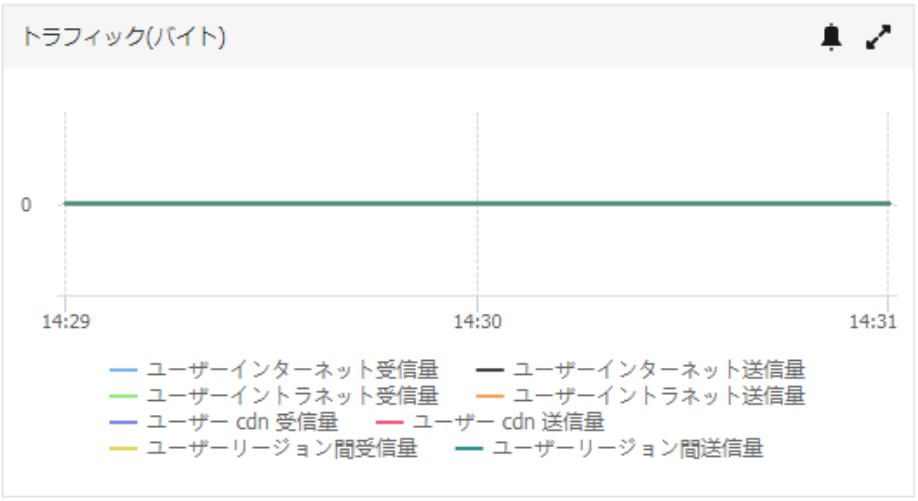
モニタリングサービスで提供される以下のメトリック項目を、パフォーマンスモニタリングの指標として使用できます。

- Average latency
  - E2E 平均レイテンシ
  - サーバー平均レイテンシ

- Maximum latency
  - E2E 最大レイテンシ
  - サーバー最大レイテンシ

- 正常なリクエストのカテゴリ

- トラフィック



前述のメトリック項目 (「トラフィック」を除きます) 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ます。

- GetObject
- HeadObject
- PutObject
- PostObject

- AppendObject
- UploadPart
- UploadPartCopy

レイテンシ指標は、リクエストを処理するために API 操作タイプで必要な平均時間または最大時間を示します。E2E レイテンシは、エンドツーエンドのレイテンシの指標です。リクエストの処理に必要な時間に加えて、リクエストを読み取って応答を送信するために必要な時間、およびネットワーク伝送による遅延を含みます。サーバーレイテンシには、サーバーでリクエストを処理するために必要な時間だけが含まれ、クライアント側の伝送ネットワーク待ち時間は含まれません。したがって、E2E レイテンシが急に増えたのに、サーバーレイテンシに大きな変化が見られない場合は、OSS システムの障害ではなく、ネットワークの不安定さがパフォーマンス低下の原因であると特定できます。

上記で説明した API に加えて、「正常なリクエストの操作カテゴリ」は次の 2 種類の API 操作に対するリクエストの数もモニターします。

- DeleteObject
- DeleteObjects

トラフィック指標は、ユーザーまたは特定のバケットの全体的状況のモニターに使用されます。インターネット、イントラネット、CDN 元検索、ドメイン間レプリケーション、他のこの種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調べます。

パフォーマンスタイプの指標では、急激かつ異常な変化に注目する必要があります。たとえば、平均レイテンシが突然急増したり、正常なリクエストレイテンシベースラインを長期間上回り続けているような場合です。パフォーマンス指標に対応するアラームルールを設定して、指標がしきい値を下回ったり上回ったりした場合に、関連する担当者にすぐ通知されるようにすることができます。定期的に山と谷があるビジネスの場合は、週単位、日単位、時間単位で比較できるように定期的なアラームルールを設定できます (定期的なアラームは近々利用できるようになります)。

課金モニタリング

これを書いた時点では、OSS モニタリングサービスがモニターできるのはストレージ容量、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、Get リクエストだけです (ドメイン間レプリケーションのアウトバウンドトラフィックおよび CDN のアウトバウンドトラフィックは含みません)。課金データに対してアラーム設定または OpenAPI 読み取り操作を行うことはできません。

OSS モニタリングサービスは、1 時間ごとにバケットレベルの課金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、連続的なモニタリングトレンドのグラフを表示できます。モニタリングビューを使用すると、ビジネスの OSS リソース使用量のトレンドを分析し、将来のコストを見積もることができます。下図のように設定します。

