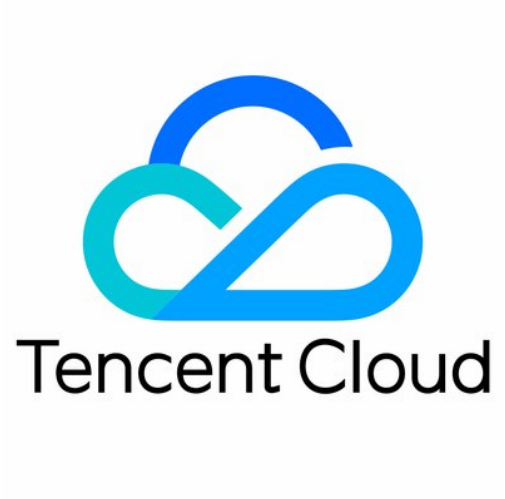
Cloud Object Storage

プラクティスチュートリアル

製品ドキュメント



カタログ：

プラクティスチュートリアル

概要

最終更新日：：2024-06-26 10:42:27

Cloud Object Storage（COS）は多方面の一般的なユースケースおよびその実践的な操作の詳細な説明を提供しており、それにはアクセス制御と権限管理、パフォーマンス最適化、データマイグレーション、データ直接転送およびバックアップ、データセキュリティとドメイン名管理、ビッグデータ、サーバーレスアーキテクチャなどの実践シナリオが含まれます。これらはCOSをより速く、より便利に使用して多様な業務ニーズを実現する上で役立ちます。具体的な実践については下表をご参照ください。

ベストプラクティス	説明
アクセス制御と権限管理	アクセス制御と権限管理はTencent Cloud COSの最も実用的な機能の一つです。実践についてのヘルプは下記の一覧をご覧ください。 ACLアクセス制御 Cloud Access Management（CAM） サブアカウントのCOSアクセス権限の承認 権限設定 COS API権限の承認 一時キー使用のセキュリティガイド 一時キーの生成および使用 サブアカウントによるタグリスト取得権限の承認 条件キーの説明および使用 他のルートアカウント下のサブアカウントへのバケット操作権限の承認
パフォーマンス最適化	Tencent Cloud COSはパフォーマンスの拡張による、より速いリクエスト速度の獲得をサポートしています。より速いリクエスト速度を獲得するための手順については、リクエスト速度とパフォーマンスの最適化をご参照ください。 Tencent Cloud COSサービスをベースに、モバイル端末が脆弱なネットワーク状態にある場合に、アップロードするパートのサイズを実際の状況に応じて動的に調整し、アップロードの成功率を向上させる方法です。詳細については、脆弱なネットワークにおけるマルチパートアップロード再開の実践をご参照ください。
AWS S3 SDKを使用したCOSアクセス	COSはAWS S3互換性を有するAPIを提供しています。この実践では、シンプルな構成変更によって、S3 SDKのインターフェースを使用してCOS上のファイルにアクセスできるようにする方法についてご説明しています。
データ障害復旧	この実践では次の3種類のバックアップシナリオについて記載するとともに、これら3種類のマイグレーションモードについてそれぞれ適用可能なバックアップ方法を示しています。 クロスリージョンレプリケーションに基づくフェイルオーバー高可用性アーキテクチャ

	クラウド上でのデータバックアップ ローカルでのデータバックアップ
ドメイン名管理の実践	HTTPSカスタムドメイン名を設定してTencent Cloud COSにアクセスする方法についてご説明します。詳細についてはカスタムドメイン名の設定によるHTTPSアクセスのサポートをご参照ください。 Tencent Cloud COSでクロスドメインアクセスルールを設定する方法についてご説明します。詳細についてはクロスドメインアクセスの設定をご参照ください。 Tencent Cloud COSで静的ウェブサイトをホストする方法についてご説明します。詳細については静的ウェブサイトのホストをご参照ください。 Tencent Cloud COSでフロントエンドシングルページアプリケーションを構築する方法についてご説明します。詳細についてはCOS静的ウェブサイト機能を使用したフロントエンドシングルページアプリケーションの構築をご参照ください。
データ直接転送の実践	この実践では次のデータ直接転送のユースケースについてご説明します。 Web端末直接転送の実践 ミニプログラム直接転送の実践 モバイルアプリケーション直接転送の実践 uni-app直接転送の実践
データセキュリティ	事前防御、事中モニタリング、事後追跡の3方面からユーザーにデータセキュリティソリューションについてご説明します。詳細についてはデータセキュリティの概要をご参照ください。 Tencent Cloud COSにおいてリンク不正アクセス防止を設定し、アクセス元に対する監視を実現する方法についてご説明します。詳細についてはリンク不正アクセス防止の実践をご参照ください。
データチェック	Tencent Cloud COSにおいてMD5検証の方式でアップロードデータの完全性を保証する方法についてご説明します。詳細についてはMD5検証をご参照ください。 CRC64チェックコードによってデータチェックを行う方法についてご説明します。詳細についてはCRC64検証をご参照ください。
ビッグデータの実践	Tencent Cloud COSをDruidのDeep storageとする操作の手順についてご説明します。詳細についてはCOSをDruidのDeep storageとして使用するをご参照ください。 Terraformを使用してTencent Cloud COSを管理する方法についてご説明します。 DataXを使用してTencent Cloud COSへのインポートまたはTencent Cloud COSからのエクスポートを行う方法についてご説明します。詳細についてはDataXを使用したCOSのインポートまたはエクスポートをご参照ください。 CDHを使用してCOSNを設定する方法についてご説明します。詳細についてはCDHのCOSN設定ガイドをご参照ください。 Tencent Cloud COS Ranger権限システムソリューションとは何かについてご説明します。詳細についてはCOS Ranger権限システムソリューションをご参照ください。 ストリームコンピューティングOceanusを使用してTencent Cloud COSに接続する方法についてご説明します。詳細についてはストリームコンピューティング

	Oceanusを使用したCOS接続をご参照ください。
サードパーティアプリケーションでのCOSの使用	S3との互換性を有するサードパーティアプリケーションでCOSの共通設定を使用し、データをTencent Cloud COSに保存する方法についてご説明します。詳細についてはS3との互換性を有するサードパーティアプリケーションでCOSの共通設定を使用するをご参照ください。 リモート添付ファイル設定機能によってフォーラムの添付ファイルをTencent Cloud COS上に保存する方法についてご説明します。 WordPressでサードパーティプラグインを使用して、マルチメディアコンテンツをTencent Cloud COS上に保存する方法についてご説明します。詳細についてはWordPressのマルチメディアコンテンツをCOSに保存するをご参照ください。
APIによる複数ファイルのパッケージ圧縮	ここでは主に、ユーザーがAPIによって複数ファイルをパッケージ圧縮する方法についてご説明します。

アクセス制御と権限管理
Cloud Access Managementの実践

最終更新日：：2024-06-26 10:42:27

Cloud Access Managementの概要

Cloud Access Management（CAM）はTencent Cloudがご提供するID認証および権限承認サービスです。これは主に、お客様がTencent Cloudアカウント下のリソースへのアクセス権限を安全に管理するために役立てられます。権限が与えられている場合、権限を持つオブジェクト、リソース、操作を管理できるほか、いくつかのポリシー制限を設定することもできます。

機能

ルートアカウントリソースの権限承認

ルートアカウントのリソース権限は、サブアカウントまたは他のルートアカウントなどの他者に付与することができ、その際ルートアカウントに関連するID証明書を共有する必要はありません。

精緻化された権限管理

リソースごとに、異なる人に異なるアクセス権限を付与することができます。例えば、いくつかのサブアカウントにはあるCloud Object Storage（COS）バケットの読み取り権限を与え、別のいくつかのサブアカウントまたはルートアカウントにはあるCOSストレージオブジェクトの書き込み権限を与えることなどができます。上記のリソース、アクセス権限およびユーザーはすべて一括パッケージ化が可能です。

最終的な整合性

CAMは現在Tencent Cloudの複数のリージョンでサポートしています。ポリシーデータをコピーすることで、クロスリージョナルなデータ同期を実現します。CAMポリシーの変更はタイムリーに送信されますが、異なるリージョン間でポリシーの同期を行うとポリシーの発効に遅延が生じる場合があります。またCAMはキャッシュを使用してパフォーマンスを向上させているため（現在のキャッシュ時間は1分間）、更新はキャッシュが期限切れになってから発効します。

ユースケース

企業のサブアカウント権限管理

企業内の各職場の従業員に、その企業のクラウドリソースへの最小のアクセス権限を持たせる必要がある場合です。例えば、ある企業にCVM、VPCインスタンス、CDNインスタンス、COSバケットおよびオブジェクトなどの、多くのクラウドリソースがあるとします。この企業には開発要員、テスト要員、運用保守要員などの多くの従業員がいます。
一部の開発要員には、その所属プロジェクトに関連する開発マシンクラウドリソースの読み取り書き込み権限が必要であり、テスト要員にはその所属プロジェクトのテストマシンクラウドリソースの読み取り書き込み権限が必要です。運用保守要員はマシンの購入と日常的な運用を担います。社内で従業員の職責または参加プロジェクトに変更が生じた場合は、対応する権限を終了します。

異なる企業間での権限管理

異なる企業間でのクラウドリソースの共有が必要な場合です。例えば、多くのクラウドリソースを持つある企業が製品開発に特化するため、クラウドリソースの運用保守業務の権限を他の運営企業に委託して実施させる場合です。運営企業との契約の終了時に、対応する管理権限を回収します。

ポリシー構文

ポリシー（policy）はいくつかの要素で構成され、権限の具体的な情報を記述するために用いられます。中心となる要素には、プリンシパル（principal）、アクション（action）、リソース（resource）、発効条件（condition）およびエフェクト（effect）が含まれます。その他の詳細についてお知りになりたい場合は、アクセスポリシーの言語概要をご参照ください。

説明：

ポリシー構文の記述に順序の要件はありません。アクション（action）の要素はアルファベットの大文字と小文字を区別する必要があります。
ポリシーが特定の条件に制約されることがなければ、conditionの要素はオプション項目です。
principalの要素はコンソールで書き込むことはできません。ポリシー管理API内およびポリシー構文関連パラメータ内でのみprincipalを使用することができます。

中心的要素

中心的要素	説明	入力必須かどうか
バージョン version	記述するポリシー構文のバージョンです。現在許可されている値は2.0のみです	はい
プリンシパル principal	ポリシーが権限を付与するエンティティの記述に用いられます。ユーザー（デベロッパー、サブアカウント、匿名ユーザー）、ユーザーグループが含まれます	ポリシー管理API内およびポリシー構文関連パラメータ内でのみこの要素を使用することができます
ステートメント	1つまたは複数の権限の詳細情報の記述に用いられます。この要素にはaction、resource、condition、effectなどのいくつかの他の要素の権	はい

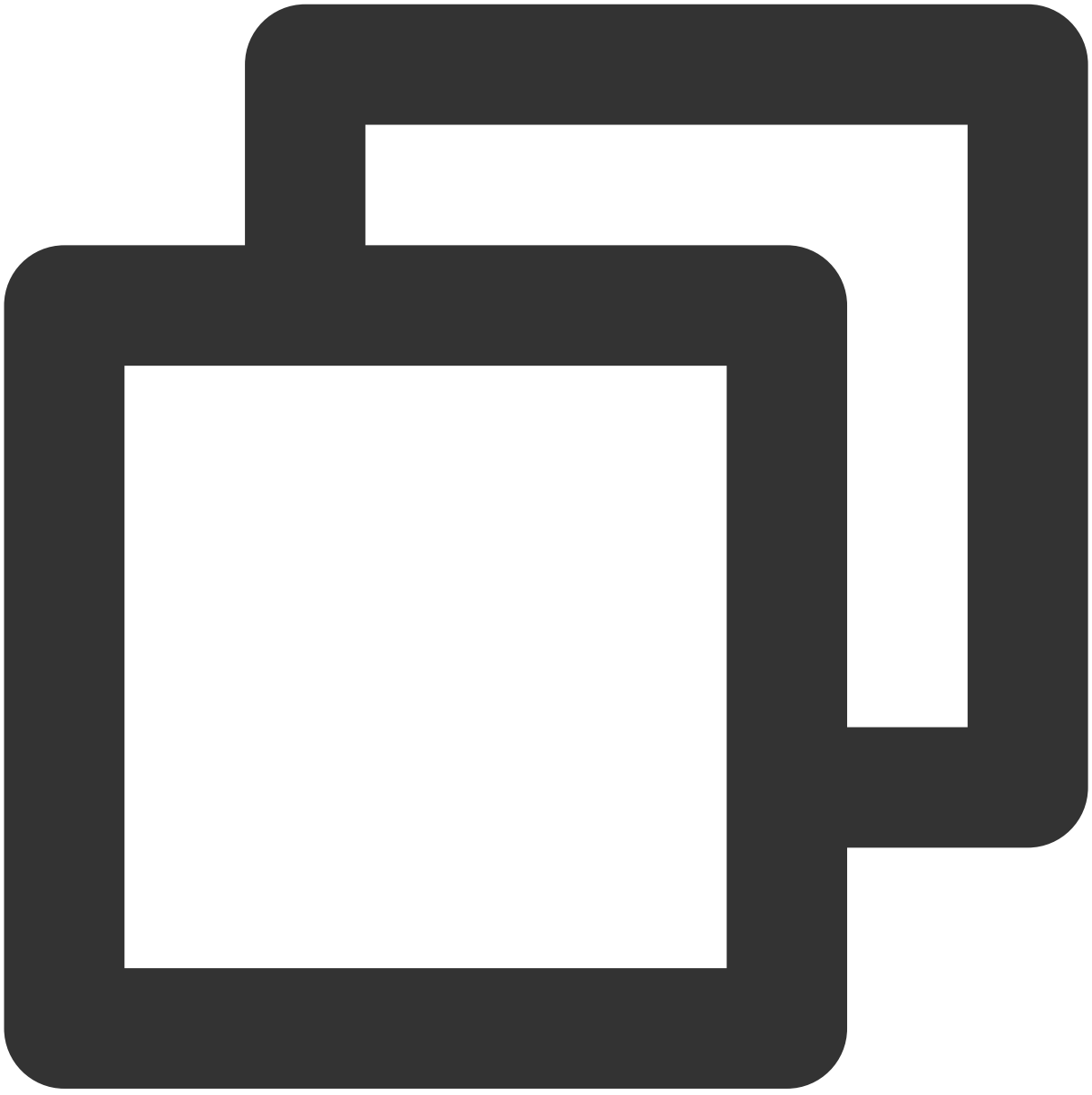
statement	限または権限セットが含まれます。1つのポリシーには1つのstatement要素しかありません	
アクション action	許可または拒否するアクションの記述に用いられます。アクションは単一のAPIのアクションまたは複数のAPIアクションのセットとすることができます。アクション（action）の要素はアルファベットの大文字と小文字を区別する必要があり、例えば `name/cos:GetService` などとします	はい
リソース resource	権限の具体的なデータの記述に用いられます。リソースは6セグメント式で記述します。リソース定義の詳細は各製品によって異なります。リソース情報の指定方法については、作成したリソースステートメントに対応する製品ドキュメントをご参照ください	はい
発効条件 condition	ポリシー発効の制約条件の記述に用いられます。条件はオペレーター、アクションキーとアクション値で構成されています。条件値には時間、IPアドレスなどの情報を含めることができます。一部のサービスは、条件に対しほかの値を指定することを認めています	いいえ
エフェクト effect	ステートメントによる結果の記述に用いられます。allow（許可）とdeny（明示的な拒否）の2種類が含まれます	はい

ポリシーの制限

制限項目	制限値
1つのルートアカウント内のユーザーグループ数	300
1つのルートアカウント内のサブアカウント数	1000
1つのルートアカウント内のロール数	1000
1つのサブアカウントに追加できるユーザーグループ数	10
1つのコラボレーターがコラボレートできるルートアカウント数	10
1つのユーザーグループ内のサブアカウント数	100
1つのルートアカウントが作成できるカスタムポリシー数	1500
1つのユーザー、ユーザーグループまたはロールに直接バインドできるポリシー数	200
1つのポリシー構文の最大文字数	4096

ポリシーの例

次のポリシーの記述例は、ルートアカウントID 100000000001（APPIDは1250000000）下のサブアカウントID 100000000011に対し、北京リージョンのバケットのexamplebucket-bjおよび広州リージョンのバケットのexamplebucket-gz下のオブジェクトexampleobjectについて、アクセスIPが `10.*.*.10/24` セグメントの場合に、**オブジェクトのアップロード**および**オブジェクトのダウンロード**権限を許可するものです。



```
{
            "version": "2.0",
            "principal":{
                "qcs": ["qcs::cam::uin/100000000001:uin/100000000011"]
            },
            "statement": [{
```

```
                "effect": "allow",
                "action": ["name/cos:PutObject", "name/cos:GetObject"],
                "resource": ["qcs::cos:ap-beijing:uid/1250000000:examplebucket-bj-1
                    "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-gz-12500000
                ],
                "condition":{
                    "ip_equal":{
                        "qcs:ip": "10.*.*.10/24"
                    }
                }
            }]
}
```

サブアカウントへのCOSアクセス権限承認

最終更新日：：2024-06-26 10:42:27

概要

Cloud Object Storage（COS）のリソースについては、異なる企業間または社内の複数のチーム間で、チームまたは要員ごとに異なるアクセス権限を持つよう設定する必要があります。Cloud Access Management（CAM）を使用することで、バケットまたはオブジェクトごとに異なる操作権限を設定し、異なるチームまたは要員同士によるコラボレーションを可能にします。
まず、重要な概念であるルートアカウント、サブアカウント（ユーザー）およびユーザーグループについての理解が必要です。CAMの関連用語、設定の詳細説明については、CAMの用語集をご参照ください。

ルートアカウント

ルートアカウントはデベロッパーとも呼ばれます。ユーザーがTencent Cloudアカウントを申請すると、システムはTencent Cloudサービスにログインするためのルートアカウント ID を作成します。ルートアカウントはTencent Cloudリソースの使用量の計算と課金における基本主体です。
ルートアカウントはデフォルトで、その名前の下にあるすべてのリソースへの完全なアクセス権限を有します。これには注文情報へのアクセス、ユーザーパスワードの変更、ユーザーおよびユーザーグループの作成、他のクラウドサービスリソースへのアクセスなどが含まれます。デフォルトでは、リソースにアクセスできるのはルートアカウントのみであり、他のユーザーからのアクセスはすべてルートアカウントによる権限承認を経る必要があります。

サブアカウント（ユーザー）とユーザーグループ

サブアカウントとはルートアカウントが作成するエンティティであり、確実なIDおよびIDクレデンシャルを有し、Tencent Cloudコンソールにログインする権限を有します。
サブアカウントはデフォルトではリソースを所有せず、所属するルートアカウントによる権限承認を受ける必要があります。
1つのルートアカウントは複数のサブアカウント（ユーザー）を作成することができます。
1つのサブアカウントは複数のルートアカウントに帰属することができ、複数のルートアカウントが各自のクラウドリソースを管理する上でそれぞれ役立てることができます。ただし、1つのサブアカウントは、同一時刻に1つのルートアカウントにしかログインできず、そこでそのルートアカウントのクラウドリソースを管理します。
サブアカウントはコンソールを通じてデベロッパー（ルートアカウント）を、あるルートアカウントから別のルートアカウントに切り替えることができます。
サブアカウントがコンソールにログインすると、デフォルトのルートアカウントに自動的に切り替わり、デフォルトのルートアカウントが承認したアクセス権限を得ることができます。

デベロッパーを切り替えると、サブアカウントは切り替えたルートアカウントによって承認されたアクセス権限を得る一方、切り替え前のルートアカウントによって承認されたアクセス権限は無効になります。
ユーザーグループは同一の職能を持つ複数のユーザー（サブアカウント）の集合です。業務上のニーズに応じてさまざまなユーザーグループを作成し、ユーザーグループに適切なポリシーをバインドして異なる権限を割り当てることができます。

操作手順

サブアカウントへのCOSアクセス権限の承認には、サブアカウントの作成、サブアカウントへの権限承認、サブアカウントによるCOSリソースへのアクセスという3つの手順があります。

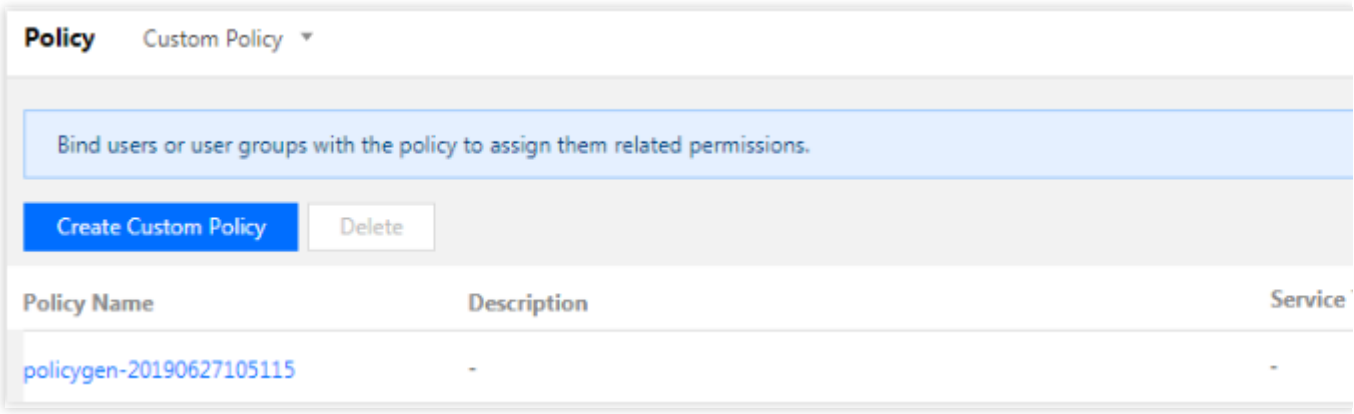
ステップ1：サブアカウントの作成

CAMコンソールでサブアカウントを作成し、サブアカウントにアクセス権限承認の設定を行うことができます。具体的な操作は次のとおりです。
1. CAMコンソールにログインします。
2. **ユーザー>ユーザーリスト> ユーザーの新規作成**を選択し、ユーザー新規作成ページに進みます。
3. **カスタム作成**を選択し、**リソースへのアクセスおよびメッセージの受信が可能**タイプを選択し、**次のステップ**をクリックします。
4. 要件に従ってユーザー関連情報を入力します。
ユーザー情報の設定：サブユーザー名を入力します（例：Sub_user）。サブユーザーのメールアドレスを入力します。サブユーザーにメールアドレスを追加し、Tencent Cloudから送信される、WeChatにバインドされたメールを受信できるようにする必要があります。
アクセス方法：プログラムアクセスとTencent Cloudコンソールアクセスから選択します。その他の設定は必要に応じて選択できます。
5. ユーザー情報の入力完了後、**次のステップ**をクリックしてID認証を行います。
6. ID認証の完了後、サブユーザー権限の設定を行います。システムから提供されるポリシーに基づいて選択することで、例えばCOSのバケットリストのアクセス権限、読み取り専用権限などの簡単なポリシーを設定することができます。より複雑なポリシーを設定したい場合は、ステップ2：サブアカウントへの権限承認を実行してください。
7. ユーザータグを設定します。この項目はオプションであり、必要に応じて設定できます。**次のステップ**をクリックします。
8. 設定情報に誤りがないことを確認後、**完了**をクリックするとサブアカウントの作成が完了します。

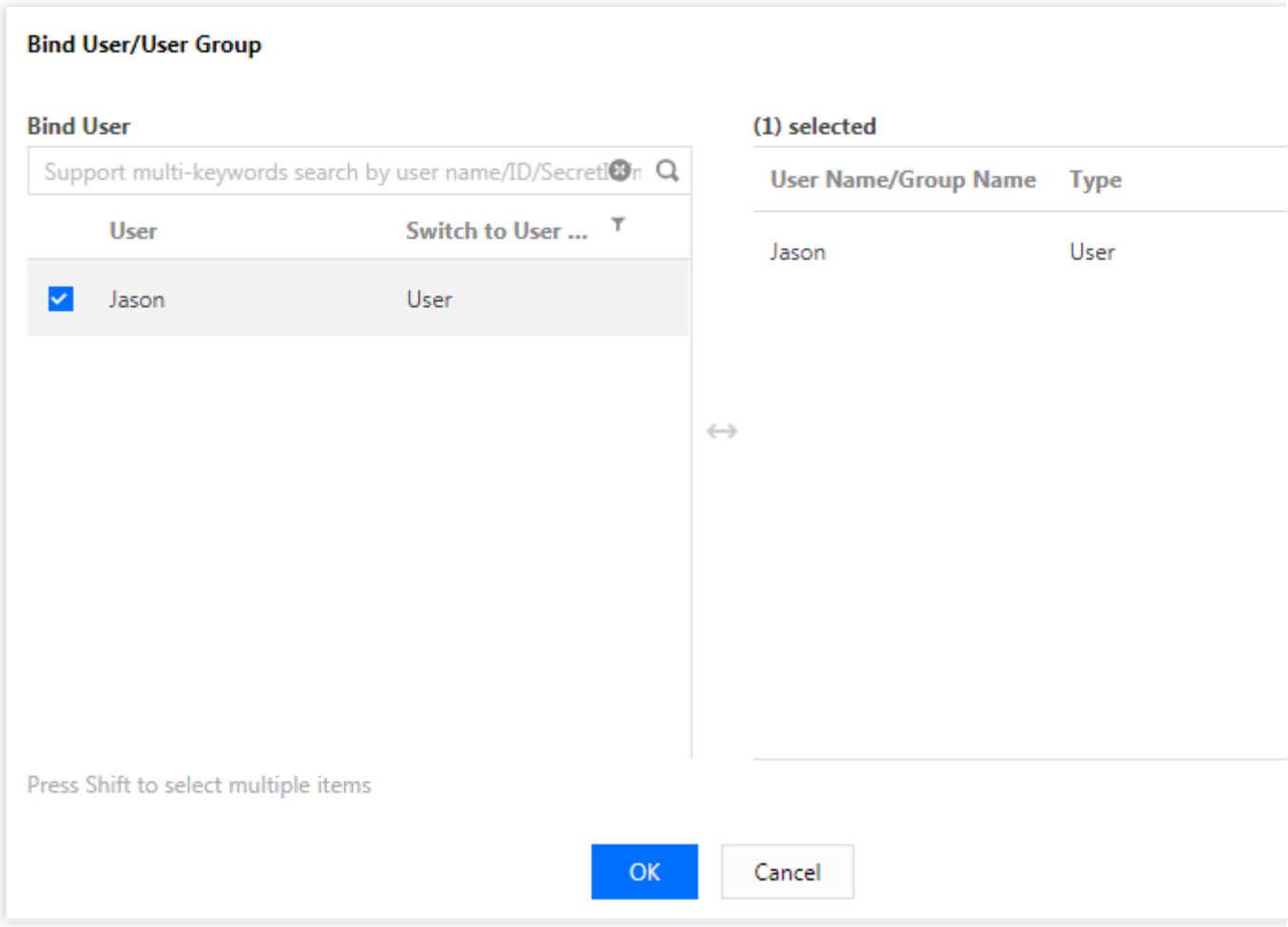
ステップ2：サブアカウントへの権限承認

カスタムポリシーを作成するか、または既存のポリシーを選択し、ポリシーをサブアカウントにバインドします。
1. CAMコンソールにログインします。
2. **ポリシー > カスタムポリシーの新規作成 > ポリシー構文で作成**を選択し、ポリシー作成ページに進みます。