また、OSS モニタリングサービスでは、毎月消費されるユーザーリソースおよびバケットレベルリソースの量についての統計も提供されます。たとえば、月の 1 日から始めて、アカウントまたはバケットによって消費された OSS リソースの総量などです。これらの統計情報は 1 時間ごとに更新されます。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料金をいっそう正確にリアルタイムで把握できます。

注意: モニタリングサービスでは、提供された課金データが可能な最大限の範囲にプッシュされますが、そのために実際の課金額と食い違う可能性があります。実際の課金アプリケーションでは課金センターのデータが使用されることに注意してください。

追跡と診断

問題の診断

パフォーマンスの診断

アプリケーションのパフォーマンスの決定には、多くの主観的要因が関係します。特定のビジネスシナリオにおけるビジネスニーズの満足度をベースラインとして使用し、パフォーマンスに問題があ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き、パフォーマンスの問題の原因になる可能性がある要因が、リクエストチェーン内のどこかからもたらされることがあります。たとえば、OSS の過負荷、クライアント TCP の設定の問題、ネットワークの基本アーキテクチャでのトラフィックのボトルネックなどが、問題の原因にな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、最初に妥当なベースラインを設定する必要があります。その後、モニタリングサービスによって提供されるパフォーマンス指標を使用して、パフォーマンス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、障害のさらなる診断とトラブルシューティングに役立てます。

後の「トラブルシューティング」セクションでは、多くの一般的なパフォーマンスの問題とトラブルシューティング手段の例を示します。これは参考資料として使用できます。

エラーの診断

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あると、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、リクエストに影響する可能性のあるさまざまなタイプのエラーの統計情報を表示します。また、個別のリクエストに対する詳細な情報を、サーバーログ、クライアントログ、ネットワークログから取得できます。一般に、返される HTTP ステータスコード、OSS エラーコード、OSS エラー情報は、リクエストの障害の原因を示すことができます。

エラー応答情報の詳細については、「OSS のエラー応答」を参照してください。

ログ機能の使用

OSS では、ユーザーリクエストに対するサーバーログ機能が提供されます。これは、エンドツーエンドの詳細なリクエストログの追跡に役立ちます。

ログ機能を有効化して使用する方法については、「ログの設定」を参照してください。

Log Service の命名規則とレコード形式の詳細については、「アクセスログの設定」を参照してください。

ネットワークログツールの使用

多くの状況では、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ログのデータを記録するだけで、問題を診断できます。ただし、一部の状況では、ネットワークログツールを使用してさらに詳細な情報を取得することが必要な場合があります。

クライアントとサーバーの間で交換されるトラフィックをキャプチャすると、クライアントとサーバーの間で交換されるデータおよび基になるネットワークの状態についてさらに詳細な情報が提供され、問題の調査に役立つことがあります。たとえば、ユーザーリクエストでエラーが報告されても、サーバーログではリクエストを見ることができない場合があります。このような場合は、OSS ログ機能によって記録されるレコードを使用して、問題の原因がクライアントにあるのかどうかを調べたり、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確認したりできます。

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。この無料のプロトコルアナライザーはパケットレベルで動作し、さまざまなネットワークプロトコルの詳細なパケット情報を提供します。これにより、パケットロスおよび接続の問題をトラブルシューティングできます。

Wireshark の操作の詳細については、「Wireshark ユーザーガイド」を参照してください。

E2E の追跡と診断

リクエストは、クライアントアプリケーションのプロセスによって開始され、ネットワーク環境を通過して OSS サーバーに送られて、そこで処理されます。その後、サーバーによってネットワーク環境経由で送信された応答が、クライアントによって受信されます。これがエンドツーエンドの追跡プロセスです。クライアントアプリケーションログ、ネットワーク追跡ログ、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 が、さまざまなログからの情報を関連付けるために使用される識別子の役割を果たします。さらに、ログのタイムスタンプは、特定のログ時刻範囲をすばやく照会できるようにするだけでなく、この期間内のどの時点でリクエストイベントおよび他のクライアントアプリケーション、ネ

ットワーク、サービスシステムのイベントが発生したかも示します。これは、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、それを一意のサーバーリクエスト ID である RequestID に割り当てます。異なるログでは、RequestID は異なるフィールドに配置されます。