3. 実際のニーズに応じて**空白テンプレート**を選択して権限承認ポリシーをカスタマイズするか、またはCOSにバインドされた**システムテンプレート**を選択し、**次のステップ**をクリックします。
4. 覚えやすいポリシー名を入力します。**空白テンプレート**を選択した場合はポリシー構文を入力する必要があります。詳細についてはポリシーの例をご参照ください。ポリシーの内容をコピーして**ポリシー内容**のエディタボックス内に貼り付け、入力に間違いがないことを確認してから**完了**をクリックします。
5. 作成完了後、作成したポリシーをサブアカウントにバインドします。



6. サブアカウントにチェックを入れ、**OK**をクリックして権限を承認すると、限定されたCOSリソースにサブアカウントを使用してアクセスできるようになります。



ステップ3：サブアカウントによるCOSリソースへのアクセス

上記のステップ1で設定した2種類のアクセス方法、プログラムアクセスとTencent Cloudコンソールアクセスについての説明は次のとおりです。

（1）プログラムアクセス

サブアカウントを使用し、プログラム（API、SDK、ツールなど）によってCOSリソースにアクセスする場合は、先にルートアカウントのAPPID、サブアカウントのSecretIdおよびSecretKey情報を取得する必要があります。サブアカウントのSecretIdとSecretKeyはCAMコンソールで生成できます。

1. ルートアカウントでCAMコンソールにログインします。
2. **ユーザーリスト**を選択し、ユーザーリストページに進みます。
3. サブアカウントのユーザー名をクリックし、サブアカウント情報詳細ページに進みます。
4. **APIキー**タブをクリックし、**キーの作成**をクリックしてこのサブアカウントのSecretIdとSecretKeyを作成します。

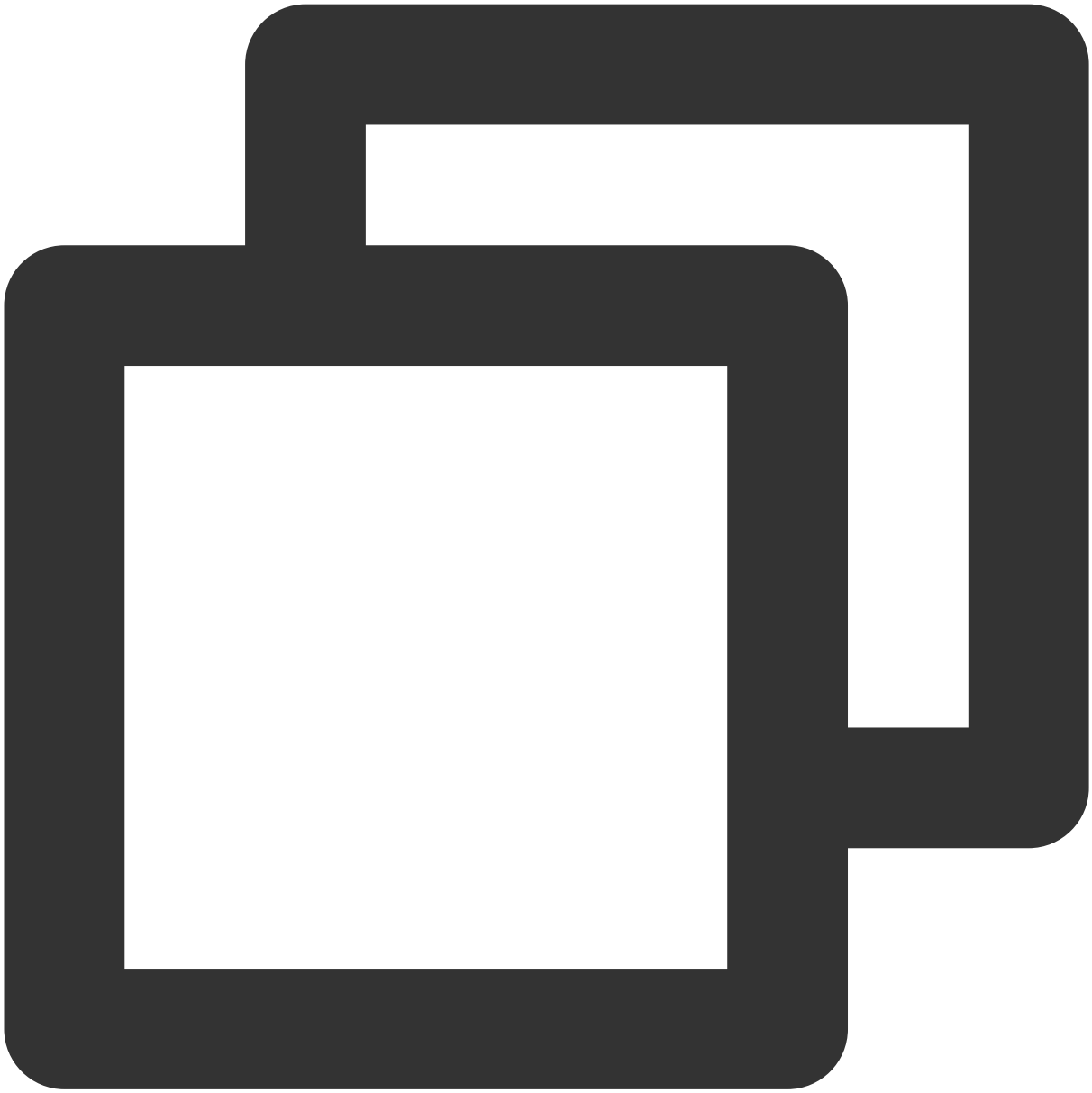
この時点で、サブアカウントのSecretIdとSecretKey、ルートアカウントのAPPIDによってCOSリソースにアクセスすることが可能になりました。

注意：

サブアカウントはXML APIまたはXML APIベースのSDKによってCOSリソースにアクセスする必要があります。

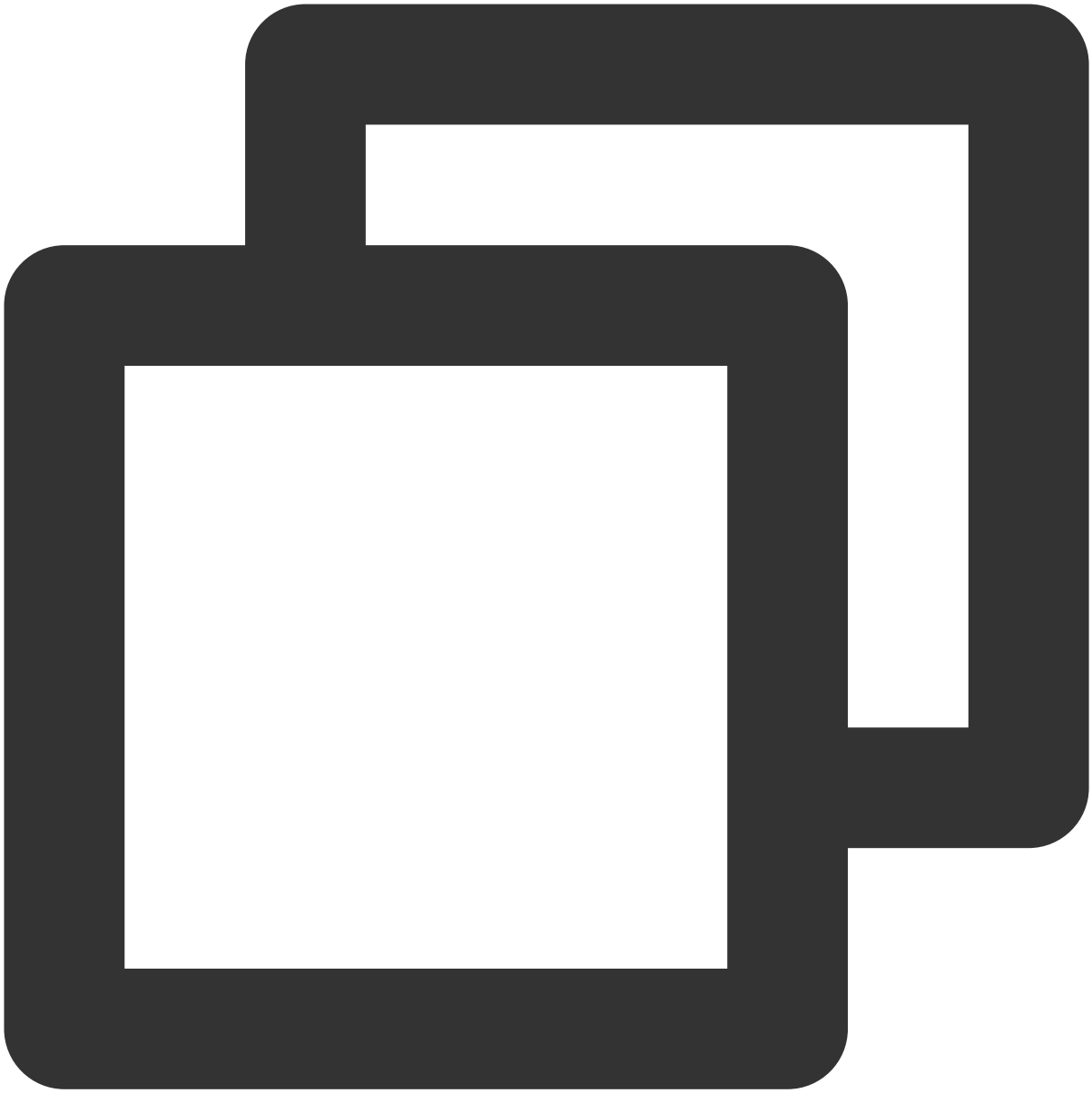
XMLベースのJava SDKによるアクセスの例

XMLベースのJava SDKコマンドラインの場合、入力する必要があるパラメータは次のとおりです。



```
// ID情報の初期化
COSCredentials cred = new BasicCOSCredentials("<ルートアカウントのAPPID>", "<サブアカウ
```

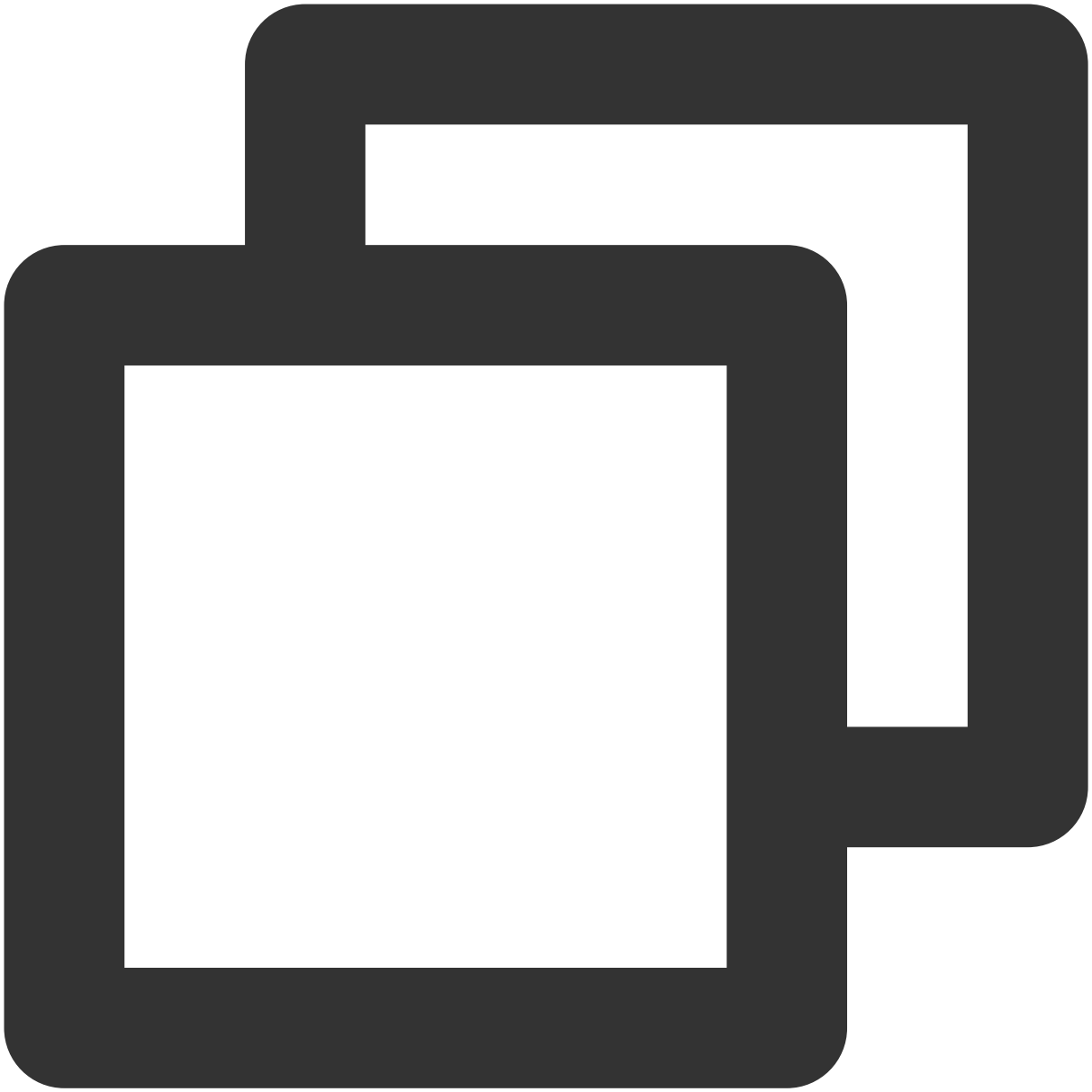
実際の例は次のとおりです。



```
String secretId = System.getenv("secretId");//サブアカウントのSecretIdです。権限承認は最小
String secretKey = System.getenv("secretKey");//サブアカウントのSecretKeyです。権限承認は
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);

// ID情報の初期化
COSCredentials cred = new BasicCOSCredentials("<ルートアカウントのAPPID>", secretId, s
```

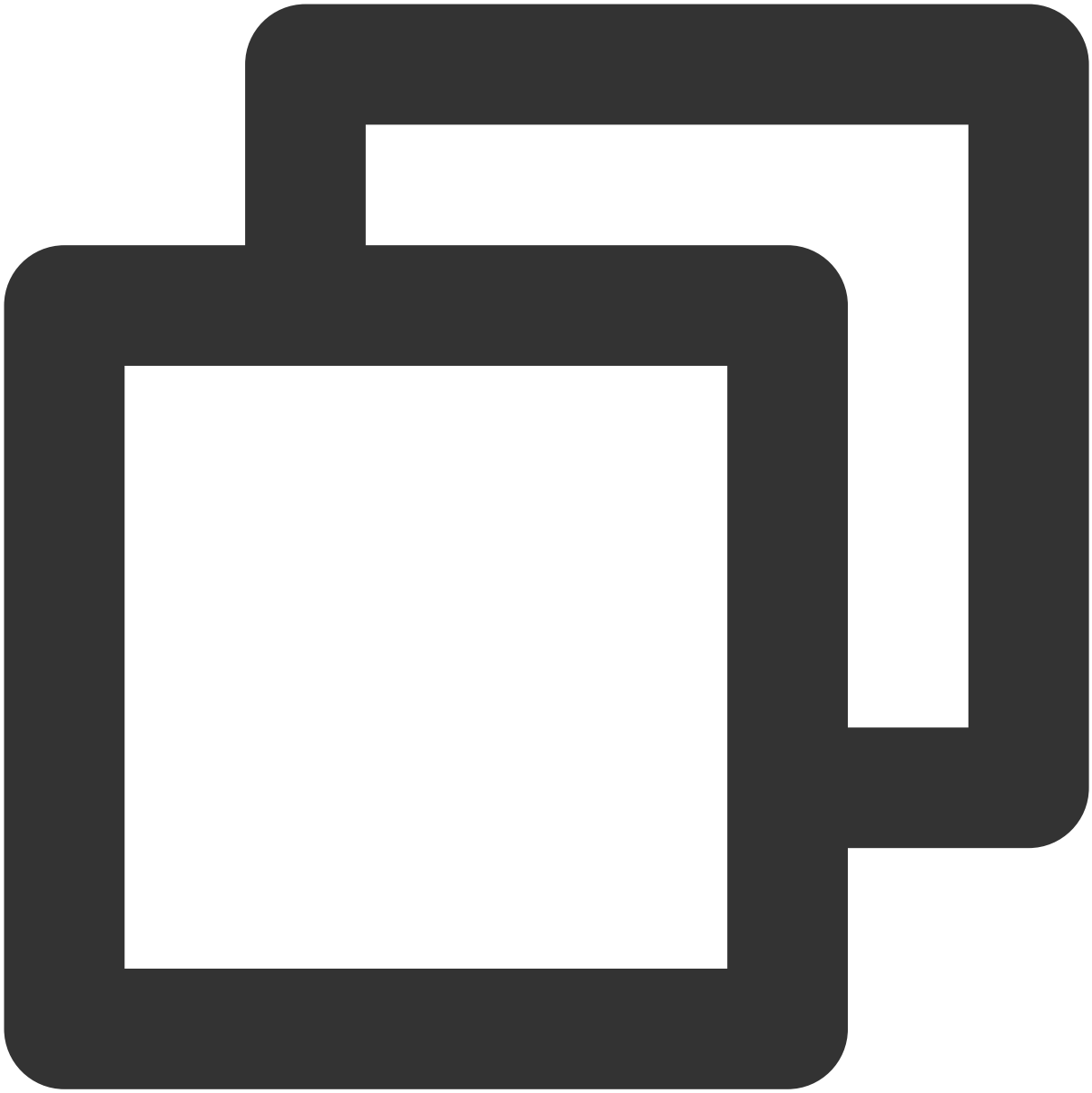
COSCMDコマンドラインツールによるアクセスの例

COSCMDの設定コマンドの場合、入力する必要があるパラメータは次のとおりです。



```
coscmd config -u <ルートアカウントのAPPID> -a <サブアカウントのSecretId> -s <サブアカウント
```

実際の例は次のとおりです。



```
coscmd config -u 1250000000 -a AKIDasdfmRxHPa9oLhJp**** -s e8Sdeasdfas2238Vi**** -b
```

（2）Tencent Cloudコンソール

サブユーザーに権限を承認した後、サブユーザーログイン画面でルートアカウントID、サブユーザー名、サブユーザーパスワードを入力してコンソールにログインし、**クラウド製品**の中の**Cloud Object Storage**を選択してクリックすると、ルートアカウント下のストレージリソースにアクセスすることができます。

ポリシーの例

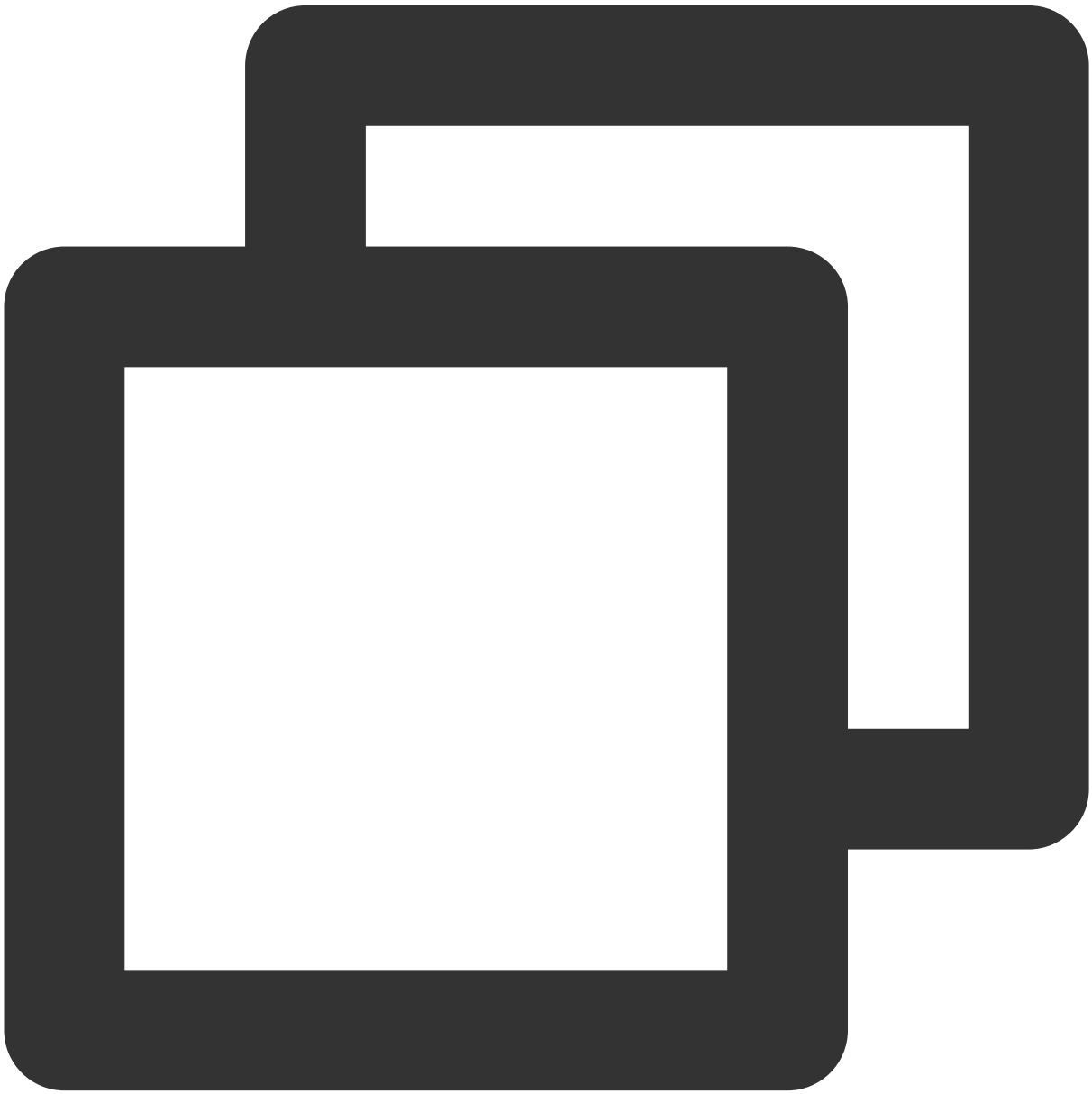
以下にいくつかの典型的なシナリオでのポリシーの例を挙げます。カスタムポリシーを設定する場合は、次の参照ポリシーをコピーして入力ボックスに貼り付け、**ポリシー内容を編集**することができます。実際の設定に応じて変更するだけで完了です。COSの一般的なシナリオにおけるその他のポリシー構文については、アクセスポリシーの言語概要またはCAM製品ドキュメントの**ビジネスユースケース**の部分をご参照ください。

事例1：サブアカウントにCOSの全読み取り書き込み権限を設定する

注意：

このポリシーは権限の範囲が大きいため、慎重に設定してください。

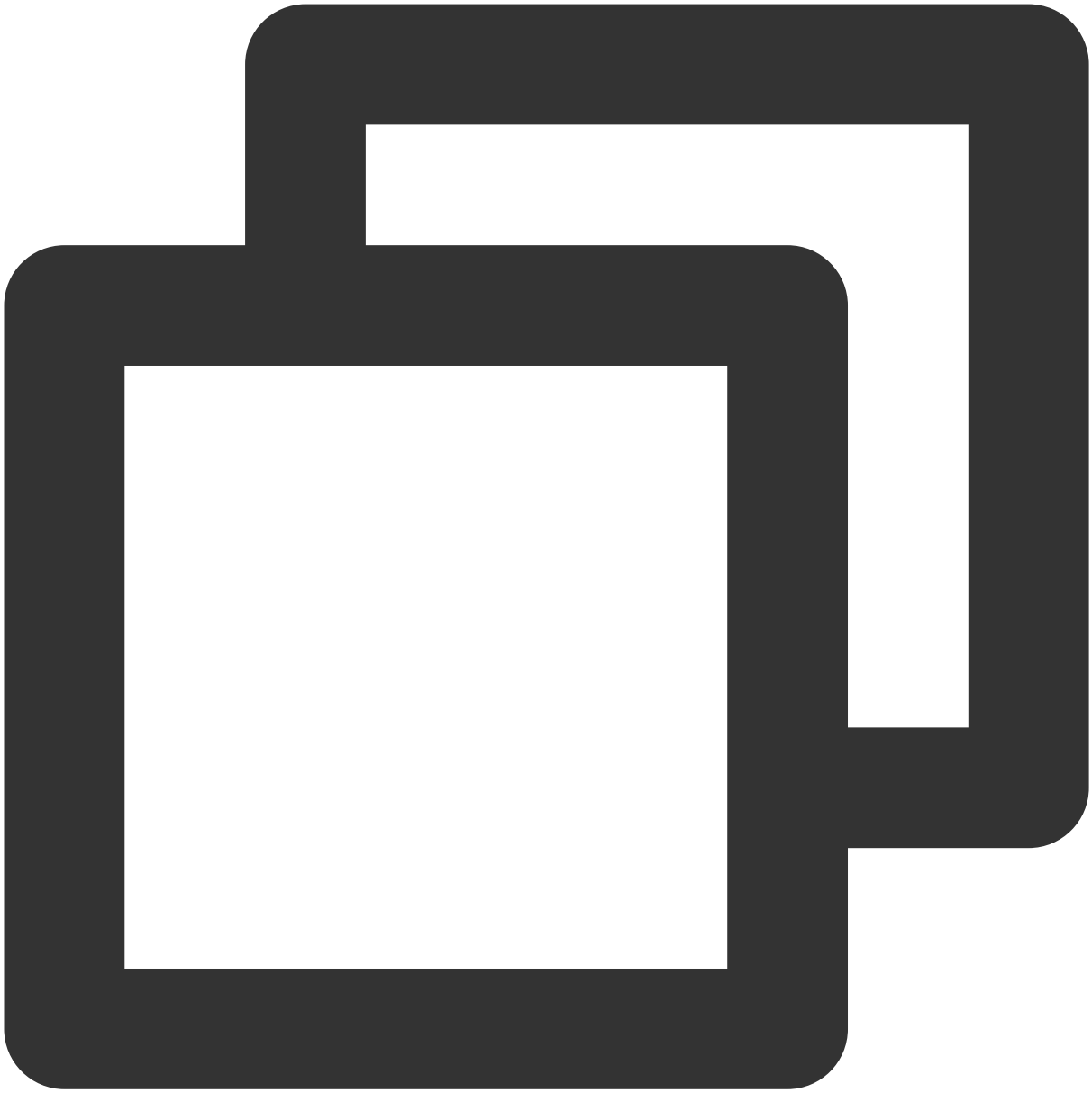
具体的なポリシーは次のとおりです。



```
{
    "version": "2.0",
    "statement":[
        {
            "action":[
                "name/cos:*"
            ],
            "resource": "*",
            "effect": "allow"
        },
        {
            "effect": "allow",
            "action": "monitor:*",
            "resource": "*"
        }
    ]
}
```

事例2：サブアカウントに読み取り専用権限を設定する

サブアカウントに読み取り専用権限のみを設定する場合の、具体的なポリシーは次のとおりです。

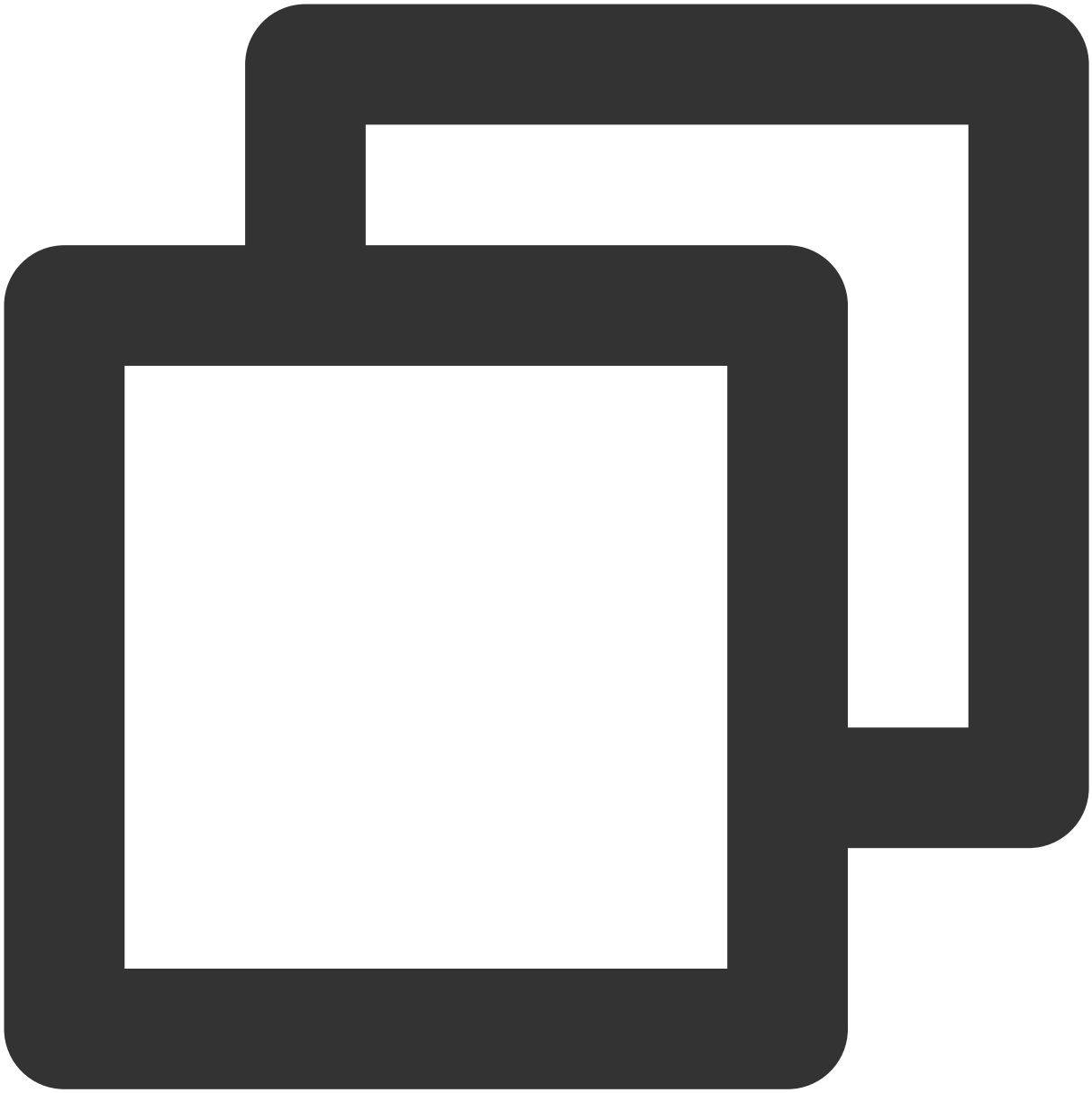


```
{
    "version": "2.0",
    "statement":[
        {
            "action":[
                "name/cos:List*",
                "name/cos:Get*",
                "name/cos:Head*",
                "name/cos:OptionsObject"
            ],
            "resource": "*",
```

```
            "effect": "allow"
        },
        {
            "effect": "allow",
            "action": "monitor:*",
            "resource": "*"
        }
    ]
}
```

事例3：サブアカウントに書き込み専用権限（削除を含めない）を設定する

具体的なポリシーは次のとおりです。

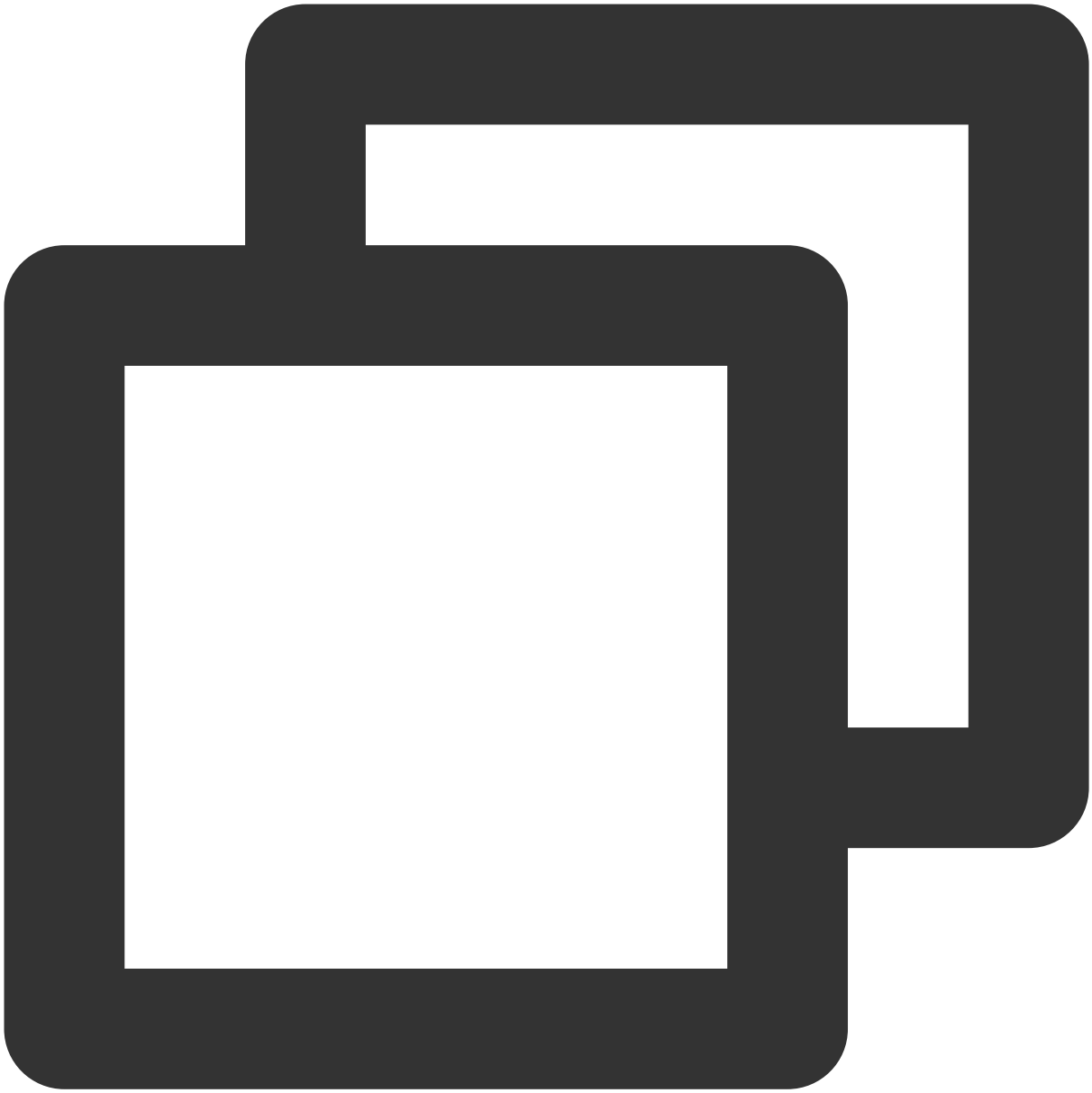


```
{
    "version": "2.0",
    "statement":[
        {
            "effect": "allow",
            "action":[
                "cos:ListParts",
                "cos:PostObject",
                "cos:PutObject*",
                "cos:InitiateMultipartUpload",
                "cos:UploadPart",
```

```
                "cos:UploadPartCopy",
                "cos:CompleteMultipartUpload",
                "cos:AbortMultipartUpload",
                "cos:ListMultipartUploads"
            ],
            "resource": "*"
        }
    ]
}
```

事例4：サブアカウントに特定のIPセグメントの読み取り書き込み権限を設定する

次に示す例では、IPネットワークセグメントが `192.168.1.0/24` および `192.168.2.0/24` のアドレスだけが読み取り書き込み権限を有するよう制限しています。
より豊富な発効条件を入力する場合は、発効条件をご参照ください。



```
{
    "version": "2.0",
    "statement":[
        {
            "action":[
    "cos:*"
            ],
            "resource": "*",
            "effect": "allow",
            "condition":{
                "ip_equal":{
```

```
                    "qcs:ip": ["192.168.1.0/24", "192.168.2.0/24"]
                }
            }
        }
    ]
}
```

一時キーの生成と使用

最終更新日：：2024-06-26 10:42:27

注意：
一時キーを使用してアクセス権限を付与する際は、必ず業務の必要性に応じて、最小権限の原則に従って権限を付与してください。すべてのリソース `(resource:*)` またはすべてのアクション `(action:*)` の権限を直接与えてしまうと、権限が大きすぎるためにデータセキュリティ上のリスクが生じます。
一時キーを申請する際、権限の範囲を指定していると、申請した一時キーも権限の範囲でしか操作を行うことができません。例えば一時キーを申請する際、バケットexamplebucket-1-1250000000にファイルをアップロードできるという権限の範囲を指定していた場合、申請したキーではファイルをexamplebucket-2-1250000000にアップロードすることも、examplebucket-1-1250000000からファイルをダウンロードすることも**できません**。

一時キー

一時キー（一時アクセス証明書）はCAMのTencent Cloud APIによって提供されるインターフェースを通じて取得する、権限が限定されたキーです。
COS APIは一時キーを使用して署名を計算し、COS APIリクエストの送信に使用することができます。
COS APIリクエストは一時キーを使用して署名を計算する際、一時キー取得インターフェースから返された情報の中の、次の3つのフィールドを使用する必要があります。
TmpSecretId
TmpSecretKey
Token

一時キーを使用するメリット

Web、iOS、AndroidでCOSを使用する際、固定のキーによる署名計算方式では権限を有効に制御できず、またパーマネントキーをクライアントコードに含めることには極めて大きな漏洩リスクが伴います。一時キー方式を使用すると、権限制御の問題を便利かつ有効に解決することができます。
例えば、一時キー申請の過程で、権限ポリシーのpolicyフィールドを設定することで、操作およびリソースを制限し、権限を指定された範囲内に限定することができます。
COS API権限承認ポリシーに関しては、次をご参照ください。
COS API一時キー権限承認ポリシーガイド
一般的なケースにおける一時キー権限ポリシーの例

一時キーの取得

一時キーは、提供されているCOS STS SDK方式で取得するか、またはSTS Cloud APIを直接リクエストする方式で取得することもできます。

注意：

例で使用しているのはJava SDKであり、GitHubでSDKコード（バージョン番号）を取得する必要があります。対応するSDKバージョン番号が見つからないと表示された場合は、GitHubで対応するバージョンのSDKを取得できないかを確認してください。

COS STS SDK

COSはSTS向けにSDKおよびサンプルを提供しています。現在はJava、Nodejs、PHP、Python、Goなどの複数の言語のサンプルがあります。具体的な内容についてはCOS STS SDKをご参照ください。各SDKの使用説明についてはGithubのREADMEとサンプルをご参照ください。各言語のGitHubアドレスは以下の表のとおりです。

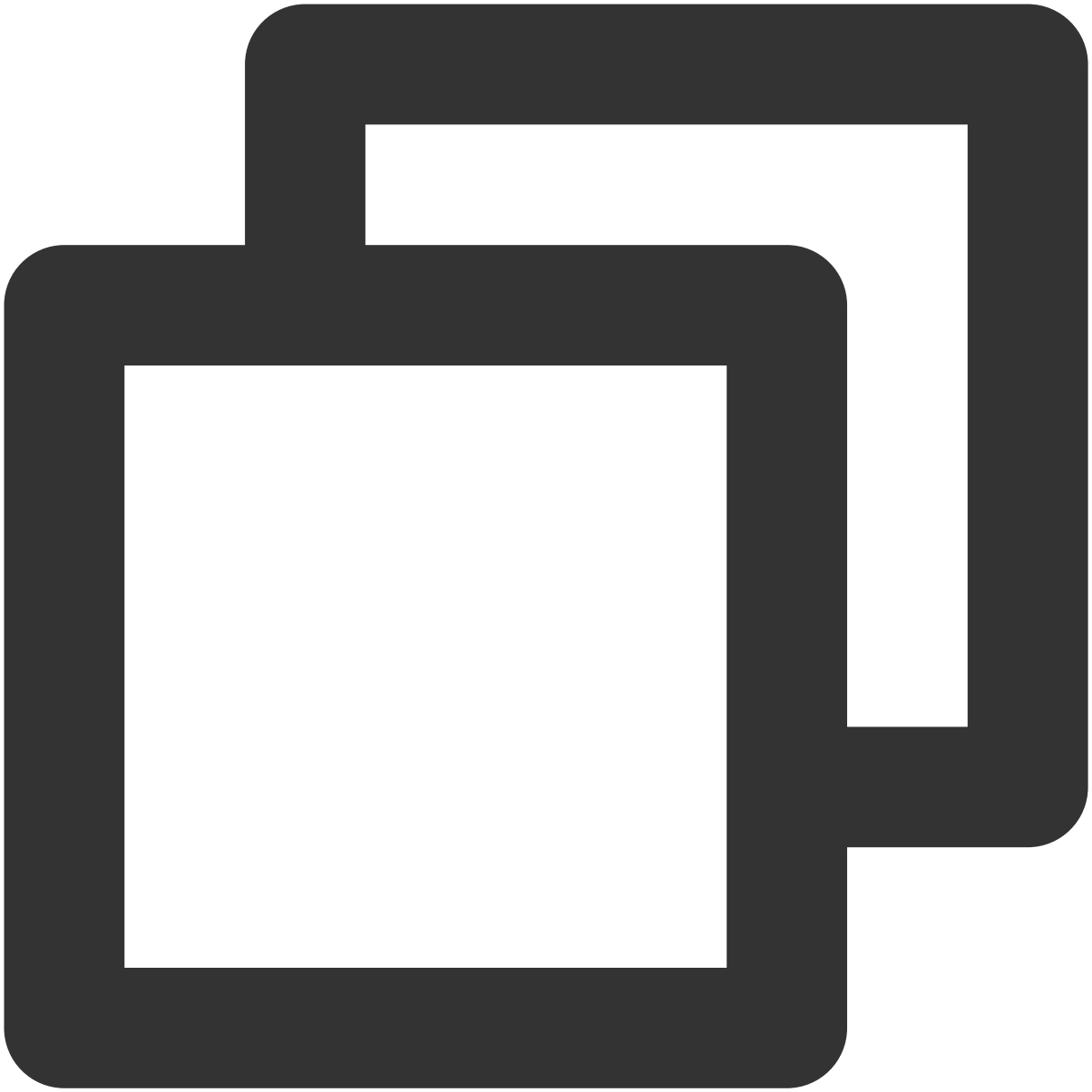
言語タイプ	コードインストールアドレス	サンプルコードアドレス
Java	インストールアドレス	サンプルアドレス
.NET	インストールアドレス	サンプルアドレス
Go	インストールアドレス	サンプルアドレス
NodeJS	インストールアドレス	サンプルアドレス
PHP	インストールアドレス	サンプルアドレス
Python	インストールアドレス	サンプルアドレス

注意：

STS SDKはSTSインターフェース自体のバージョン間の違いを非表示にするため、返されるパラメータの構造がSTSインターフェースと完全には一致しない場合があります。詳細については、Java SDKドキュメントをご参照ください。

例えばJava SDKをご利用中の場合、まずJava SDKをダウンロードした後、次の一時キー取得のサンプルを実行してください。

サンプルコード



```
// githubが提供するmaven統合メソッドによってjava sts sdkをインポートします。3.1.0以上のバージ
public class Demo {
    public static void main(String[] args) {
        TreeMap<String, Object> config = new TreeMap<String, Object>();

        try {
            //ここでのSecretIdとSecretKeyは一時キーの申請に用いる永続的なID（ルートアカウント
            String secretId = System.getenv("secretId");//ユーザーのSecretIdです。サブ
             String secretKey = System.getenv("secretKey");//ユーザーのSecretKeyです。
            // ご自身のCloud APIキーのSecretIdに置き換えます
            config.put("secretId", secretId);
```

```
        // ご自身のCloud APIキーのSecretKeyに置き換えます
        config.put("secretKey", secretKey);

        // ドメイン名を設定します:
        // Tencent Cloud CVMを使用している場合は、内部ドメイン名を設定できます
        //config.put("host", "sts.internal.tencentcloudapi.com");

        // 一時キーの有効期間の単位は秒であり、デフォルトでは1800秒です。現在はルートアカウン
        config.put("durationSeconds", 1800);

        // ご自身のbucketに置き換えます
        config.put("bucket", "examplebucket-1250000000");
        // bucketの所在リージョンに置き換えます
        config.put("region", "ap-guangzhou");

        // ここを許可するパスのプレフィックスに変更します。ご自身のウェブサイトのユーザーログ
        // いくつかの典型的なプレフィックスによる権限承認のケースを挙げます。
        // 1、すべてのオブジェクトへのアクセスを許可する："*"
        // 2、指定のオブジェクトへのアクセスを許可する："a/a1.txt", "b/b1.txt"
        // 3、指定のプレフィックスを持つオブジェクトへのアクセスを許可する："a*", "a/*", "
        // 「*」を入力すると、ユーザーにすべてのリソースへのアクセスを許可することになります。
        config.put("allowPrefixes", new String[] {
                "exampleobject",
                "exampleobject2"
        });

        // キーの権限リストです。ここで今回の一時キーに必要な権限を指定する必要があります。
        // シンプルアップロード、フォームアップロード、マルチパートアップロードには次の権限が
        String[] allowActions = new String[] {
                 // シンプルアップロード
                "name/cos:PutObject",
                // フォームアップロード、ミニプログラムアップロード
                "name/cos:PostObject",
                // マルチパートアップロード
                "name/cos:InitiateMultipartUpload",
                "name/cos:ListMultipartUploads",
                "name/cos:ListParts",
                "name/cos:UploadPart",
                "name/cos:CompleteMultipartUpload"
        };
        config.put("allowActions", allowActions);
            /**
         * conditionの設定（必要な場合）
         //# 一時キーの発効条件、conditionに関する詳細な設定ルールおよびCOSがサポートするc
         final String raw_policy = "{\\n" +
         "  \\"version\\":\\"2.0\\",\\n" +
         "  \\"statement\\":[\\n" +
```

```
                "    {\\n" +
                "      \\"effect\\":\\"allow\\",\\n" +
                "      \\"action\\":[\\n" +
                "          \\"name/cos:PutObject\\",\\n" +
                "          \\"name/cos:PostObject\\",\\n" +
                "          \\"name/cos:InitiateMultipartUpload\\",\\n" +
                "          \\"name/cos:ListMultipartUploads\\",\\n" +
                "          \\"name/cos:ListParts\\",\\n" +
                "          \\"name/cos:UploadPart\\",\\n" +
                "          \\"name/cos:CompleteMultipartUpload\\"\\n" +
                "        ],\\n" +
                "      \\"resource\\":[\\n" +
                "          \\"qcs::cos:ap-shanghai:uid/1250000000:examplebucket-125000
                "      ],\\n" +
                "      \\"condition\\": {\\n" +
                "        \\"ip_equal\\": {\\n" +
                "            \\"qcs:ip\\": [\\n" +
                "                \\"192.168.1.0/24\\",\\n" +
                "                \\"101.226.100.185\\",\\n" +
                "                \\"101.226.100.186\\"\\n" +
                "            ]\\n" +
                "        }\\n" +
                "      }\\n" +
                "    }\\n" +
                "  ]\\n" +
                "}";

                config.put("policy", raw_policy);
                */


            Response response = CosStsClient.getCredential(config);
            System.out.println(response.credentials.tmpSecretId);
            System.out.println(response.credentials.tmpSecretKey);
            System.out.println(response.credentials.sessionToken);
        } catch (Exception e) {
            e.printStackTrace();
            throw new IllegalArgumentException("no valid secret !");
        }
    }
}
```

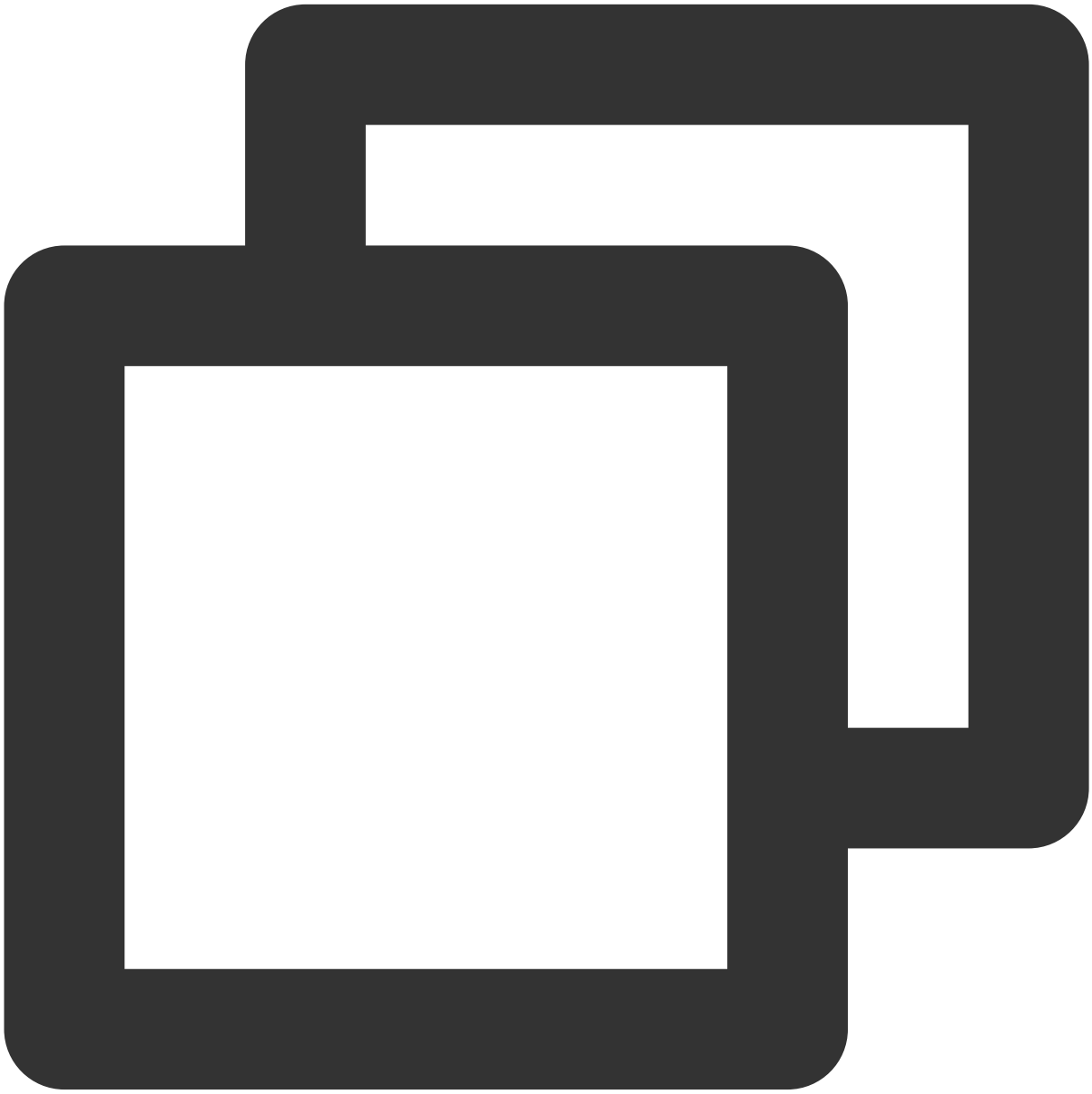
よくあるご質問と回答

JSONObjectパッケージの競合によるNoSuchMethodError

3.1.0以上のバージョンを使用してください。

一時キーを使用したCOSアクセス

COS APIが一時キーを使用してCOSサービスにアクセスする際、x-cos-security-tokenフィールドによって一時sessionTokenを渡し、一時SecretIdおよびSecretKeyによって署名を計算します。
COS Java SDKの例では、一時キーを使用したCOSアクセスのサンプルは次のようになります。
説明：
次のサンプルを実行する前に、GithubプロジェクトでJava SDKインストールパッケージを取得してください。



```
// githubが提供するmaven統合メソッドによってcos xml java sdkをインポートします
import com.qcloud.cos.*;
import com.qcloud.cos.auth.*;
```

```
import com.qcloud.cos.exception.*;
import com.qcloud.cos.model.*;
import com.qcloud.cos.region.*;
public class Demo {
    public static void main(String[] args) throws Exception {

        // ユーザー基本情報
        String tmpSecretId = "COS_SECRETID";    // STSインターフェースから返された一時Sec
        String tmpSecretKey = "COS_SECRETKEY";  // STSインターフェースから返された一時Se
        String sessionToken = "Token";  // STSインターフェースから返された一時Tokenに置き

        // 1 ユーザーID情報(secretId, secretKey)を初期化します
        COSCredentials cred = new BasicCOSCredentials(tmpSecretId, tmpSecretKey);
        // 2 bucketのリージョンを設定します。詳細についてはCOSリージョン https://cloud.tence
        ClientConfig clientConfig = new ClientConfig(new Region("ap-guangzhou"));
        // 3 cosクライアントを生成します。
        COSClient cosclient = new COSClient(cred, clientConfig);
        // bucket名にはappidを含める必要があります
        String bucketName = "examplebucket-1250000000";

        String key = "exampleobject";
        // objectをアップロードします。20M以下のファイルにはこのインターフェースの使用をお勧めし
        File localFile = new File("src/test/resources/text.txt");
        PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, l

        // x-cos-security-token headerフィールドを設定します
        ObjectMetadata objectMetadata = new ObjectMetadata();
        objectMetadata.setSecurityToken(sessionToken);
        putObjectRequest.setMetadata(objectMetadata);

        try {
            PutObjectResult putObjectResult = cosclient.putObject(putObjectRequest)
            // 成功：putobjectResultはファイルのetagを返します
            String etag = putObjectResult.getETag();
        } catch (CosServiceException e) {
            //失敗、CosServiceExceptionをスローします
            e.printStackTrace();
        } catch (CosClientException e) {
            //失敗、CosClientExceptionをスローします
            e.printStackTrace();
        }

        // クライアントを閉じる
        cosclient.shutdown();

    }
}
```