- OSS のログ機能によって記録されるサーバーログでは、RequestID は “Request ID” 列に配置されます。
- ネットワーク追跡のプロセスでは (たとえば、Wireshark を使用してデータストリームをキャプチャするとき)、RequestID は応答メッセージの x-oss-request-id ヘッダーの値に設定されます。
- クライアントアプリケーションでは、ユーザーがクライアントコードを使用して手動でクライアントログに RequestID を書き込む必要があります。これを書いている時点で、最新の Java SDK バージョンでは既に正常なリクエストの RequestID 情報が出力されています。getRequestId 操作を使用して、異なる API によって返される結果から RequestID を取得できます。OSS SDK のすべてのバージョンで、異常なリクエストに対する RequestID を出力できます。OSSException の getRequestId メソッドを呼び出して、この情報を取得できます。

タイムスタンプ

タイムスタンプを使用して、関連するログエントリを見つけることができます。クライアントの時刻とサーバーの時刻の間には若干のずれが存在する可能性があることに注意する必要があります。クライアントでは、ログ機能によって記録されたサーバーログのエントリを、タイムスタンプを使用して検索できます。この場合、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

パフォーマンスに関する一般的な問題

平均 E2E レイテンシが高く、平均サーバーレイテンシが低い

平均 E2E レイテンシと平均サーバーレイテンシの間の違いについては既に説明しました。それを基にすると、E2E レイテンシが高くてサーバーレイテンシが低い場合は、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
- ネットワークの要因

クライアントのパフォーマンスの問題の調査

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

使用できる接続またはスレッドの数の制限

- 使用可能な接続の数の問題は、次の方法を使用して解決できます。

a. 関連するコマンドを使用して、システムに存在する TIME_WAIT ステータスの接続の数が十分かどうかを確認します。
b. 十分存在する場合は、コアパラメーターを調整してこの問題を解決します。

- 使用可能なスレッドの数が限られている場合は、最初に、クライアント CPU、メモリ、ネットワーク、または他のリソースに影響するボトルネックを確認します。ボトルネックがない場合は、同時スレッドの数を適切に増やします。
- 問題が解決しない場合は、クライアントの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できます。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できます。

CPU、メモリ、帯域幅などのリソースの不足

- この種の問題の場合は、最初に、関連するシステムモニタリング機能を使用して、クライアントリソースのボトルネックを見つける必要があります。その後、クライアントのコードを最適化してリソースの使用を合理化するか、またはクライアントのリソースを増やします (コアまたはメモリの数)。

ネットワーク待ち時間の問題の調査

一般に、ネットワークの要因による高い E2E レイテンシは一時的なものです。一時的および持続的なネットワークの問題 (パケットロスの問題など) を調査するには、Wireshark を使用できます。

平均 E2E レイテンシと平均サーバーレイテンシは低いが、クライアントリクエストレイテンシが高い

クライアントでリクエストレイテンシが高い場合、最も可能性のある原因は、リクエストがサーバーに届いていないことです。したがって、クライアントのリクエストがサーバーに届かない理由を見つける必要があります。

クライアント側の 2 つの要因により、クライアントリクエストの送信レイテンシが高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の制限: 前のセクションで説明した解決策を参照してください。

クライアントリクエストが何回も再試行される: この場合は、再試行の情報を基にして、リクエストが再試行される原因を見つけて解決する必要があります。クライアントに再試行の問題があるかどうかは、次の方法を使用して特定できます。

クライアントのログを調べます。再試行が発生したかどうかは、詳細なログエントリを見るとわかります。OSS Java SDK を例として使用すると、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します。このようなエントリがログで見つかった場合、これは要求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探します (やはり、OSS Java SDK を例として使用しています)。このようなエントリが存在する場合は、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、ネットワークの問題の可能性を調べる必要があります (パケットロスなど)。Wireshark などのツールを使用して、ネットワークの問題を調査できます。

平均サーバーレイテンシが高い

ダウンロードまたはアップロードの間のサーバーレイテンシが高い場合は、原因として次の 2 つの要因の可能性があります。

多数のクライアントが、同じ小さいオブジェクトに頻繁にアクセスしています。

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見て、1 つまたは複数の小さいオブジェクトが短い期間に頻繁にアクセスされているかどうかを特定できます。

ダウンロードシナリオでは、そのバケットに対して CDN サービスを有効化してパフォーマンスを向上させることをお勧めします。これにより、トラフィックの料金を抑えることもできます。アップロードの場合は、ビジネスに影響がないという前提で、そのオブジェクト (バケット) に対する書き込み権限の取り消しを考えてみることができます。