サブアカウントを権限承認し、バケットタグに基づきバケットリストをプルします

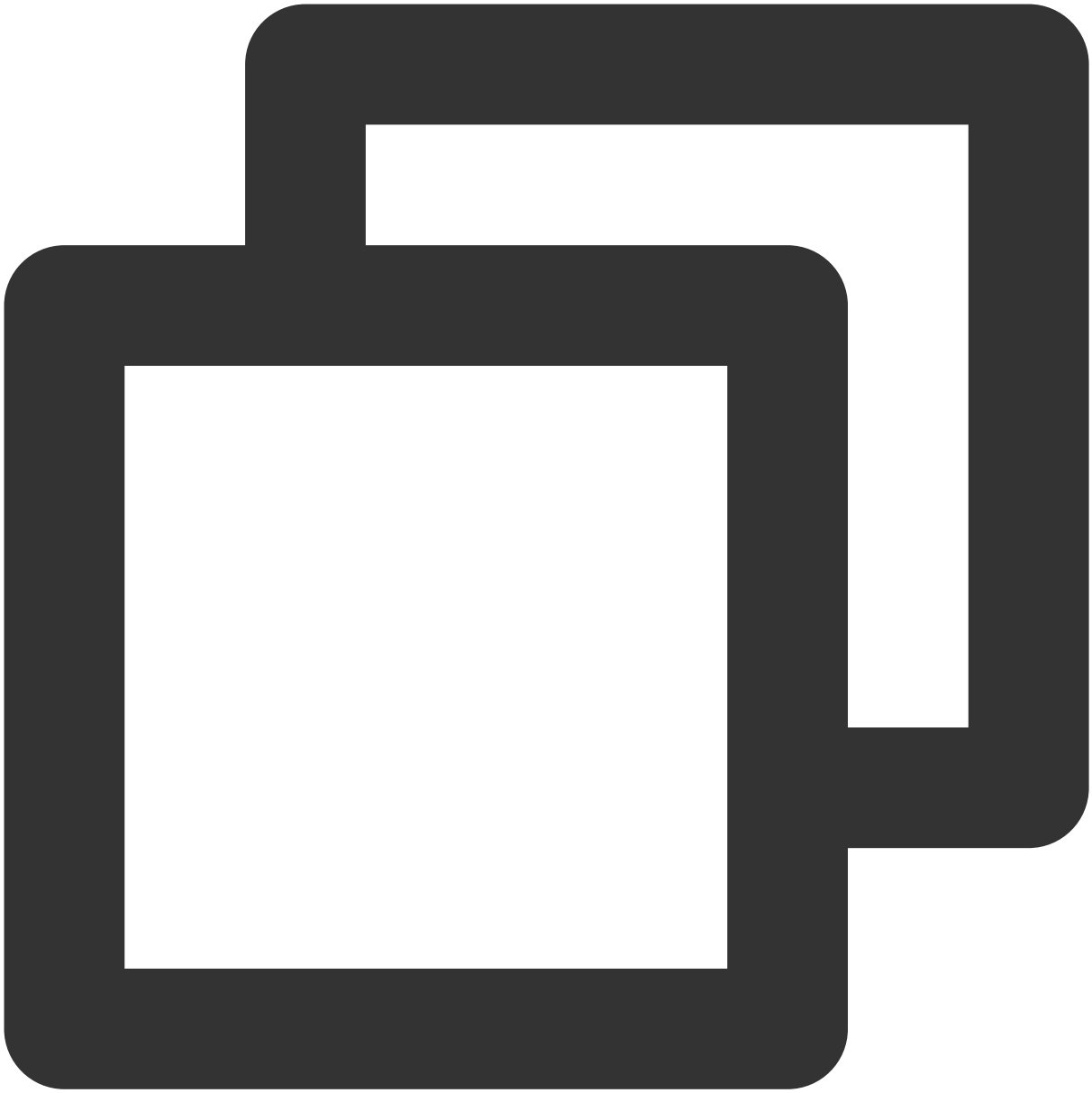
最終更新日：：2024-06-26 10:42:27

概要

Cloud Object Storage（COS）コンソール、APIは、バケットタグに基づいてバケットリストをフィルタリングする機能を提供しています。この機能はタグの権限を承認することで実装します。

権限承認の手順

1. ルートアカウントOwnerを使用して、CAMコンソールにログインし、ポリシー設定ページに進みます。
2. 次の手順に従って、サブアカウントSubUserに、指定のタグを持つバケットへのアクセス権限を承認します。**ポリシージェネレーター**または**ポリシー構文**で実装できます。
ポリシージェネレーターの承認
ユーザーポリシー構文の承認
1. CAMのポリシー設定ページに進みます。
2. **カスタムポリシーの新規作成 > ポリシージェネレーターに従って作成**をクリックします。
3. 権限承認の設定ページに移動します。設定情報は以下のとおりです。
効果：許可を選択します。デフォルトは変わりません。
サービス：COSを選択します。
アクション：**読み取り操作 > GetService(バケットリストのプル)**を選択します。
リソース：**すべてのリソース**を選択します。
条件：**その他の条件を追加**をクリックします。サイドウィンドウで次の情報の設定を行います。
条件キー：`qcs:resource_tag` を選択します。
演算子：`string_equal` を選択します。
条件値：`key&val` 形式でタグを入力し、keyをタグキーに、valueをタグ値にそれぞれ置き換えます。
4. **次のステップ**をクリックし、ポリシー名を入力します。
5. **完了**をクリックすれば、作成完了です。
1. CAMのポリシー設定ページに進みます。
2. **カスタムポリシーの新規作成 > ポリシー構文に従って作成**をクリックします。
3. 空白テンプレートによる作成を選択し、**次のステップ**をクリックします。
4. 次のポリシー形式に従って入力します。このうち、keyとvalueはそれぞれ指定のタグキーとタグ値に置き換えます。



```
{
 "version": "2.0",
 "statement":[
     {
         "effect": "allow",
         "action":[
             "name/cos:GetService"
         ],
         "resource": "*",
         "condition":{
             "for_any_value:string_equal":{
```

```
                "qcs:resource_tag": [
                    "key&value"
                ]
            }
        }
    }
  ]
}
```

5. **完了**をクリックすれば、作成完了です。

3. ポリシーをサブアカウントSubUserにバインドします。ポリシーページで、ステップ2で作成したポリシーを見つけ、右側の**ユーザー/グループ/ロールのバインド**をクリックします。

4. ポップアップウィンドウで、サブアカウントSubUserにチェックを入れ、**OK**をクリックすると、サブアカウントSubUserをそのポリシーにバインドできます。

コンソールでの確認

1. サブアカウントSubUserでCOSコンソールにログインします。

2. バケットリストページには、**このサブアカウントがアクセス権限を持つバケットのリストが自動的に表示されます**。

上記の手順によって、指定のタグ（key、value）を持つバケットへのアクセス権限がサブアカウントに承認されました。

インターフェースの呼び出し

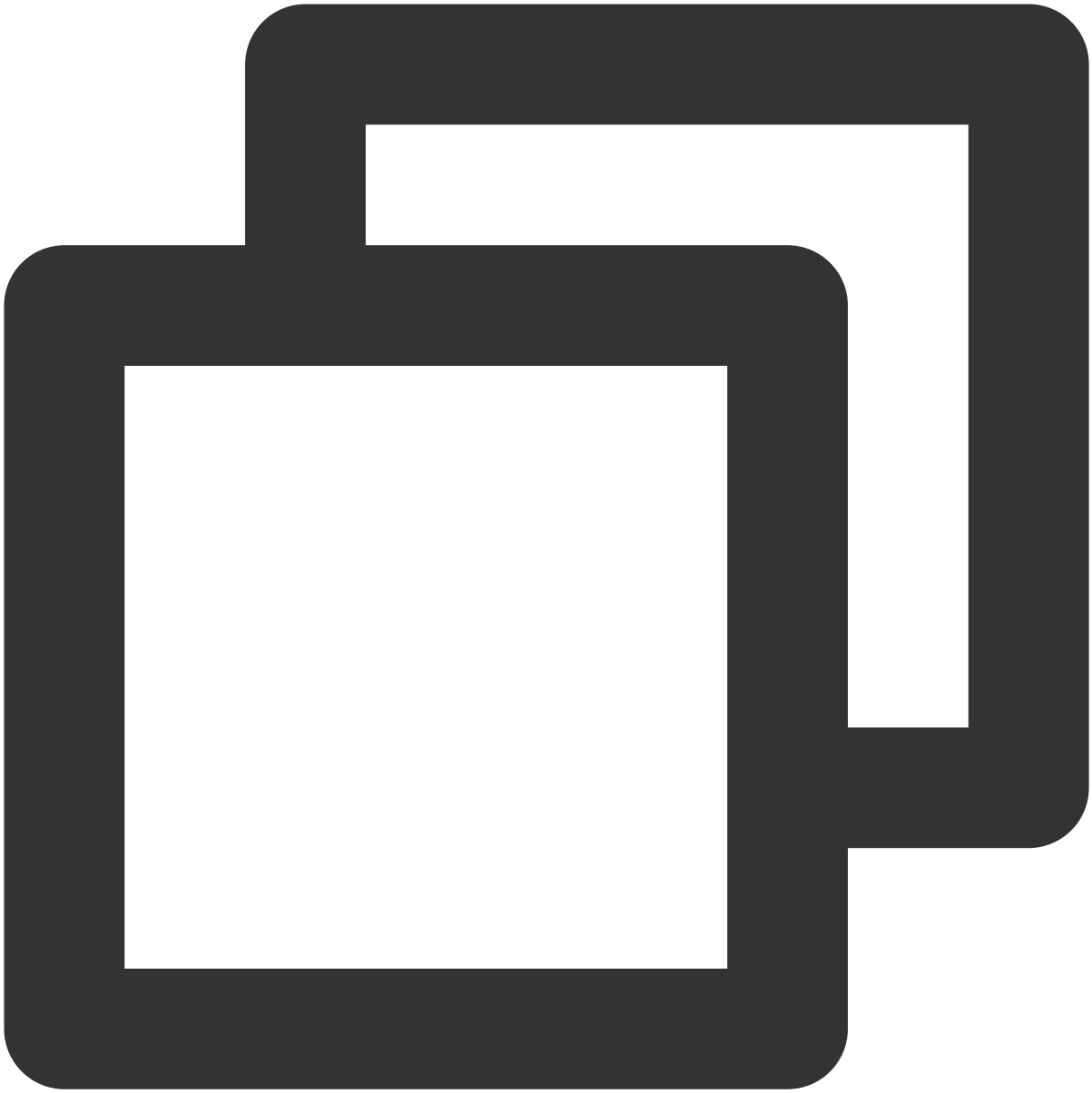
注意：

コンソールと異なり、GetService APIインターフェースの呼び出しでは、サブアカウントがアクセス権限を持つバケットのリストは自動表示されません。タグパラメータを渡す必要があります。

現在GetServiceインターフェースで渡すことができるタグは1つのみです。

1. サブアカウントSubUserのキーを使用してリクエストを送信します。

2. GetServiceインターフェースを呼び出し、(key,value)などのタグフィルタリングパラメータを渡します。リクエストの例は次のとおりです。詳細についてはGET Service（List Buckets）をご参照ください。



```
GET /?tagkey=key1&tagvalue=value1 HTTP/1.1
Date: Fri, 24 May 2019 11:59:51 GMT
Authorization: Auth String
```

条件キーの説明および使用例

最終更新日：：2024-06-26 10:42:27

アクセスポリシーを使用して権限承認を行う場合、ポリシーの発効条件を指定することができます。例えば、ユーザーのアクセス元、アップロードファイルのストレージタイプなどの制限があります。
ここでは、バケットポリシーにおけるCloud Object Storage（COS）条件キー使用の一般的な例について記載します。発効条件のドキュメント内で、COSがサポートするすべての条件キーおよび適用可能なリクエストを確認することができます。
説明：
条件キーを使用してポリシーを作成する際は、必ず最小権限の原則を遵守し、適用可能なリクエスト（action）のみに該当する条件キーを追加してください。アクション（action）を指定する際にワイルドカード「*」を使用すると、リクエストが失敗しますので避けてください。条件キーに関する説明は、発効条件のドキュメントをご参照ください。
Cloud Access Management（CAM）コンソールを使用してポリシーを作成する際は、構文形式にご注意ください。version、principal、statement、effect、action、resource、conditionの構文要素はアルファベットの先頭の文字を大文字にするか、またはすべて小文字にする必要があります。

条件キーの使用例

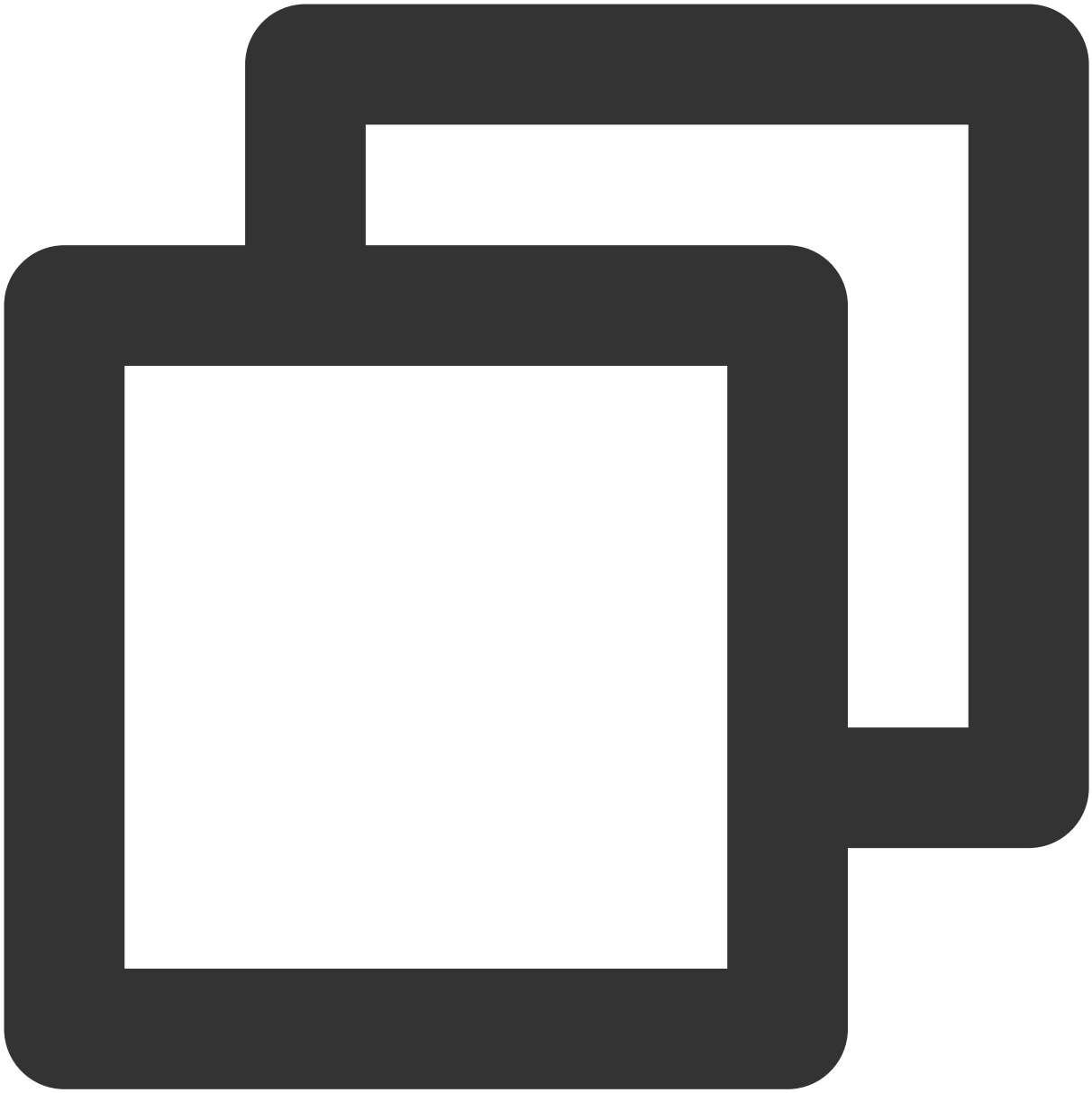
ユーザーアクセスIPの制限（qcs:ip）

条件キー qcs:ip

条件キー `qcs:ip` を使用してユーザーアクセスIPを制限します。すべてのリクエストに適用可能です。

例：指定IPからのユーザーアクセスのみを許可

次のポリシーの記述例は、ルートアカウントID 100000000001（APPIDは1250000000）下のサブアカウントID 100000000002に対し、北京リージョンのバケットのexamplebucket-bjおよび広州リージョンのバケットのexamplebucket-gz下のオブジェクトexampleobjectについて、アクセスIPが `192.168.1.0/24` ネットワークセグメントにある場合およびIPが `101.226.100.185` または `101.226.100.186` である場合に、オブジェクトのアップロードおよびオブジェクトのダウンロード権限を許可するものです。



```
{
    "version": "2.0",
    "principal":{
        "qcs": [
            "qcs::cam::uin/100000000001:uin/100000000002"
        ]
    },
    "statement":[
        {
            "effect": "allow",
            "action":[
```

```
                "name/cos:PutObject",
                "name/cos:GetObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-gz-1250000000/e
            ],
            "condition":{
                "ip_equal":{
                    "qcs:ip":[
                        "192.168.1.0/24",
                        "101.226.100.185",
                        "101.226.100.186"
                    ]
                }
            }
        }
    ]
}
```

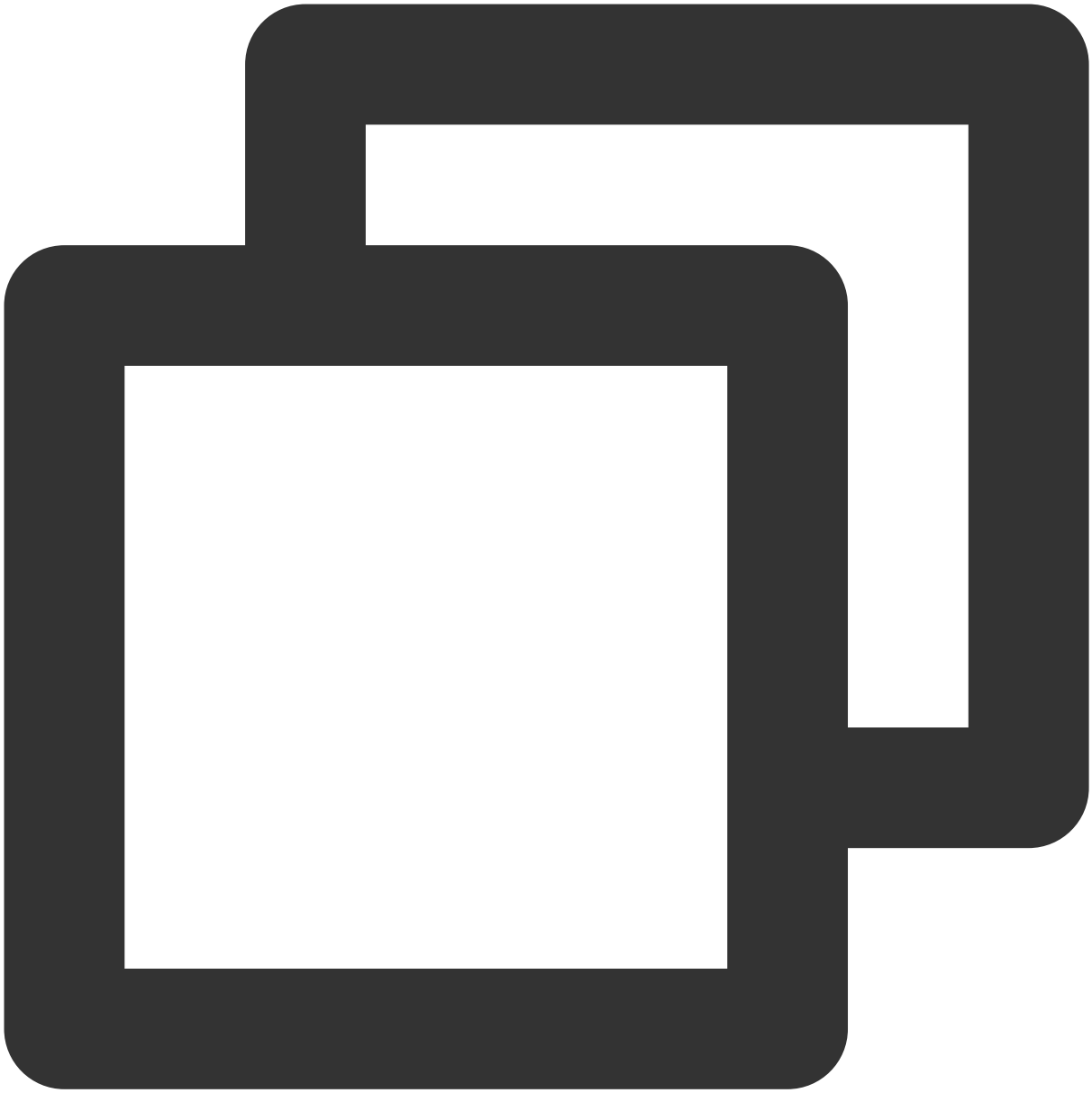
vpcidの制限（vpc:requester_vpc）

条件キー vpc:requester_vpc

条件キー `vpc:requester_vpc` を使用してユーザーアクセスのvpcidを制限します。vpcidに関するその他の説明については、Tencent Cloud製品VPCをご参照ください。

例：vpcidをaqp5jrc1に制限

次のポリシーの記述例は、ルートアカウントID 100000000001（APPIDは1250000000）下のサブアカウントID 100000000002によるバケットexamplebucket-1250000000へのアクセスについて、vpcidがaqp5jrc1の場合にリクエストが権限を得ることを許可するものです。



```
{
  "statement":[
    {
      "action":[
        "name/cos:*"
      ],
      "condition":{
        "string_equal":{
          "vpc:requester_vpc": [
            "vpc-aqp5jrc1"
          ]
```

```
          }
        },
        "effect": "allow",
        "principal":{
          "qcs": [
            "qcs::cam::uin/100000000001:uin/100000000002"
          ]
        },
        "resource":[
          "qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/*"
        ]
      }
    ],
    "version": "2.0"
  }
```

オブジェクトの最新バージョンまたは指定されたバージョンへのアクセスのみを許可（cos:versionid）

リクエストパラメータ versionid

リクエストパラメータ `versionid` はオブジェクトのバージョン番号を表します。バージョン管理に関する内容は、バージョン管理の概要をご参照ください。オブジェクトのダウンロード（GetObject）、オブジェクトの削除（DeleteObject）の際に、リクエストパラメータ `versionid` を使用して、アクションを行いたいオブジェクトのバージョンを指定することができます。
`versionid` が含まれないリクエストパラメータの場合、リクエストはデフォルトでオブジェクトの最新バージョンに作用します。
`versionid` リクエストパラメータを空の文字列にすると、 `versionid` が含まれないリクエストパラメータの場合と同じになります。
`versionid` リクエストパラメータが文字列 `"null"` の場合、あるバケットにバージョン管理を有効にする前にオブジェクトをアップロードし、その後バージョン管理を有効にすると、それらのオブジェクトのバージョン番号はすべて文字列 `"null"` となります。

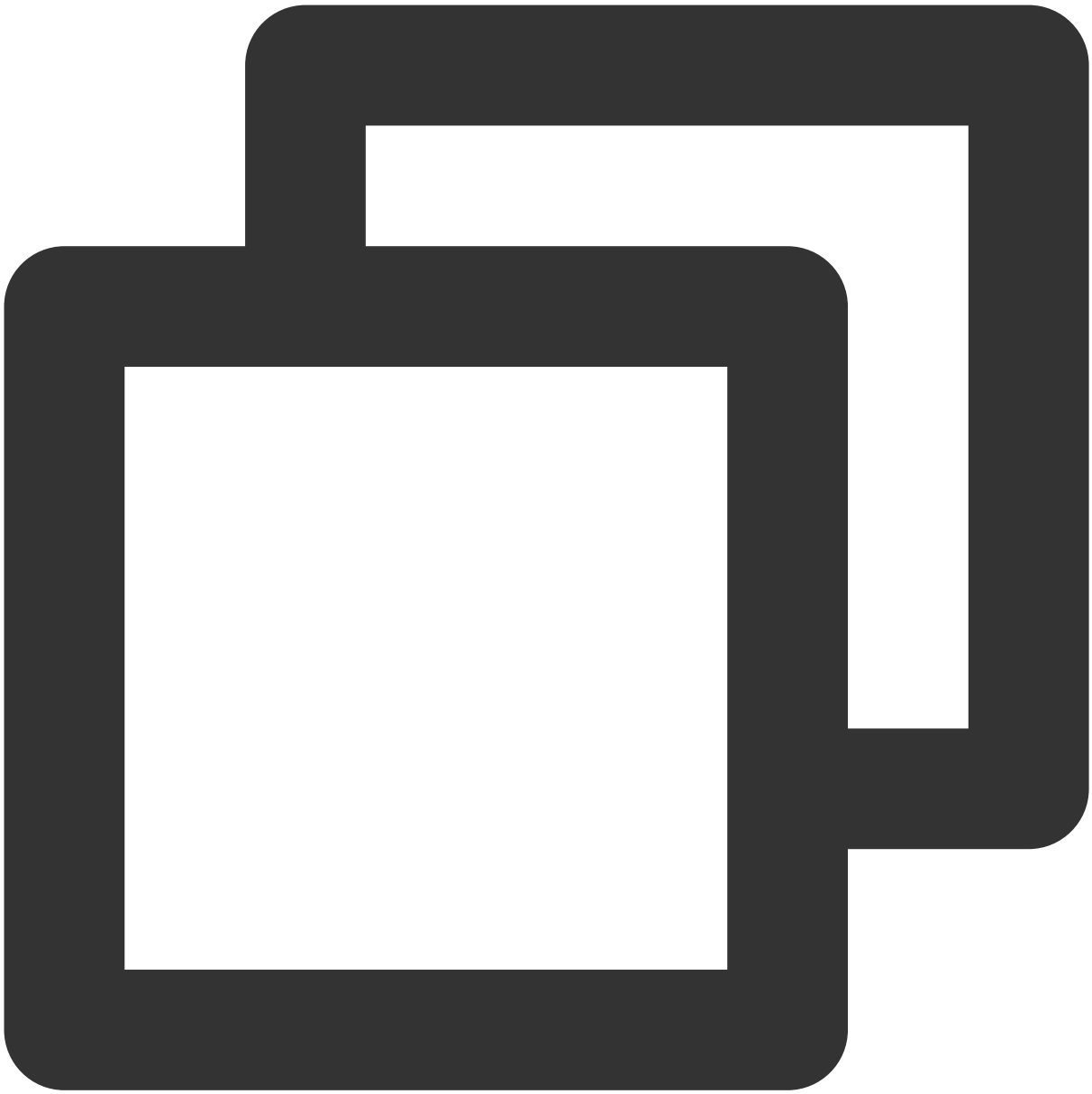
条件キー cos:versionid

条件キー `cos:versionid` はリクエストパラメータ `versionid` の制限に用いられます。

事例1：指定されたバージョン番号のオブジェクトの取得のみをユーザーに許可する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有し、そのサブユーザー（uin:100000000002）に対し権限承認を行い、指定されたバージョン番号のオブジェクトの取得のみをサブユーザーに許可する必要があるとします。
次のバケットポリシーを採用した後、サブユーザー（uin:100000000002）がオブジェクトダウンロードリクエストを送信した場合、versionidパラメータが含まれ、なおかつversionidの値がバージョン番号

「MTg0NDUxNTc1NjIzMTQ1MDAwODg」である場合にのみ、リクエストが成功します。

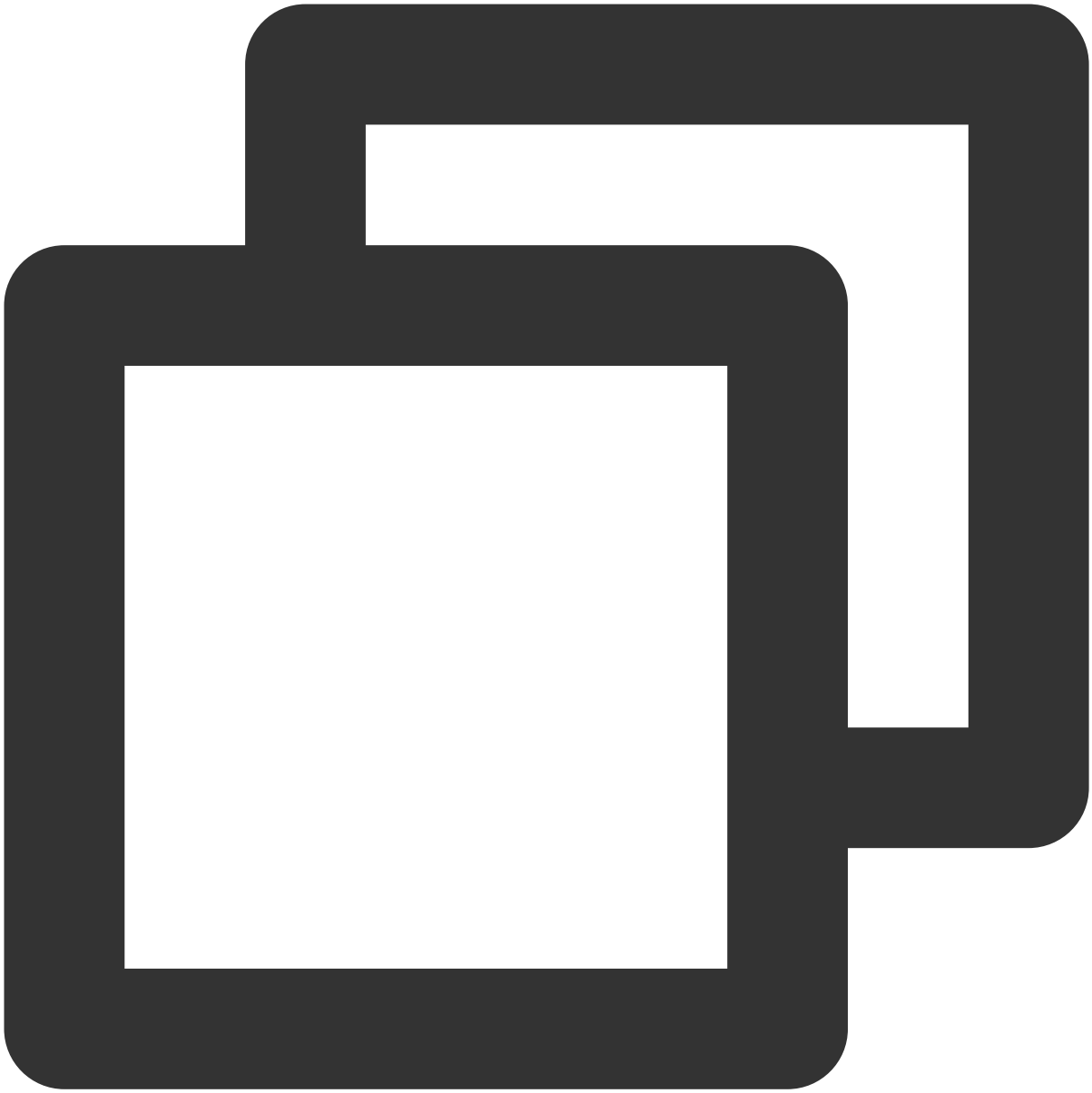


```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
```

```
            "effect":"allow",
            "action":[
                "name/cos:GetObject"
            ],
            "condition":{
                "string_equal":{
                    "cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ]
        }
    ]
}
```

明示的な拒否の追加

上記のポリシーを使用してサブユーザーへの権限承認を行う場合、サブユーザーが他の手段によって何の条件もない同一の権限を取得しているために、より広範囲の権限が発効する可能性があります。例えば、サブユーザーがあるユーザーグループに所属しており、ルートアカウントがユーザーグループにGetObject権限を与え、それに何の条件も付け加えなかった場合、上記のポリシーはバージョン番号に対する制限の役割を発揮しません。
このような状況に対応するため、上記のポリシーをベースにした上で、明示的な拒否のポリシー（deny）を追加することで、より厳格な権限制限を行うことができます。下記におけるこのdenyポリシーは、サブユーザーがオブジェクトダウンロードリクエストを送信する際に、 versionidパラメータを含めなかった場合、またはversionidのバージョン番号が「MTg0NDUxNTc1NjIzMTQ1MDAwODg」ではなかった場合に、そのリクエストは拒否されることを意味します。denyの優先順位は他のポリシーより高いため、明示的な拒否の追加によって、権限の脆弱性を最も高いレベルで回避することができます。



```
{
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:GetObject"
```

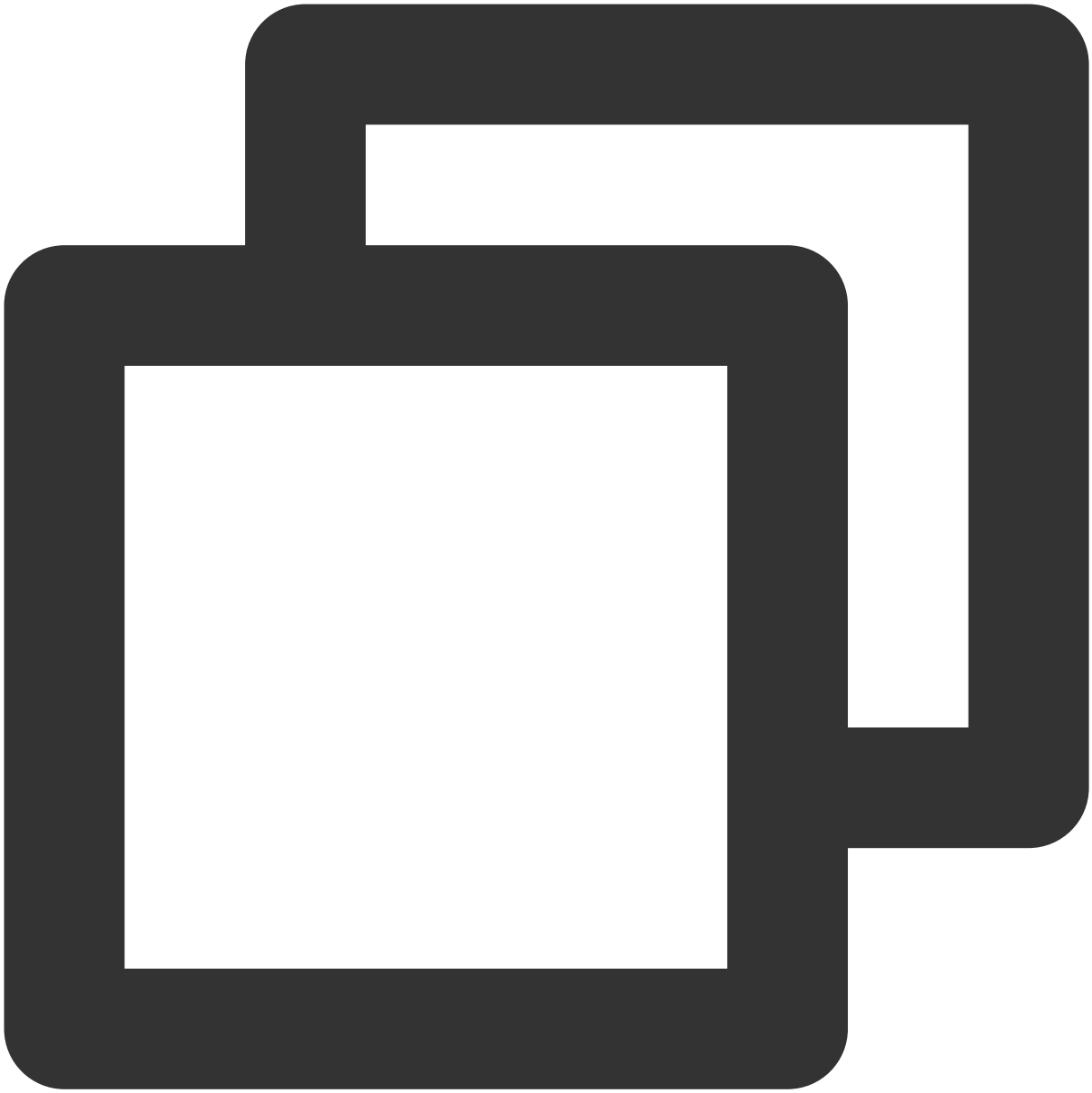
```
            ],
            "condition":{
                "string_equal":{
                    "cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ]
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:GetObject"
            ],
            "condition":{
                "string_not_equal_if_exist":{
                    "cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ]
        }
    ],
    "version":"2.0"
}
```

事例2：最新バージョンのオブジェクトの取得のみをユーザーに許可する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有し、そのサブユーザー（uin:100000000002）が最新バージョンのオブジェクトのみを取得できるように制限する必要があるとします。リクエストパラメータ `versionid` が含まれない場合、または `versionid` が空の文字列の場合、GetObjectはデフォルトで最新バージョンのオブジェクトを取得します。このため、条件の中でstring_equal_if_exsitを使用することができます。

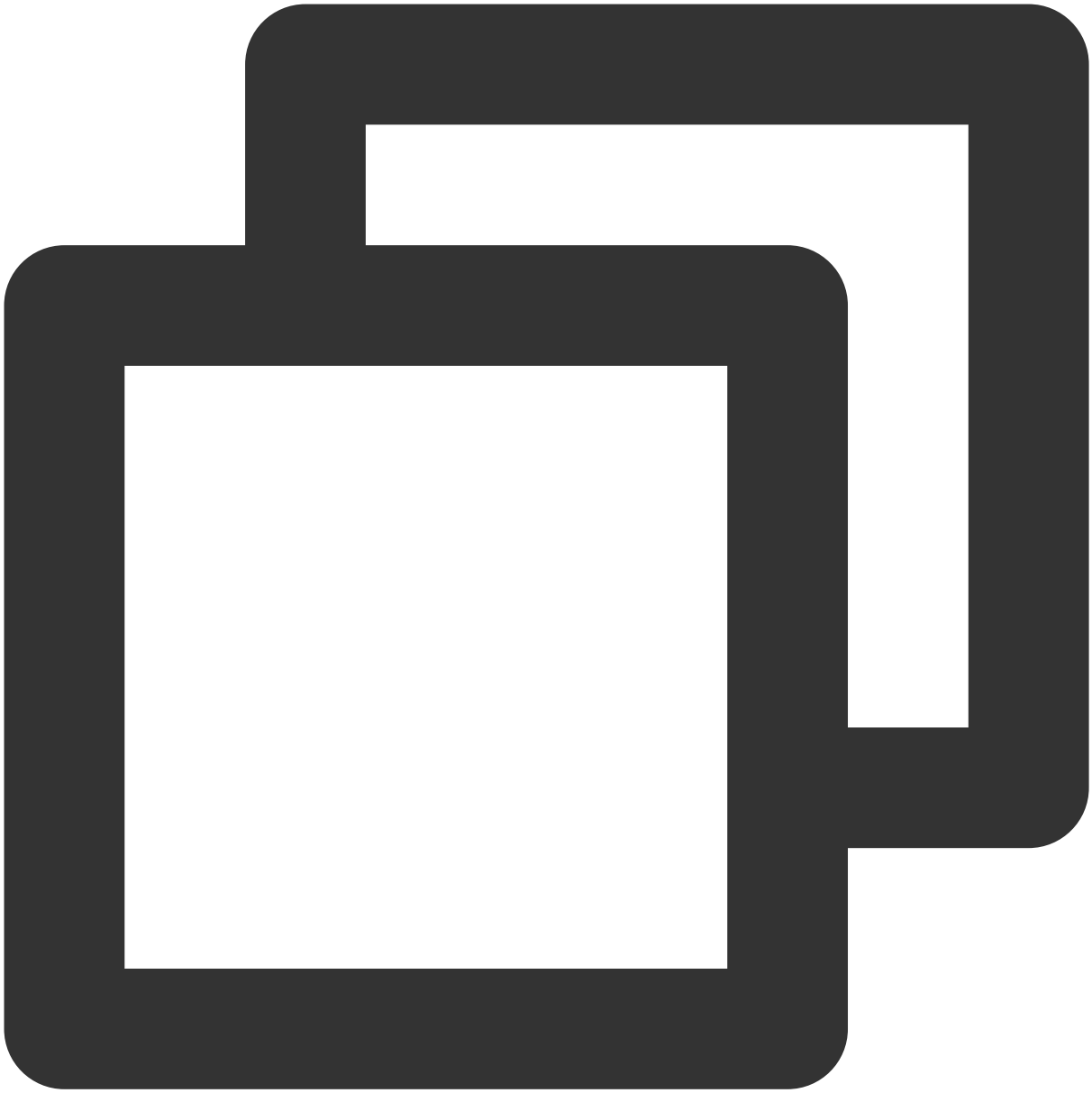
1. versionidが含まれない場合は、デフォルトでtrueとして処理し、allow条件にヒットすれば、リクエストはallowされます。

2. リクエストパラメータversionidが空、すなわち `“”` の場合も同様にallowポリシーにヒットし、最新バージョンのオブジェクト取得に対するリクエストのみ権限が承認されます。



```
"condition":{
    "string_equal_if_exist":{
        "cos:versionid": ""
    }
}
```

明示的な拒否を追加した、完全なバケットポリシーは次のようになります。



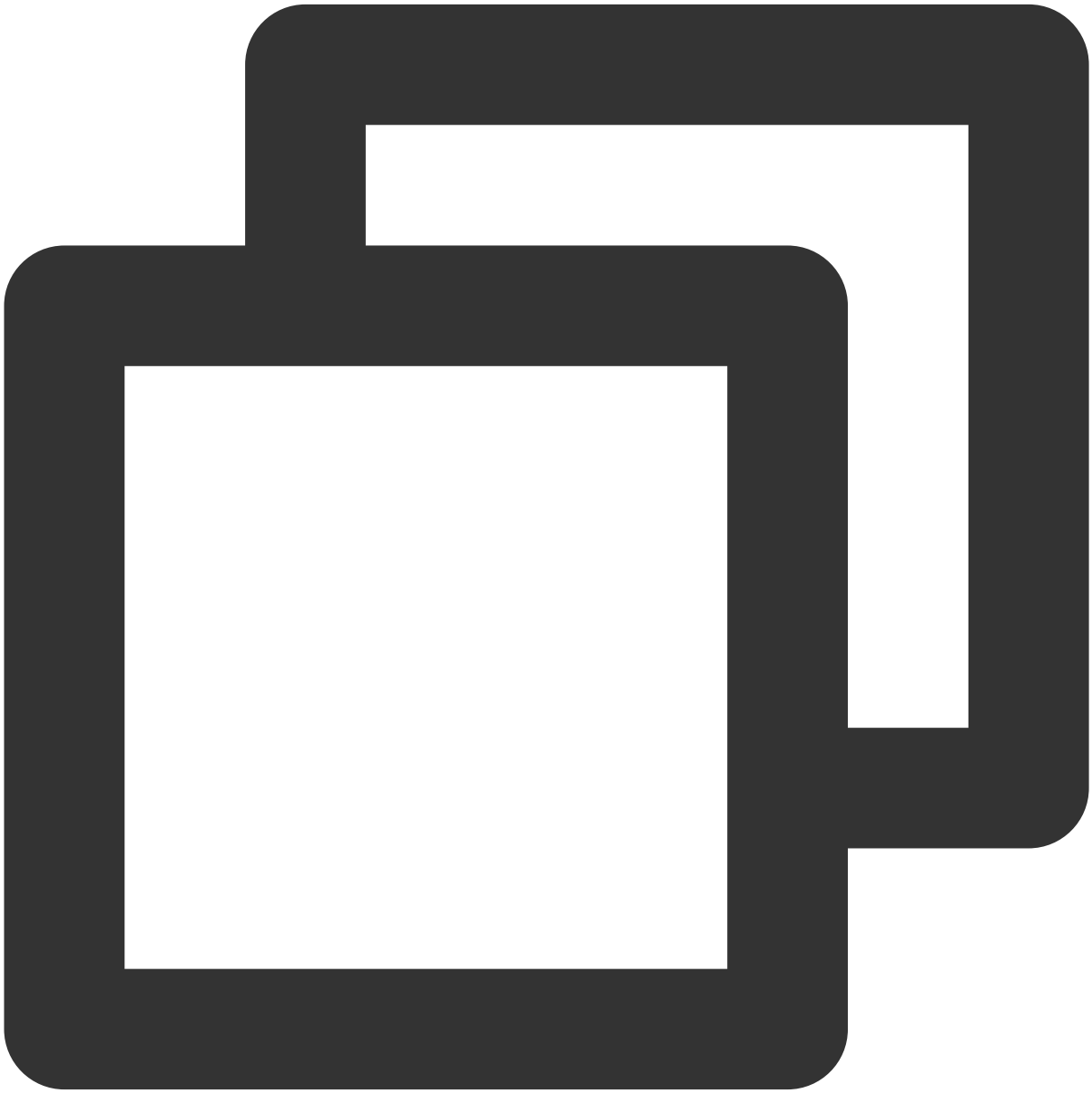
```
{
 "statement":[
     {
         "principal":{
             "qcs":[
                 "qcs::cam::uin/100000000001:uin/100000000002"
             ]
         },
         "effect":"allow",
         "action":[
             "name/cos:GetObject"
```

```
            ],
            "condition":{
                "string_equal_if_exist":{
                    "cos:versionid":""
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ]
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:GetObject"
            ],
            "condition":{
                "string_not_equal":{
                    "cos:versionid":""
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ]
        }
    ],
    "version":"2.0"
}
```

事例3：バージョン管理の有効化前にアップロードしたオブジェクトの削除をユーザーに許可しない

バージョン管理を有効化する前に、一部のオブジェクトがバケットにアップロードされている可能性があり、それらのオブジェクトのバージョン番号は“null”となります。これらのオブジェクトに対しては、場合によっては追加の保護を有効化する必要があります。例えば、ユーザーに対しこれらのオブジェクトの完全削除操作を禁止する、すなわちバージョン番号があるものの削除操作を拒否するなどです。
次に挙げるこのバケットポリシーの例には、2つのポリシーが含まれます。
1. DeleteObjectリクエストを使用してバケット内のオブジェクトを削除する権限をサブユーザーに承認します。
2. DeleteObjectリクエストの発効条件を制限しています。DeleteObjectリクエストにリクエストパラメータversionidが含まれ、なおかつversionidが「null」の場合は、このDeleteObjectリクエストを拒否します。

このため、バケットexamplebucket-1250000000に先にオブジェクトAをアップロードし、その後バケットのバージョン管理を有効化した場合、オブジェクトAのバージョン番号は文字列の「null」となります。
このバケットポリシーを追加すると、オブジェクトAは保護されます。サブユーザーがオブジェクトAに対しDeleteObjectリクエストを送信し、リクエストにバージョン番号が含まれていない場合、バージョン管理を有効化しているため、オブジェクトA自体は完全削除されず、削除マーカーだけが付与されます。リクエストにAのバージョン番号「null」が含まれていた場合、このリクエストは拒否され、オブジェクトAは完全削除されることなく保護されます。



```
{
    "statement":[
```

```
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:DeleteObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ]
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:DeleteObject"
            ],
            "condition":{
                "string_equal":{
                    "cos:versionid":"null"
                }
            },
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ]
        }
    ],
    "version":"2.0"
}
```

アップロードファイルサイズの制限（cos:content-length）

リクエストヘッダーContent-Length

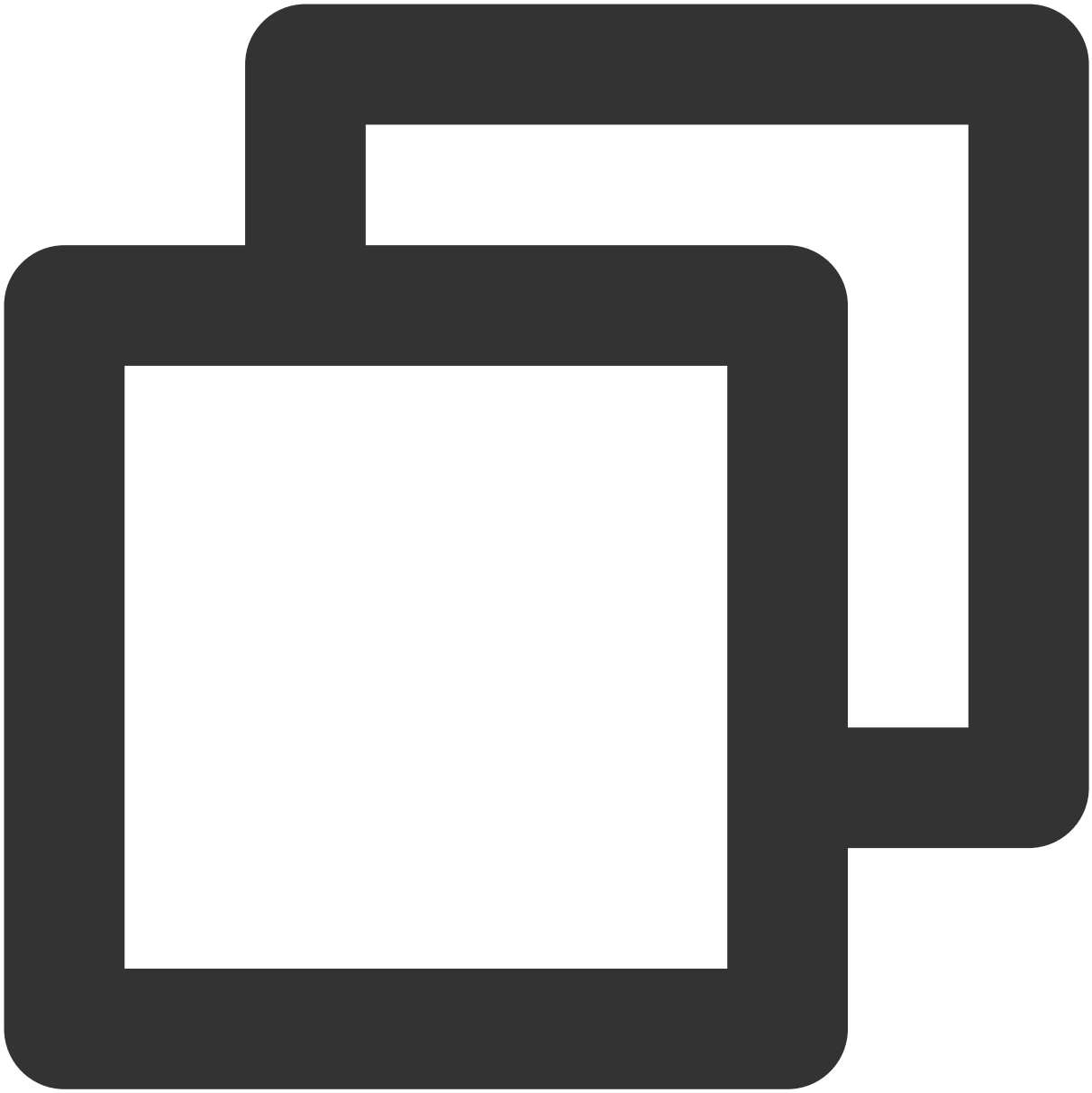
RFC 2616で定義されたHTTPリクエストのコンテンツの長さ（バイト）です。PUTおよびPOSTリクエストで頻繁に使用されます。詳細については、リクエストヘッダーリストをご参照ください