内部システムの要因

内部システムの問題または最適化では解決できない問題の場合は、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 を弊社のシステムスタッフに知らせてください。スタッフが問題の解決をお手伝いします。

サーバーエラー

サーバー側のエラーが増えている場合は、次の 2 つのシナリオを検討します。

- 一時的な増加

この種の問題の場合は、クライアントプログラムで再試行ポリシーを調整し、指数バックオフなどの適切な譲歩メカニズムを採用する必要があります。これは、システムの最適化、アップグレード、他の同様の操作 (システムの負荷分散のためのパーティションの移行など) によって一時的にサービスが使用できなくなるのを防ぐだけでなく、ビジネスピークの間の高い負荷も防ぎます。

- 永続的な増加

サーバー側エラーの数が持続的に増加する場合は、、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 を弊社のバックエンドスタッフに知らせてください。スタッフが問題の発見をお手伝いします。

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 (サーバー側の問題ではなく) ネットワークエラーが発生して接続が失われると、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、システムはそのリクエストに対して HTTP ステータスコード 499 を記録します。

次のような状況では、サーバーはリクエストのステータスコードを 499 に変更することがあります。

- 受け取った読み取り/書き込みリクエストを処理する前に、接続を使用できないことをサーバーが検出した場合、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、クライアントが早まって接続を閉じると、リクエストは 499 として記録されます。

まとめると、リクエストの処理中に、クライアントが単独でリクエストを閉じると、またはクライアントがネットワークから切断されると、ネットワークエラーが発生します。クライアントが単独でリクエストを閉じる場合は、クライアントコードを調べて、クライアントが OSS から切断する原因とタイミングを明らかに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う場合は、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できます。

クライアントエラー

クライアントの権限付与エラーの増加

クライアントの権限付与エラーが増加している場合、またはクライアントが多数の 403 リクエストエラーを受け取っている場合、最も一般的な原因は以下の問題です。

- ユーザーがアクセスしているバケットのドメイン名が正しくありません。
    i. ユーザーがサードレベルまたはセカンドレベルのドメイン名を使用してバケットにアクセスしていて、バケットがドメイン名によって示されるリージョンに存在しない場合、403 エラーの原因になる可能性があります。たとえば、バケットは杭州リージョンに作成されているにもかかわらず、ユーザーが Bucket.oss-cn-shanghai.aliyuncs.com というドメイン名を使用してアクセスを試みているような場合です。この場合は、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。
    ii. CDN 高速化サービスを有効化されている場合、CDN が正しくない 元検索 ドメイン名をバインドすると、この問題が発生する可能性があります。この場合は、CDN の 元検索 ドメイン名がバケットのサードレベルドメイン名であることを確認します。
- JavaScript クライアントを使用すると 403 エラーが発生する場合は、Web ブラウザーが「同じソースポリシー」セキュリティ制約を実装しているために、CORS (Cross-Origin Resource Sharing) の設定での問題が原因である可能性があります。この場合は、バケットの CORS の設定を調べて、エラーを修正する必要があります。CORS の設定については、「CORS」を参照してください。
- アクセス制御の問題は次の 4 つの種類に分類できます。
    - プライマリ AK をアクセスに使用している場合は、AK の設定を調べて AK が無効である

場合のエラーかどうかを確認する必要があります。
  - RAM サブアカウントをアクセスに使用している場合は、サブアカウントが正しいサブアカウント AK を使用していて、サブアカウントに関連する権限があることを、確認する必要があります。
  - 一時的な STS トークンをアクセスに使用している場合は、一時的なトークンが期限切れでないことを確認する必要があります。トークンが期限切れの場合は、新しいトークンを適用します。
  - バケットまたはオブジェクトの設定をアクセス制御に使用している場合は、アクセス対象のバケットまたはオブジェクトが関連する操作に対応していることを確認する必要があります。
- サードパーティのダウンロード (署名付き URL を使用した OSS リソースへのアクセス) の権限を付与しているとき、以前は正常であったアクセスが急に 403 エラーを報告するようになった場合は、URL が期限切れの可能性があります。
- RAM サブアカウントが OSS ユーティリティを使用しているときは、これも 403 エラーの原因になることがあります。このようなユーティリティとしては、ossftp、ossbrowser、OSS コンソールクライアントなどがあります。ログオンの間に関連する AK 情報を入力し、システムがエラーをスローするとき、正しい AK を入力した場合は、AK がサブアカウント AK であること、およびそのサブアカウントに GetService や他の操作に対する権限があることを、確認する必要があります。