条件キーcos:content-length

オブジェクトをアップロードする際、条件キー `cos:content-length` によってリクエストヘッダー `Content-Length` を制限することができます。それによりアップロードするオブジェクトのファイルサイズを制限することで、ストレージスペースをより柔軟に管理し、大きすぎるファイルや小さすぎるファイルのアップロードによる、ストレージスペースやネットワーク帯域幅の浪費防止に役立ちます。
下記の2つの例では、ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有している場合、 `cos:content-length` 条件キーによってサブユーザー（uin:100000000002）のアップロードリクエストのContent-Lengthヘッダーのサイズを制限することができます。

事例1: リクエストヘッダーContent-Lengthの最大値を制限する

PutObjectおよびPostObjectのアップロードリクエストには必ずContent-Lengthヘッダーが含まれなければならず、かつこのヘッダーの値を10バイト以下とするよう制限します。

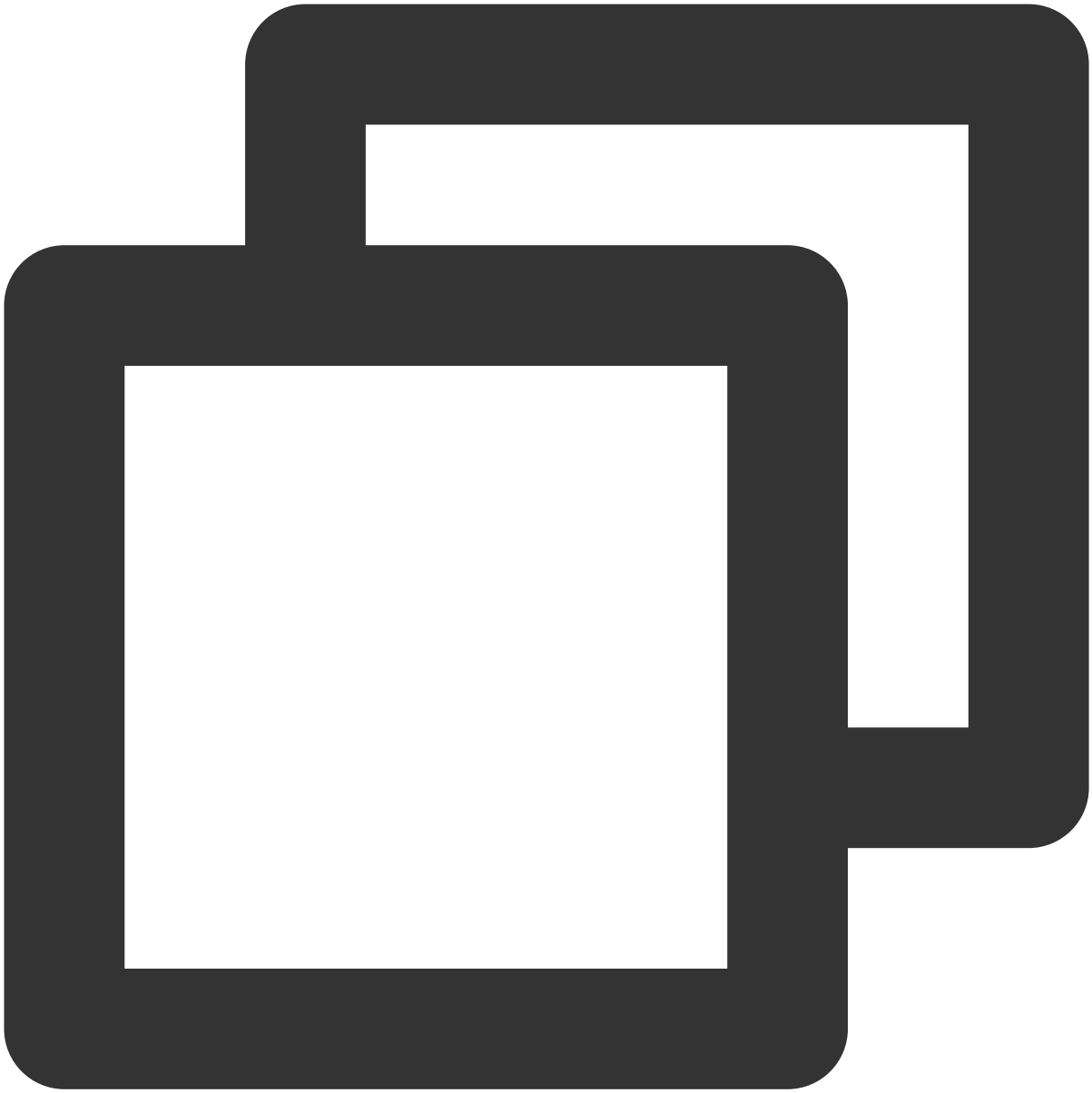


```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
```

```
                "name/cos:PutObject",
                "name/cos:PostObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "numeric_less_than_equal":{
                    "cos:content-length":10
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:PutObject",
                "name/cos:PostObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "numeric_greater_than_if_exist":{
                    "cos:content-length":10
                }
            }
        }
    ]
}
```

事例2: リクエストヘッダーContent-Lengthの最小値を制限する

PutObjectおよびPostObjectのアップロードリクエストには必ずContent-Lengthヘッダーが含まれなければならず、かつContent-Lengthの値を2バイト以上とするよう制限します。



```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
```

```
                "name/cos:PutObject",
                "name/cos:PostObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "numeric_greater_than_equal":{
                    "cos:content-length":2
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:PutObject",
                "name/cos:PostObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "numeric_less_than_if_exist":{
                    "cos:content-length":2
                }
            }
        }
    ]
}
```

アップロードファイルタイプの制限（cos:content-type）

リクエストヘッダーContent-Type

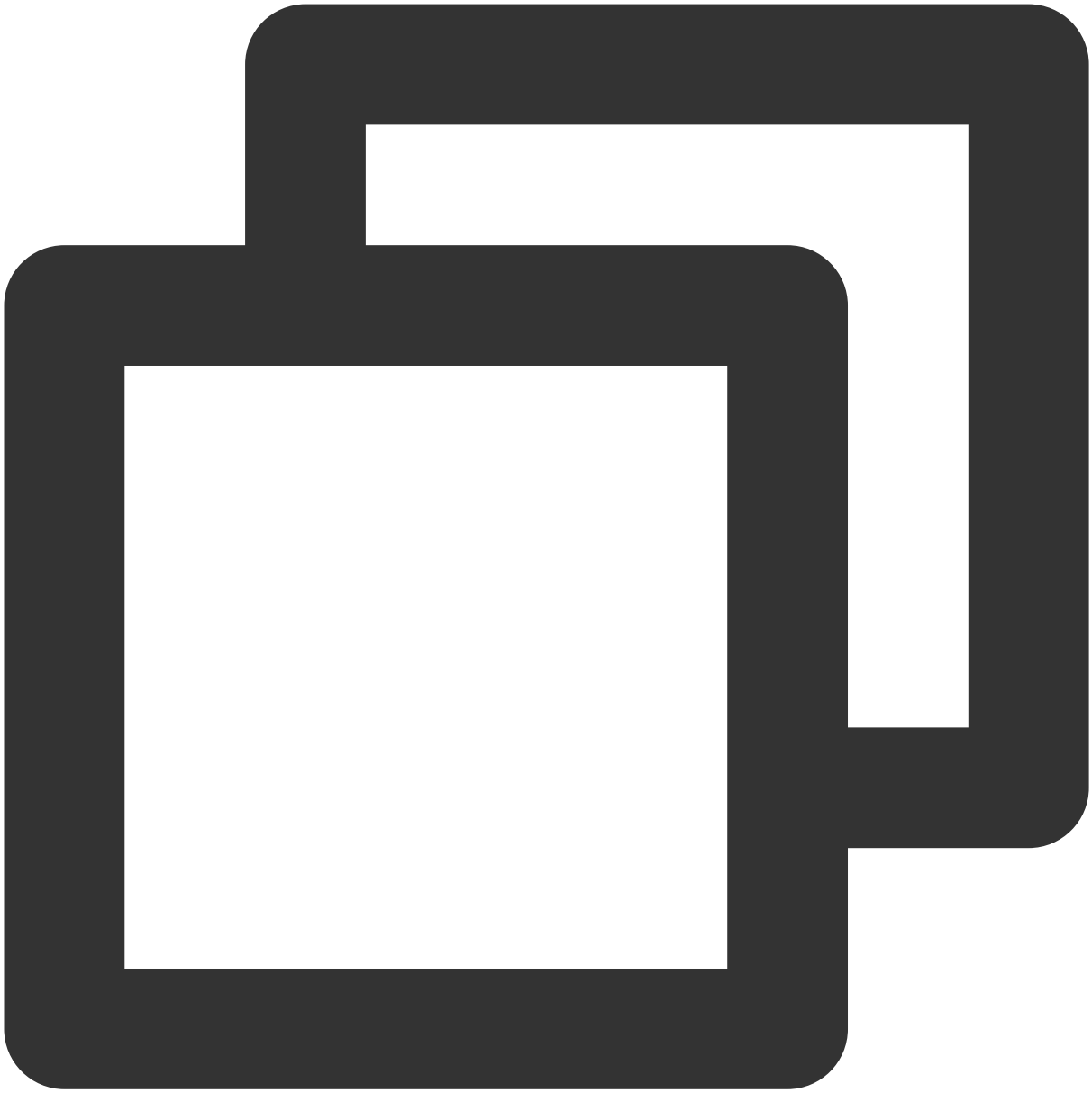
RFC 2616で定義されたHTTPリクエストのコンテンツタイプ（MIME）であり、例えば `application/xml` や `image/jpeg` などです。詳細については、リクエストヘッダーリストをご参照ください。

条件キーcos:content-type

条件キー `cos:content-type` を使用すると、リクエストの `Content-Type` ヘッダーを制限することができます。

事例1: オブジェクトのアップロード（PutObject）のContent-Typeを必ず「image/jpeg」とするよう限定する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有している場合、 `cos:content-type` 条件キーによってサブユーザー（uin:100000000002）のアップロードリクエストのContent-Typeヘッダーの具体的な内容を制限することができます。
以下のこのバケットポリシーの意味は、PutObjectを使用してオブジェクトをアップロードする場合に、必ずContent-Typeヘッダーを含め、かつContent-Typeの値を「image/jpeg」とするよう制限するものです。
注意すべきは、string_equalはリクエストに必ずContent-Typeヘッダーを含め、なおかつContent-Typeの値が規定値と完全に一致することを要求するという点です。実際のリクエストでは、**リクエストのContent-Typeヘッダーを明確に指定する**必要があります。それを行わず、リクエストにContent-Typegヘッダーが含まれない場合、リクエストは失敗します。また、何らかのツールを使用してリクエストを送信し、Content-Typeを明確に指定しなかった場合、ツールが想定と異なるContent-Typeヘッダーを自動的に追加する可能性があり、この場合もリクエストの失敗につながる可能性があります。



```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
```

```
                "name/cos:PutObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_equal":{
                    "cos:content-type":"image/jpeg"
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:PutObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_not_equal_if_exist":{
                    "cos:content-type":"image/jpeg"
                }
            }
        }
    ]
}
```

ダウンロードリクエストに返されるファイルタイプの制限（cos:response-content-type）

リクエストパラメータresponse-content-type

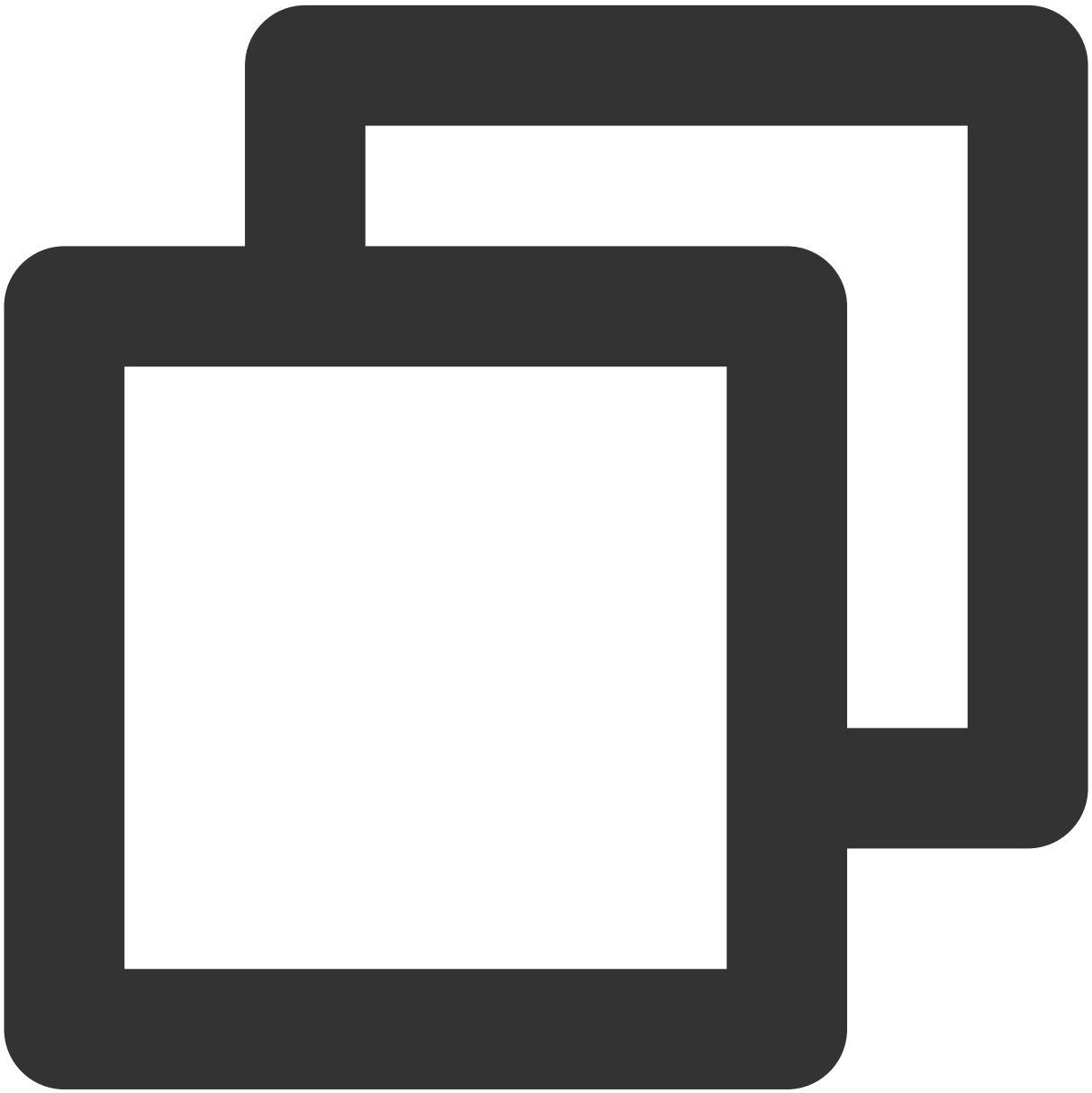
GetObjectインターフェースは、リクエストパラメータ `response-content-type` を追加し、レスポンスのContent-Typeヘッダーの値を設定するために用いることができます。

条件キーcos:response-content-type

条件キー `cos:response-content-type` を使用すると、リクエストにリクエストパラメータ `response-content-type` のパラメータ値を必ず含めるかどうかを制限することができます。

事例1: Get Objectのリクエストパラメータresponse-content-typeを必ず「image/jpeg」とするよう限定する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有している場合、以下のバケットポリシーの意味は、サブユーザー（uin:100000000002）のGet Objectリクエストに必ずリクエストパラメータresponse-content-typeを含め、かつリクエストパラメータの値を必ず「image/jpeg」とするよう制限するものです。 `response-content-type` はリクエストパラメータであるため、リクエストの送信時にurlencodeを経る必要があり、 `response-content-type=image%2Fjpeg` となります。そのため、Policyを設定する際は、「image/jpeg」にもurlencodeを行って「image%2Fjpeg」と入力する必要があります。



```
{
    "version":"2.0",
    "statement":[
```

```
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:GetObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_equal":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:GetObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_not_equal_if_exist":{
                    "cos:response-content-type":"image%2Fjpeg"
                }
            }
        }
    ]
}
```

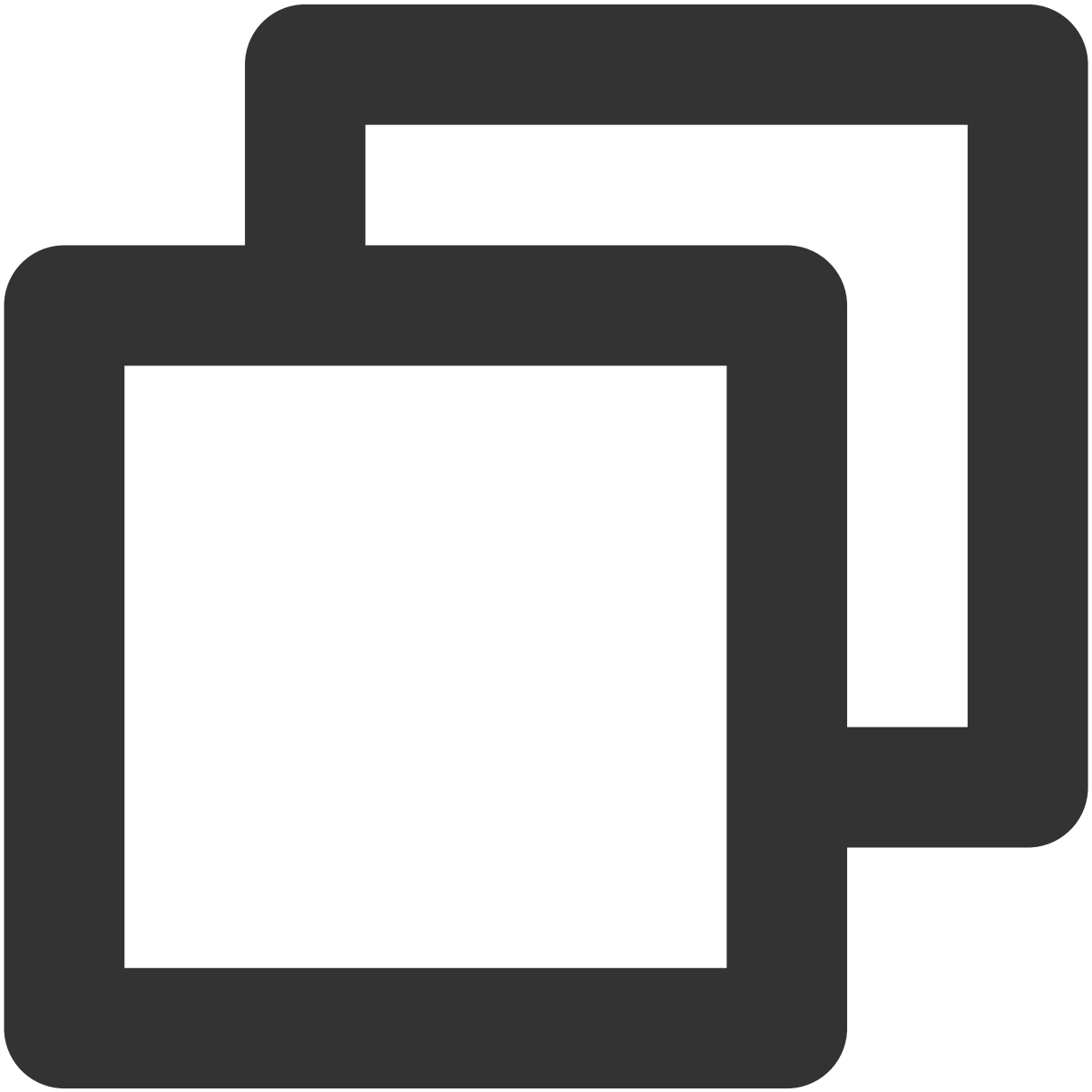
HTTPSプロトコルを使用したリクエストのみ承認する（cos:secure-transport）

条件キーcos:secure-transport

条件キー `cos:secure-transport` を使用して、リクエストが必ずHTTPSプロトコルを使用するよう制限することができます。

事例1：ダウンロードリクエストにHTTPSプロトコルの使用を必須とする

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有している場合、以下のバケットポリシーの意味は、サブユーザー（uin:100000000002）からのHTTPSプロトコルを使用したGetObjectリクエストに対してのみ権限承認を行うことを表します。

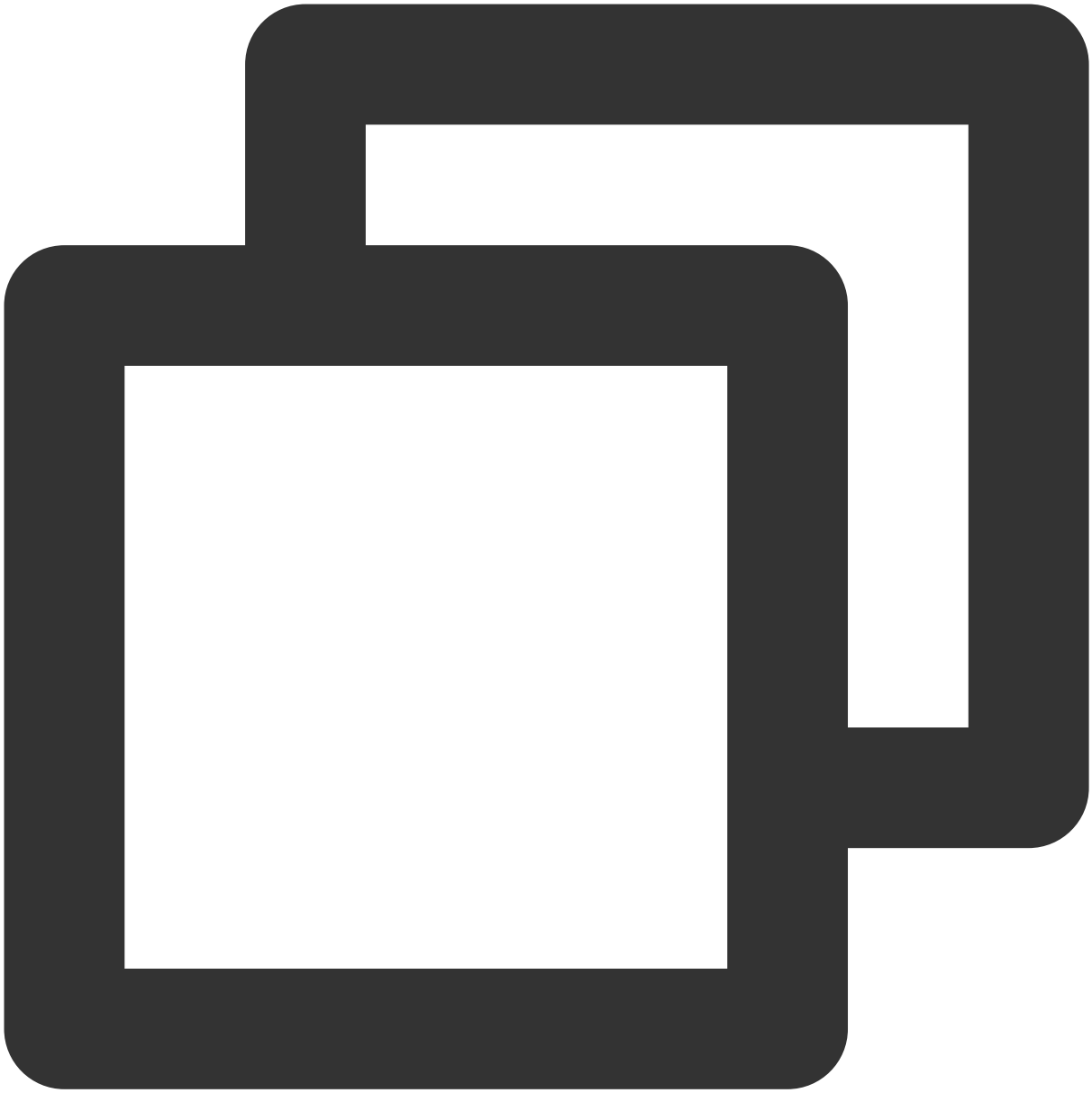


```
{
    "version":"2.0",
```

```
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:GetObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "bool_equal":{
                    "cos:secure-transport":"true"
                }
            }
        }
    ]
}
```

事例2：HTTPSプロトコルを使用していないあらゆるリクエストを拒否する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有している場合、以下のバケットポリシーの意味は、サブユーザー（uin:100000000002）からの、HTTPSプロトコルを使用していないあらゆるリクエストを拒否することを表します。



```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
```

```
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "bool_equal":{
                    "cos:secure-transport":"false"
                }
            }
        }
    ]
}
```

指定されたストレージタイプの設定のみを許可（cos:x-cos-storage-class）

リクエストヘッダーx-cos-storage-class

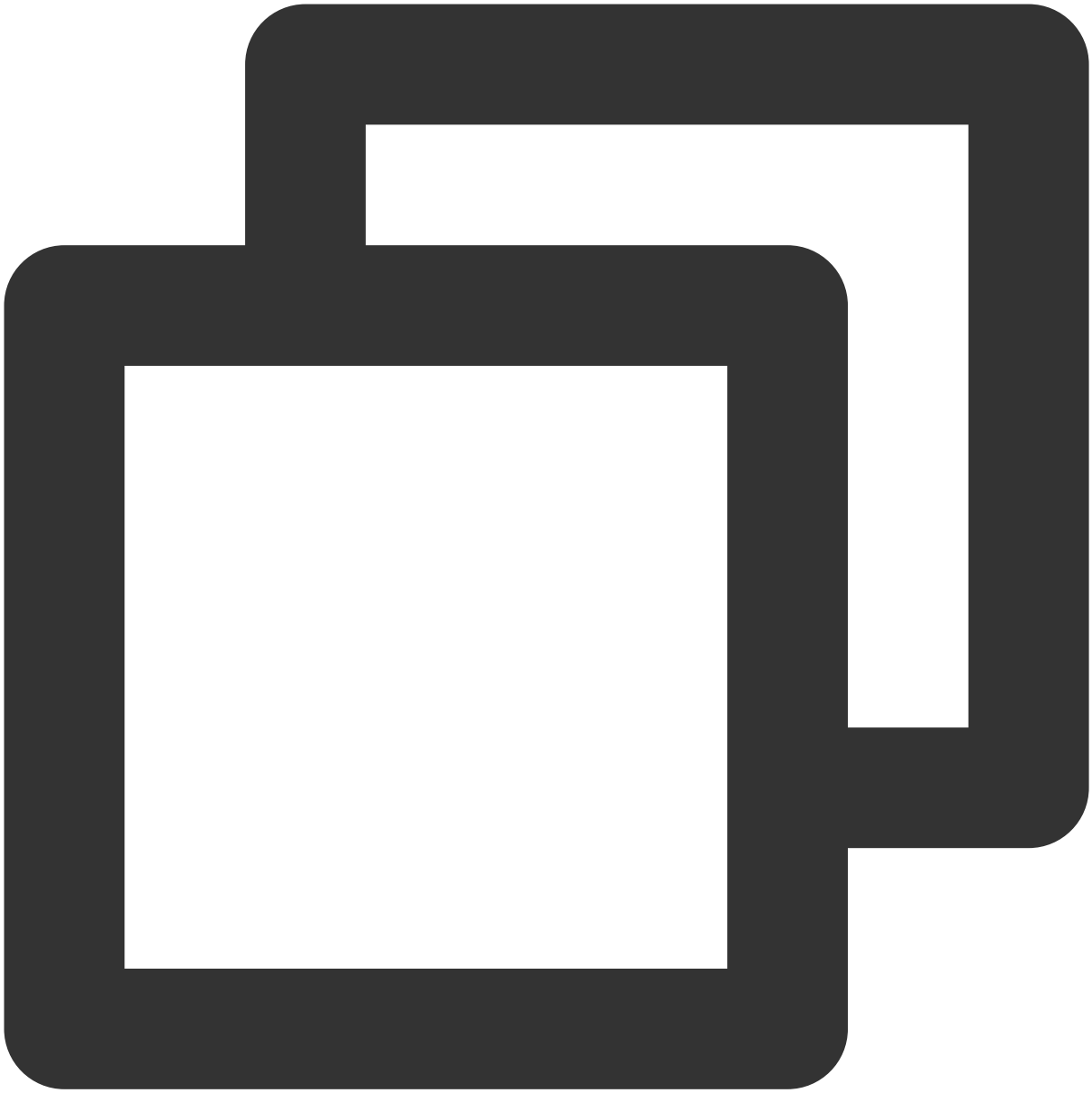
ユーザーはリクエストヘッダー `x-cos-storage-class` によって、オブジェクトをアップロードする際にストレージタイプを指定したり、オブジェクトのストレージタイプを変更したりすることができます。

条件キーcos:x-cos-storage-class

条件キー `cos:x-cos-storage-class` によって、リクエストヘッダー `x-cos-storage-class` を制限し、それによりストレージタイプを変更する可能性のあるリクエストを制限することができます。
COSのストレージタイプフィールドには、 `STANDARD` 、 `MAZ_STANDARD` , `STANDARD_IA` 、 `MAZ_STANDARD_IA` 、 `INTELLIGENT_TIERING` 、 `MAZ_INTELLIGENT_TIERING` 、 `ARCHIVE` 、 `DEEP_ARCHIVE` があります。

事例1：PutObjectの際にストレージタイプを必ず標準タイプに設定するよう要求する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有している場合、バケットポリシーによって、サブユーザー（uin:100000000002）からのPutObjectリクエストには必ずx-cos-storage-classヘッダーを含めなければならず、かつヘッダー値は `STANDARD` とするよう制限します。



```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
```

```
                "name/cos:PutObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_equal":{
                    "cos:x-cos-storage-class":"STANDARD"
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:PutObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_not_equal_if_exist":{
                    "cos:x-cos-storage-class":"STANDARD"
                }
            }
        }
    ]
}
```

指定されたバケット/オブジェクトACLの設定のみを許可（cos:x-cos-acl）

リクエストヘッダーx-cos-acl

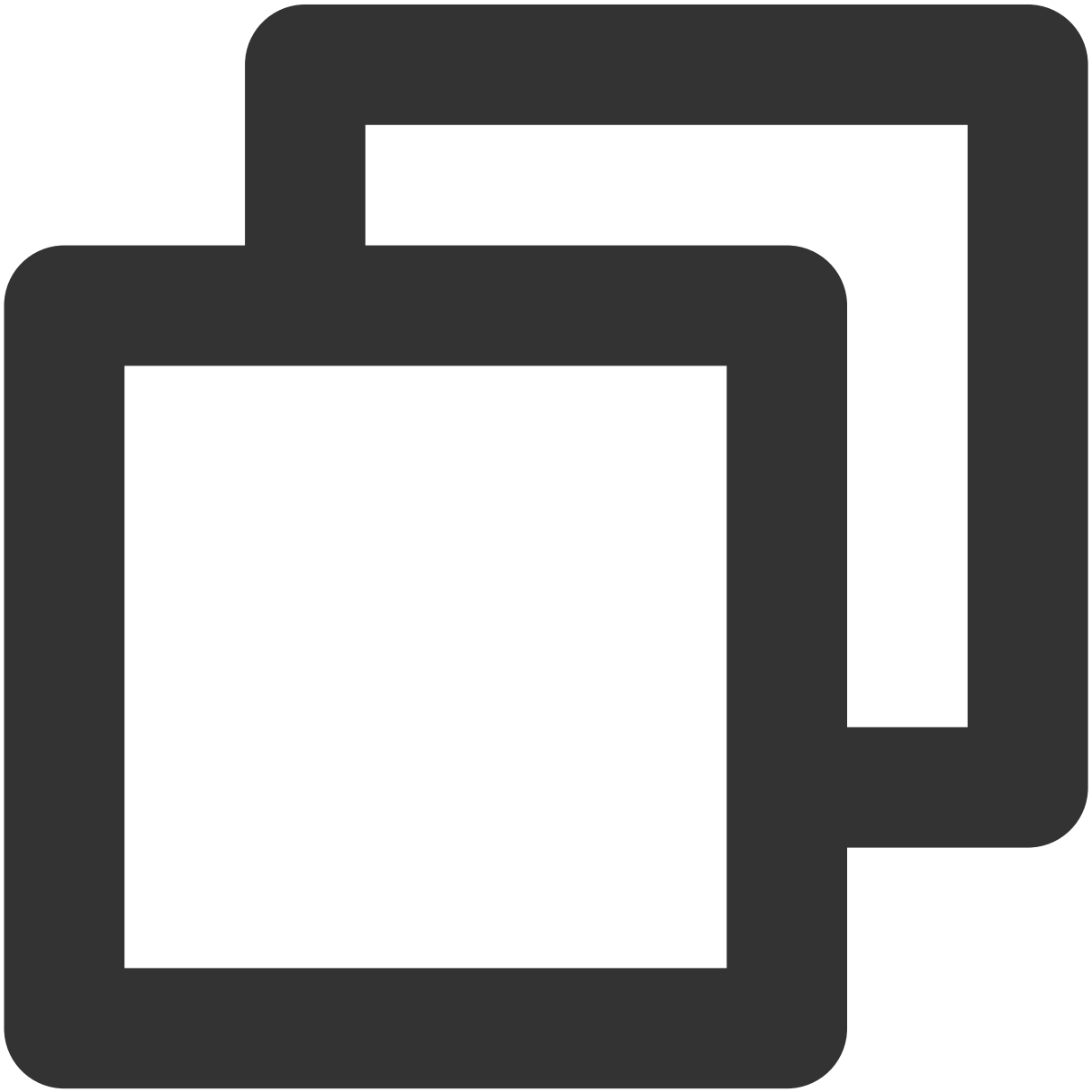
リクエストヘッダー `x-cos-acl` を使用すると、オブジェクトのアップロード、バケットの作成時にアクセス制御リスト（ACL）を指定するか、またはオブジェクト、バケットACLを変更することができます。ACLに関する説明については、ACLの概要をご参照ください。
バケットのプリセットACL： `private` 、 `public-read` 、 `public-read-write` 、 `authenticated-read` 。
オブジェクトのプリセットACL： `default` 、 `private` 、 `public-read` 、 `authenticated-read` 、 `bucket-owner-read` 、 `bucket-owner-full-control` 。

条件キーcos:x-cos-acl

条件キー `cos:x-cos-acl` によって、リクエストヘッダー `x-cos-acl` を制限し、それによりオブジェクトまたはバケットACLを変更する可能性のあるリクエストを制限することができます。

事例1：PutObjectの際に必ず同時にオブジェクトのACLをプライベートに設定する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有し、サブユーザー（uin:100000000002）にプライベートオブジェクトのみをアップロードできるよう制限する必要があるとします。以下のポリシーによって、PutObjectリクエストの送信時に必ずx-cos-aclヘッダーを含め、かつヘッダー値を `private` とするよう要求します。



```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:PutObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_equal":{
                    "cos:x-cos-acl":"private"
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:PutObject"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_not_equal_if_exist":{
                    "cos:x-cos-acl":"private"
                }
            }
        }
    ]
}
```

指定ディレクトリ下のオブジェクトのリストアップのみを許可する（cos:prefix）

条件キーcos:prefix

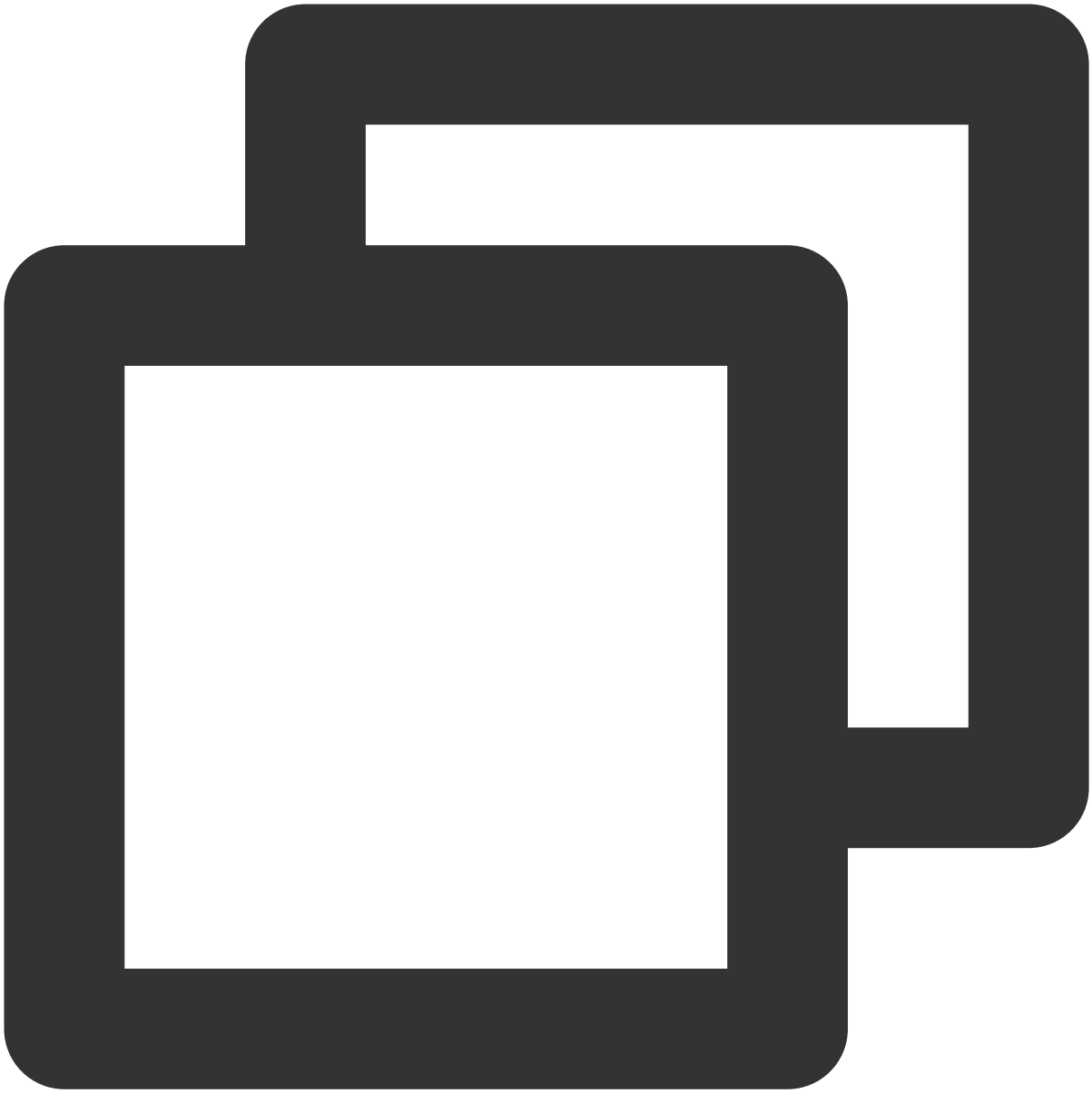
条件キー `cos:prefix` によってリクエストパラメータprefixを制限することができます。

注意：

prefixの値が特殊文字（中国語、 `/` など）の場合は、バケットポリシーに書き込む前にurlencodeを経る必要があります。

事例1：バケットの指定ディレクトリ下のオブジェクトのリストアップのみを許可する

ルートアカウント（uin:100000000001）がバケットexamplebucket-1250000000を所有し、サブユーザー（uin:100000000002）にバケットのfolder1ディレクトリ下のオブジェクトのみをリストアップできるよう制限する必要があるとします。以下のバケットポリシーによって、サブユーザーが送信するGetBucketリクエストには必ずprefixパラメータを含め、かつその値を“folder1”とするよう規定します。



```
{
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
                "name/cos:GetBucket"
```

```
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_equal":{
                    "cos:prefix":"folder1"
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "name/cos:GetBucket"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "string_equal_if_exist":{
                    "cos:prefix":"folder1"
                }
            }
        }
    ],
    "version":"2.0"
}
```

指定されたバージョンのTLSプロトコルの使用のみを許可（cos:tls-version）

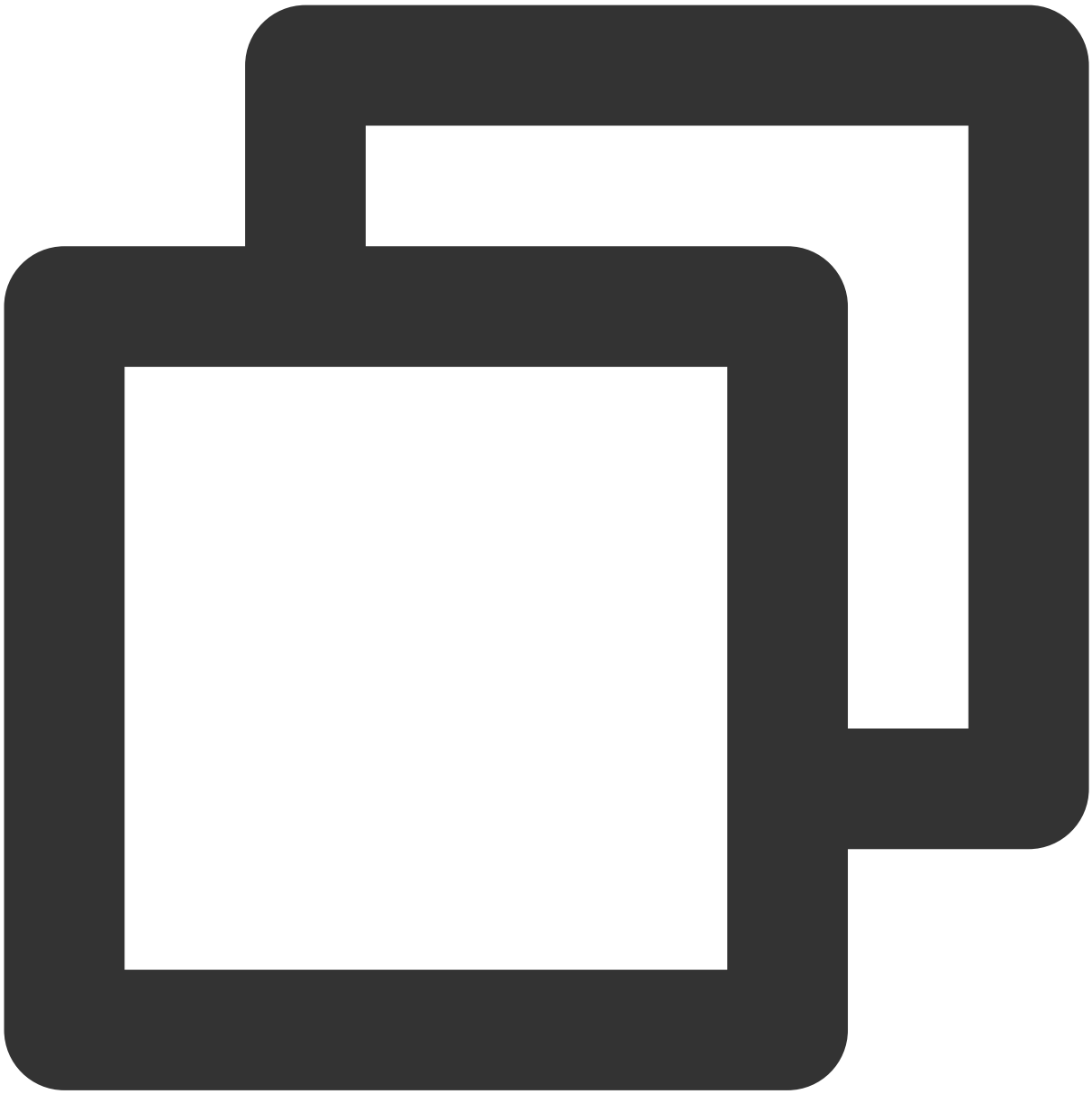
条件キー cos:tls-version

条件キー `cos:tls-version` によってHTTPSリクエストのTLSバージョンを制限することができます。この条件キーはNumricタイプであり、1.0、1.1、1.2などの浮動小数点数の入力が可能です。

事例1：TLSプロトコルのバージョンが1.2であるHTTPSリクエストにのみ権限を承認する

リクエストのシナリオ	予測
HTTPSリクエスト、TLSバージョンは1.0	403、失敗
HTTPSリクエスト、TLSバージョンは1.2	200、成功

ポリシーの例は次のとおりです。



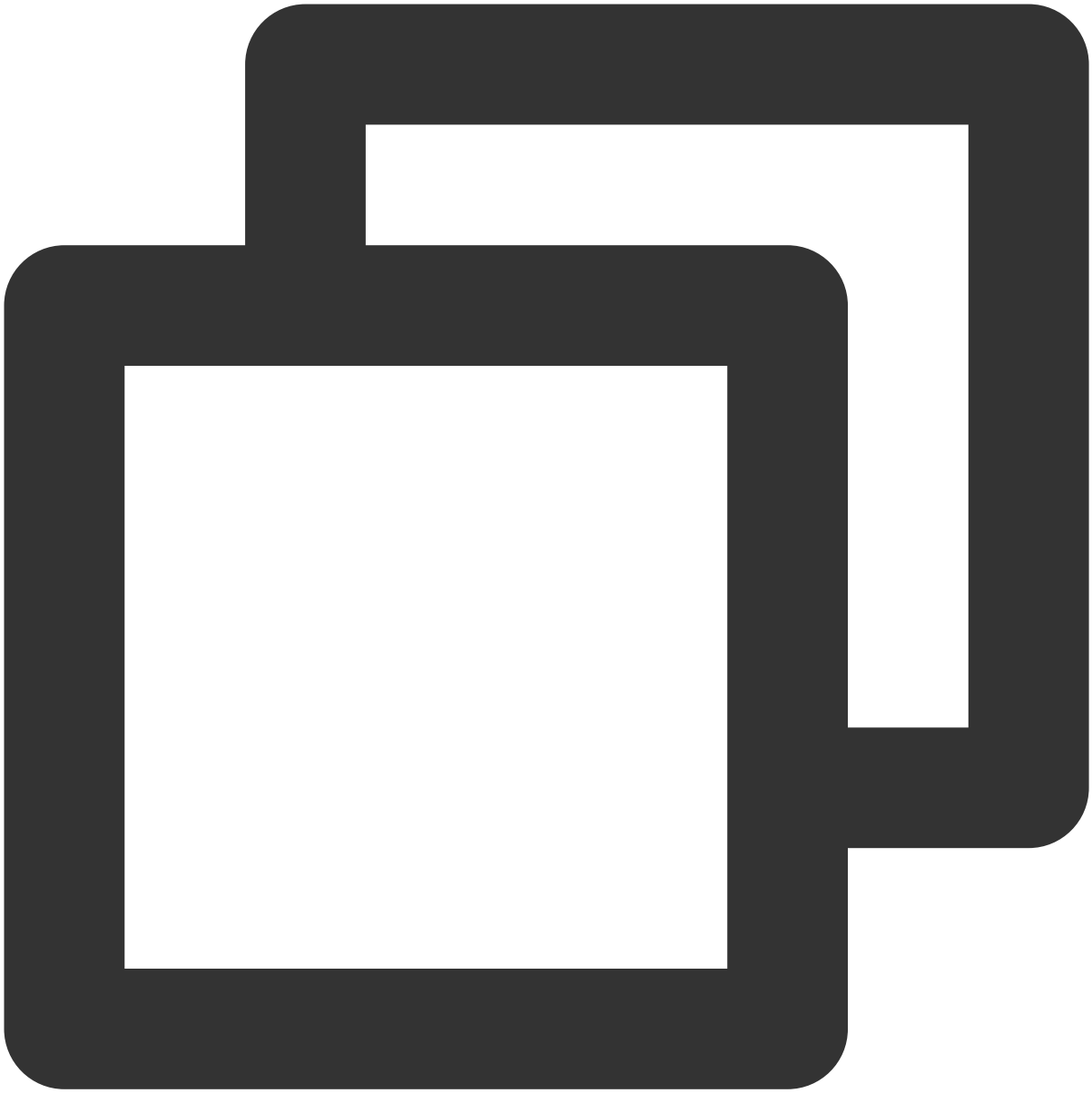
```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
```

```
            },
            "effect":"allow",
            "action":[
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "numeric_equal":{
                    "cos:tls-version":1.2
                }
            }
        }
    ]
}
```

事例2：TLSプロトコルのバージョンが1.2より低いHTTPSリクエストを拒否する

リクエストのシナリオ	予測
HTTPSリクエスト、TLSバージョンは1.0	403、失敗
HTTPSリクエスト、TLSバージョンは1.2	200、成功

ポリシーの例は次のとおりです。



```
{
    "version":"2.0",
    "statement":[
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"allow",
            "action":[
```

```
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "numeric_greater_than_equal":{
                    "cos:tls-version":1.2
                }
            }
        },
        {
            "principal":{
                "qcs":[
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect":"deny",
            "action":[
                "*"
            ],
            "resource":[
                "qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
            ],
            "condition":{
                "numeric_less_than_if_exist":{
                    "cos:tls-version":1.2
                }
            }
        }
    ]
}
```

バケット作成時に指定のバケットタグを強制的に設定（qcs:request_tag）

説明：

条件キーrequest_tagはPutBucket、PutBucketTagging操作にのみ適用可能です。GetService、PutObject、PutObjectTaggingなどの操作はこの条件キーをサポートしていません。

条件キー qcs:request_tag

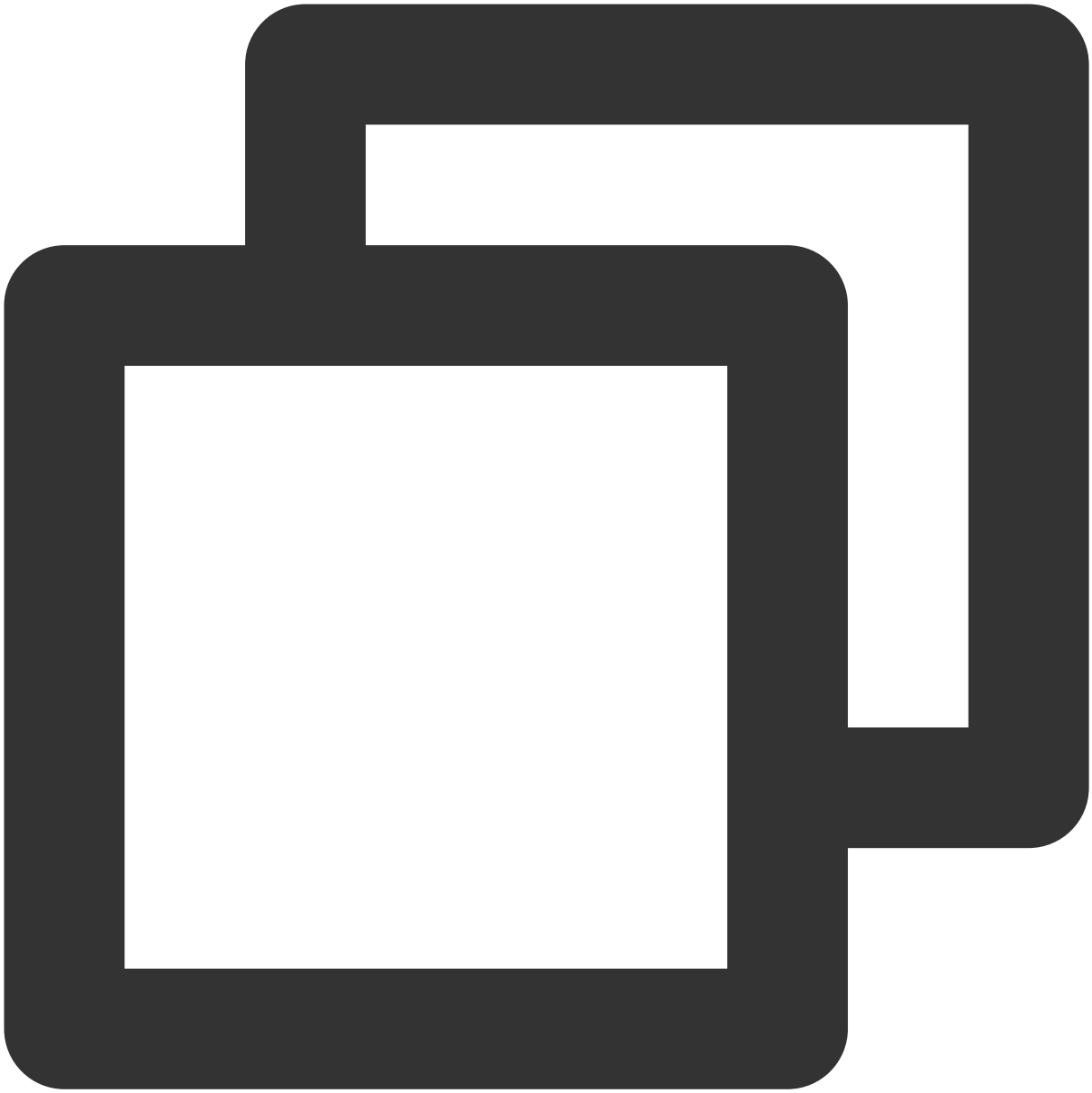
条件キー `qcs:request_tag` によって、ユーザーがPutBucket、PutBucketTaggingリクエストを送信する際に、必ず指定したバケットタグを含めるよう制限することができます。