クライアント側での「リソースが存在しない」エラーの増加

クライアントが 404 エラーを受け取る場合は、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「リソースが存在しない」エラーの増加を検出したときは、通常、次のいずれかの問題が原因である可能性があります。

サービスの使用: たとえば、別の操作を実行する前にまずオブジェクトの存在を確認する必要があり、(たとえば Java SDK を使用して) doesObjectExist メソッドを呼び出したとき、オブジェクトが存在しない場合は、クライアントは値 “false” を受け取ります。しかし、サーバーは実際には 404 リクエストエラーを生成します。したがって、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別のプロセスによって、そのオブジェクトが以前に削除されています。この問題は、ログ機能によって記録されたサーバーログで関連するオブジェクト操作を検索することにより確認できます。

ネットワークの障害によってパケットロスと再試行が発生しています。たとえば、クライアントが削除操作を開始して特定のオブジェクトを削除します。リクエストはサーバーに到達して、削除操作が正常に実行されます。しかし、ネットワークでの伝送の間に応答パケットが失われた場合、クライアントは再試行を開始します。この 2 番目のリクエストでは、404 エラーが生成されます。ネットワークの問題によって 404 エラーが発生していることは、クライアントログとサーバーログを使用して確認できます。

- クライアントアプリケーションログで再試行リクエストを確認します。

- サーバーログにそのオブジェクトに対する 2 つの削除操作が記録されているかどうか、および 1 番目の削除操作の HTTP ステータスが 2xx であるかを確認します。

有効リクエスト率が低く、他のクライアント側リクエストエラーの数が多い

有効リクエスト率は、全リクエストのうち、HTTP ステータスコード 2xx/3xx を返したリクエストの割合です。ステータスコード 4XX または 5XX 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リクエストエラーは、サーバーエラー (5xx)、ネットワークエラー (499)、クライアント権限付与エラー (403)、リソース非存在エラー (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) を除く、その他のリクエストエラーを示します。

ログ機能によって記録されたサーバーログを調べて、これらのリクエストで発生した具体的なエラーを特定します。OSS によって返される一般的なエラーコードの一覧については、「OSS のエラー応答」を参照してください。その後、クライアントのコードを確認し、エラーの具体的な原因を探して解決します。

ストレージ容量の異常な増加

対応するアップロードリクエストの増加がないのに、ストレージ容量が異常に増加している場合は、通常、削除の問題が原因です。このような場合は、次の 2 つの要因を確認します。

クライアントアプリケーションが特定のプロセスを使用して定期的にストレージオブジェクトを削除してスペースを空けている場合:

i. 有効リクエスト率が低下しているかどうかを確認します。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意図したように削除されない可能性があります。
ii. リクエストのエラーの種類を調べて、有効リクエスト率低下の具体的な原因を見つけます。その後、特定のクライアントログを組み合わせて、詳細なエラー情報を確認できます (たとえば、ストレージ容量の解放に使用されている STS 一時トークンが期限切れの可能性があります)。

クライアントが LifeCycle を設定してストレージオブジェクトを削除している場合: コンソールまたは API を使用して、バケットの現在の LifeCycle の値が以前と同じであることを確認します。異なる場合は、単に設定を変更し、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探します。LifeCycle が正常ではあっても無効になっている場合は、OSS システム管理者に連絡して問題を明らかにします。

その他の OSS の問題

これまでに説明したトラブルシューティングが実際の問題に当てはまらない場合は、次のいずれかの方法を使用して、問題の診断とトラブルシューティングを行います。

OSS モニタリングサービスを調べて、予想されるベースライン動作と比較して変更が行われたかどうかを確認します。モニタリングビューを使用すると、その問題が一時的か永続的か、および影響を受けるストレージ操作を特定できます。

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、問題が始まったときに発生した可能性のあるエラーについての情報を探すのに役立ちます。この情報は、問題の発見と解決に役立つ可能性があります。

サーバーログの情報が十分ではない場合は、クライアントログを使用してクライアントアプリケーションを調査するか、Wireshark などのネットワークツールを使用してネットワークの問題を確認します。