事例：ユーザーのバケット作成時に必ず指定したバケットタグを含めるよう制限する

多くのユーザーはバケットタグによってバケットを管理しています。次のポリシーの例は、ユーザーがバケットを作成する際、指定のバケットタグ `<a,b>` および `<c,d>` を設定した場合にのみ権限を取得できるよう制限するものです。
バケットタグは複数設定することができ、バケットタグのキー値、タグの数が異なると、それらはすべて異なるセットとなります。ユーザーの持つ複数のパラメータ値をセットA、条件で規定する複数のパラメータ値をセットBと仮定します。この条件キーを使用する際、限定語for_any_value、for_all_valueの組み合わせによって異なる意味を表すことができます。
`for_any_value:string_equal` はAとBに共通部分が存在する場合に発効することを表します。
`for_any_value:string_equal` はAがBのサブセットである場合に発効することを表します。
`for_any_value:string_equal` を使用する場合、対応するポリシーとリクエストは次のように表されます。

リクエストのシナリオ	予測
PutBucket、リクエストヘッダー `x-cos-tagging: a=b&c=d`	200、成功
PutBucket、リクエストヘッダー `x-cos-tagging: a=b`	200、成功
PutBucket、リクエストヘッダー `x-cos-tagging: a=b&c=d&e=f`	200、成功

ポリシーの例は次のとおりです。



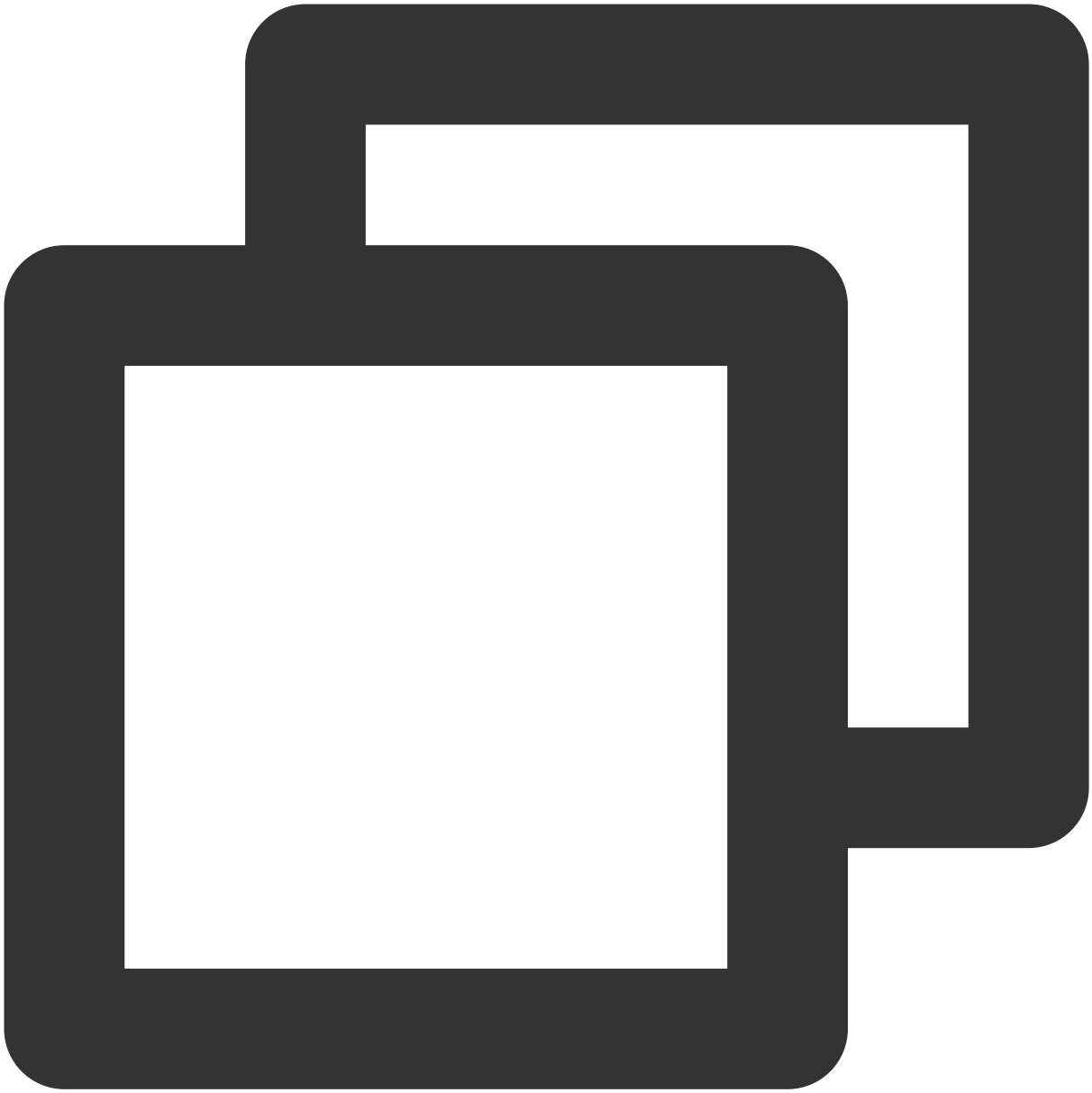
```
{
    "version": "2.0",
    "statement":[
        {
            "principal":{
                "qcs": [
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect": "allow",
            "action":[
```

```
                "name/cos:PutBucket"
            ],
            "resource": "*",
            "condition":{
                "for_any_value:string_equal":{
                    "qcs:request_tag": [
                        "a&b",
                        "c&d"
                    ]
                }
            }
        }
    ]
}
```

`for_all_value:string_equal` を使用する場合、対応するポリシーとリクエストは次のように表されます。

リクエストのシナリオ	予測
PutBucket、リクエストヘッダー `x-cos-tagging: a=b&c=d`	200、成功
PutBucket、リクエストヘッダー `x-cos-tagging: a=b`	200、成功
PutBucket、リクエストヘッダー `x-cos-tagging: a=b&c=d&e=f`	403、失敗

ポリシーの例は次のとおりです。



```
{
    "version": "2.0",
    "statement":[
        {
            "principal":{
                "qcs": [
                    "qcs::cam::uin/100000000001:uin/100000000002"
                ]
            },
            "effect": "allow",
            "action":[
```

```
                "name/cos:PutBucket"
            ],
            "resource": "*",
            "condition":{
                "for_all_value:string_equal": {
                    "qcs:request_tag": [
                        "a&b",
                        "c&d"
                    ]
                }
            }
        }
    ]
}
```

他のルートアカウント下のサブアカウントに対し、名前のバケットの操作権限を承認する

最終更新日：：2024-06-26 10:42:27

ユースケース

現在、ルートアカウントA（APPIDは1250000000）の名前の下にバケット `examplebucket1-1250000000` と `examplebucket2-1250000000` があり、これとは別にルートアカウントBとそのサブアカウントB0があります。B0は業務上の必要性により、アカウントA下の2つのバケットを操作したいとします。その場合の関連の権限承認操作の方法について以下でご説明します。

操作手順

ルートアカウントBにAの名前のバケットの操作権限を承認する

1. ルートアカウントAを使用してCOSコンソールにログインします。
2. **バケットリスト**をクリックし、権限を承認したいバケットを見つけ、その名前をクリックしてバケット詳細ページに進みます。
3. 左側ナビゲーションバーで**権限管理**をクリックし、そのバケットの権限管理ページに進みます。
4. **Policy権限設定**の設定項目を見つけ、**ポリシーの追加**をクリックし、カスタムポリシーを選択して**次のステップ**をクリックします。
5. 次のフォーマットの項目を入力します。
ポリシーID：未入力でもかまいません。
エフェクト：許可。
ユーザー：ユーザーをクリックして追加します。ユーザータイプはルートアカウントを選択し、アカウントIDにはルートアカウントBのUIN（例：100000000002）を入力します。
リソース：必要に応じて選択します。デフォルトではバケット全体です。
リソースパス：リソースを指定する場合のみ入力が必要です。必要に応じて入力します。
アクション：クリックしてアクションを追加し、すべてのアクションを選択します。一部のアクションにのみ権限を承認したい場合は、実際に必要なアクションを1つまたは複数選択することもできます。
条件：必要に応じて入力します。不要な場合は空にしておくことができます。
6. **完了**をクリックします。この時点でルートアカウントBはこのバケットの操作に関するすべての権限を取得しています。
7. 他のバケットへの権限承認を行いたい場合は、上記の手順を繰り返してください。

サブアカウントB0にAの名前のバケットの操作権限を承認する

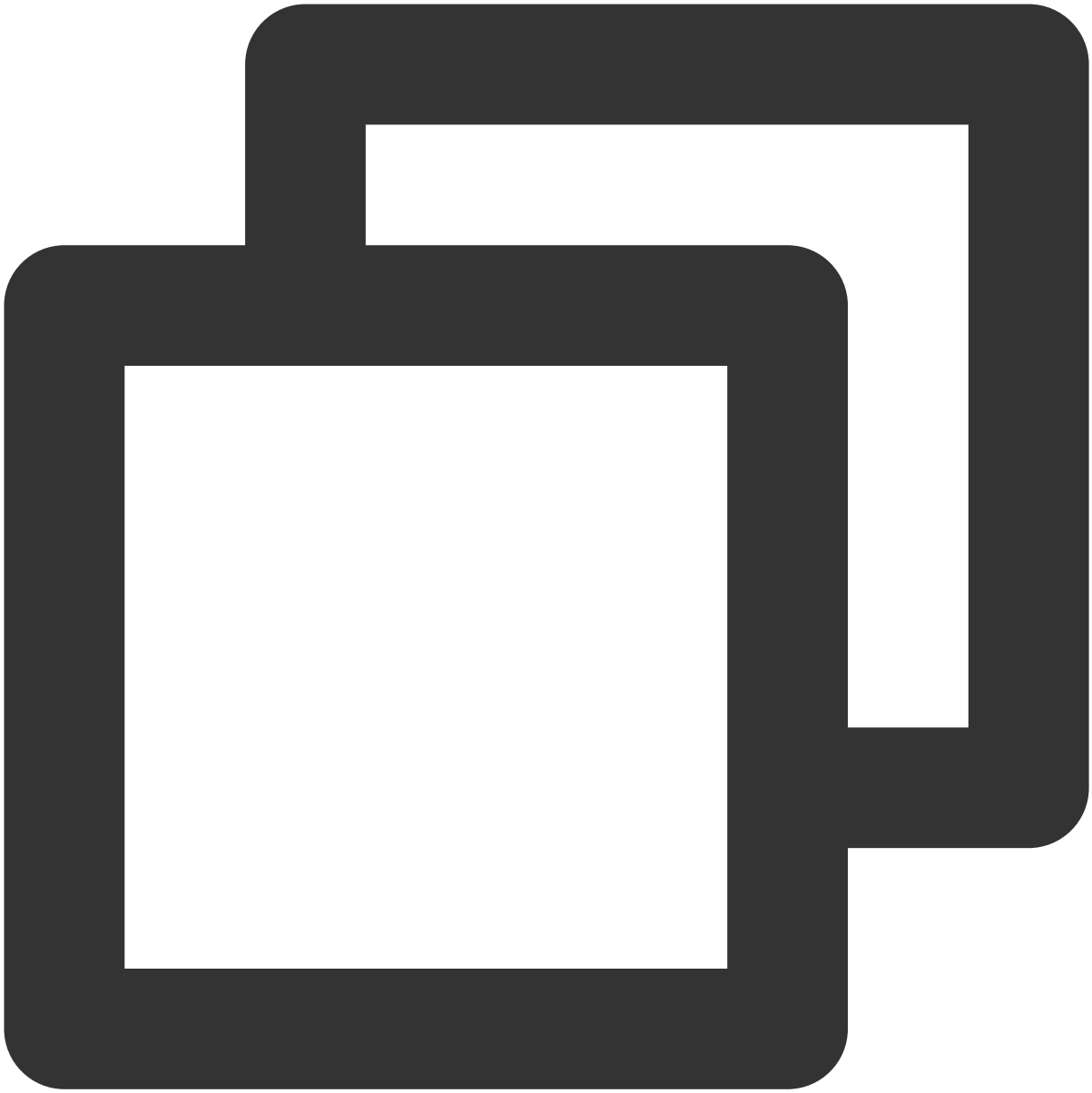
1. ルートアカウントBを使用してCloud Access Management（CAM）コンソールのポリシーページにログインします。
2. **カスタムポリシーの新規作成 > ポリシー構文で作成**を選択し、続いて空白テンプレートを選択し、**次のステップ**をクリックします。

説明：

ルートアカウントBがサブアカウントB0に権限承認を行う場合は、カスタムポリシーによる権限承認のみ可能であり、プリセットポリシーによる権限承認は行えません。

3. 次のフォーマットに入力します。

ポリシー名：重複せず、かつ意味のあるポリシー名（例：cos-child-account）をご自身で定義します。
備考：オプションです。ご自身で入力します。
ポリシー内容：



```
{
 "version": "2.0",
 "statement": [
  {
      "action": "cos:*",
      "effect": "allow",
            "resource": [
                "qcs::cos::uid/1250000000:examplebucket1-1250000000/*",
                "qcs::cos::uid/1250000000:examplebucket2-1250000000/*"
            ]
  }
```

```
  ]
}
```

上記のポリシーはメインアカウントBが操作権限を持っているAにおけるストレージバケットをすべてサブアカウントB0に許可することを示します。その中、 `uid/1250000000` の中の1250000000はメインアカウントのAのAPPIDであり、 `examplebucket1-1250000000` と `examplebucket2-1250000000` はメインアカウントBが操作権限を持っているAにおけるストレージバケットです。

4. **完了**をクリックすると、ポリシーの作成が完了します。
5. **ポリシー**リストで先ほど作成したポリシーを見つけ、右側の**ユーザー/グループ/ロールのバインド**をクリックします。
6. ポップアップウィンドウで、サブアカウントB0にチェックを入れ、**OK**をクリックします。
7. 権限承認操作が完了し、サブアカウントB0のキーを使用して、Aの名前のバケットのテスト操作を行うことができるようになります。

パフォーマンスの最適化
リクエスト速度とパフォーマンスの最適化

最終更新日：：2024-06-26 10:42:27

注意：
現在COSでは、下層インデックス分散メカニズムによって高QPSを実現しています。よりハイパフォーマンスなQPSが必要な場合は、お問い合わせください。日常的なファイルの取り扱いでは過度に集中するインデックスストレージ方式を使用せず、このドキュメントに沿って行うことを推奨します。

概要

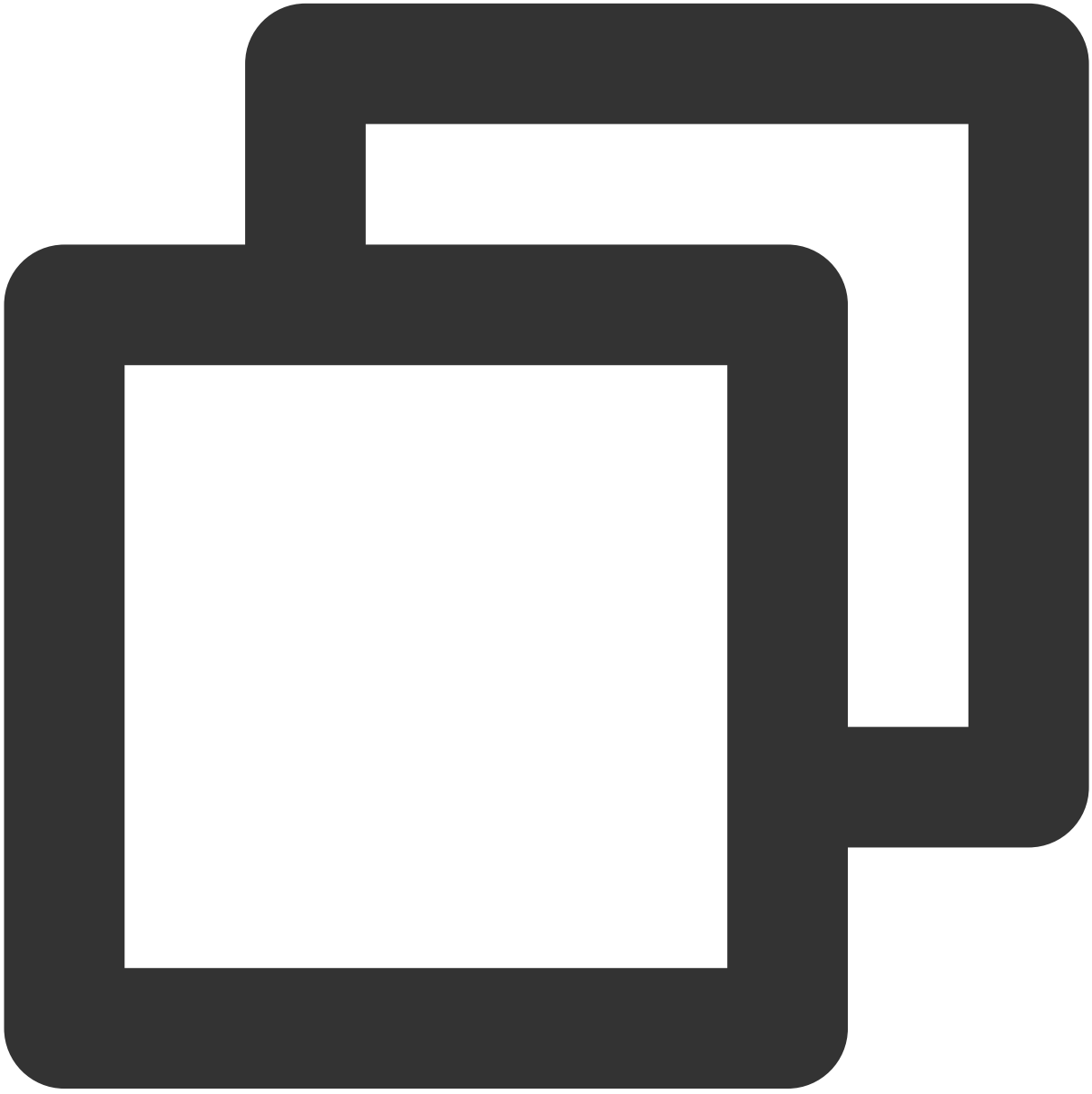
ここでは、リクエスト速度パフォーマンスの最適化のTencent Cloud Object Storage（COS）上でのベストプラクティスについて検討します。
Tencent Cloud COSがご提供する代表的なワークロード能力は、PUTリクエストで1秒あたり30000、またはGETリクエストで1秒あたり30000です。ワークロードが上記の能力を超える場合は、このガイドに従ってリクエスト速度パフォーマンスの拡張と最適化を行うことをお勧めします。
説明：
リクエストロードとは1秒間に送信されるリクエスト数のことであり、同時接続数ではありません。すなわち、数千の接続数を維持すると同時に、1秒間に数百の新規接続リクエストを送信できることを意味します。
Tencent Cloud COSはパフォーマンスの拡張によるリクエスト速度の向上をサポートしています。リクエストのうち、GETリクエストの負荷が高い場合は、Tencent Cloud CDN製品を併用することをお勧めします。詳細については、ドメイン名管理をご参照ください。バケットの総合リクエスト速度が1秒あたり30000 PUT/LIST/DELETEリクエストを超えることが見込まれる場合は、ワークロードへの対応を準備し、リクエスト制限を回避できるよう、お問い合わせいただくことをお勧めします。
説明：
混合リクエストロードが偶発的に1秒あたり30000を超えることがあるだけで、突発的に1秒あたり30000を超えることがない場合は、このガイドに従わなくても構いません。

実践手順

混合リクエストロード

大量のオブジェクトをアップロードする必要がある場合は、選択するオブジェクトキーによってパフォーマンスの問題が起こる可能性があります。Tencent Cloud COSのObjectキー値のストレージメソッドについて、以下で簡単にご説明します。

Tencent Cloud COSの各サービスリージョンでは、バケット（Bucket）とオブジェクト（Object）のキー値をインデックスとして保守しています。オブジェクトキーはUTF-8バイナリーシーケンスに従ってインデックスの複数のパーティション内に保存されます。タイムスタンプの使用やアルファベット順など、大量のキー値がある場合、キー値の存在するパーティションの読み取り書き込みパフォーマンスが上限に達する可能性があります。バケットパス `examplebucket-1250000000.cos.ap-beijing.myqcloud.com` の例では、以下の例のような場合にインデックスパフォーマンスが上限に達する可能性があります。



```
20170701/log120000.tar.gz
20170701/log120500.tar.gz
20170701/log121000.tar.gz
```

```
20170701/log121500.tar.gz
...
image001/indexpage1.jpg
image002/indexpage2.jpg
image003/indexpage3.jpg
...
```

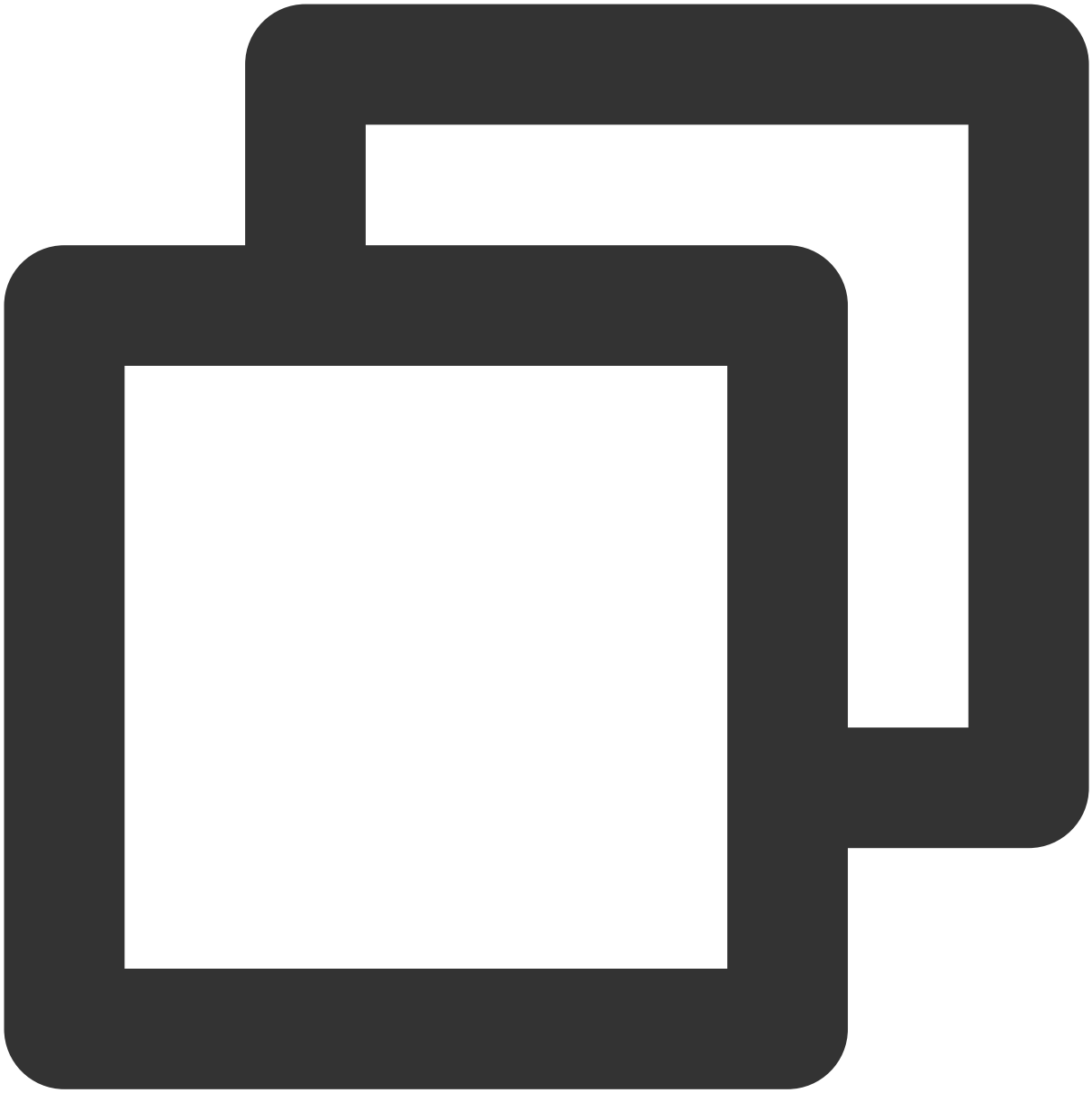
業務の代表的なロードが1秒あたり30000リクエストを超えている場合は、上記の例のようなシーケンシャルなキー値の使用を避ける必要があります。業務上、シーケンシャルな番号または日付・時刻などの文字をオブジェクトキーとしなければならない場合は、いくつかのメソッドを使用してオブジェクトキー名にランダムなプレフィックスを追加することで、複数のインデックスパーティションでのキー値管理を実現し、ロード集中時のパフォーマンスを向上させることができます。キー値のランダム性を上げるためのいくつかのメソッドを以下に記載します。

注意：

以下に記載するメソッドはすべて、単一のバケットへのアクセスパフォーマンスを向上させる可能性があるメソッドです。業務の代表的なロードが1秒あたり30000リクエストを超えている場合は、以下のメソッドを実行すると同時に、ワークロードへの対応を事前に準備できるよう、お問い合わせをいただく必要があります。

16進数のハッシュプレフィックスを追加する

オブジェクトキーのランダム性を最も直接的に向上させる方法です。オブジェクトキー名の一番前にハッシュ文字列をプレフィックスとして追加します。例えばオブジェクトをアップロードする際、パスのキー値に対しSHA1またはMD5ハッシュ計算を行い、数桁の文字を選択して、プレフィックスとしてキー値名に追加することができます。通常は2～4桁の文字ハッシュプレフィックスがあれば十分です。



```
faf1-20170701/log120000.tar.gz
e073-20170701/log120500.tar.gz
333c-20170701/log121000.tar.gz
2c32-20170701/log121500.tar.gz
```

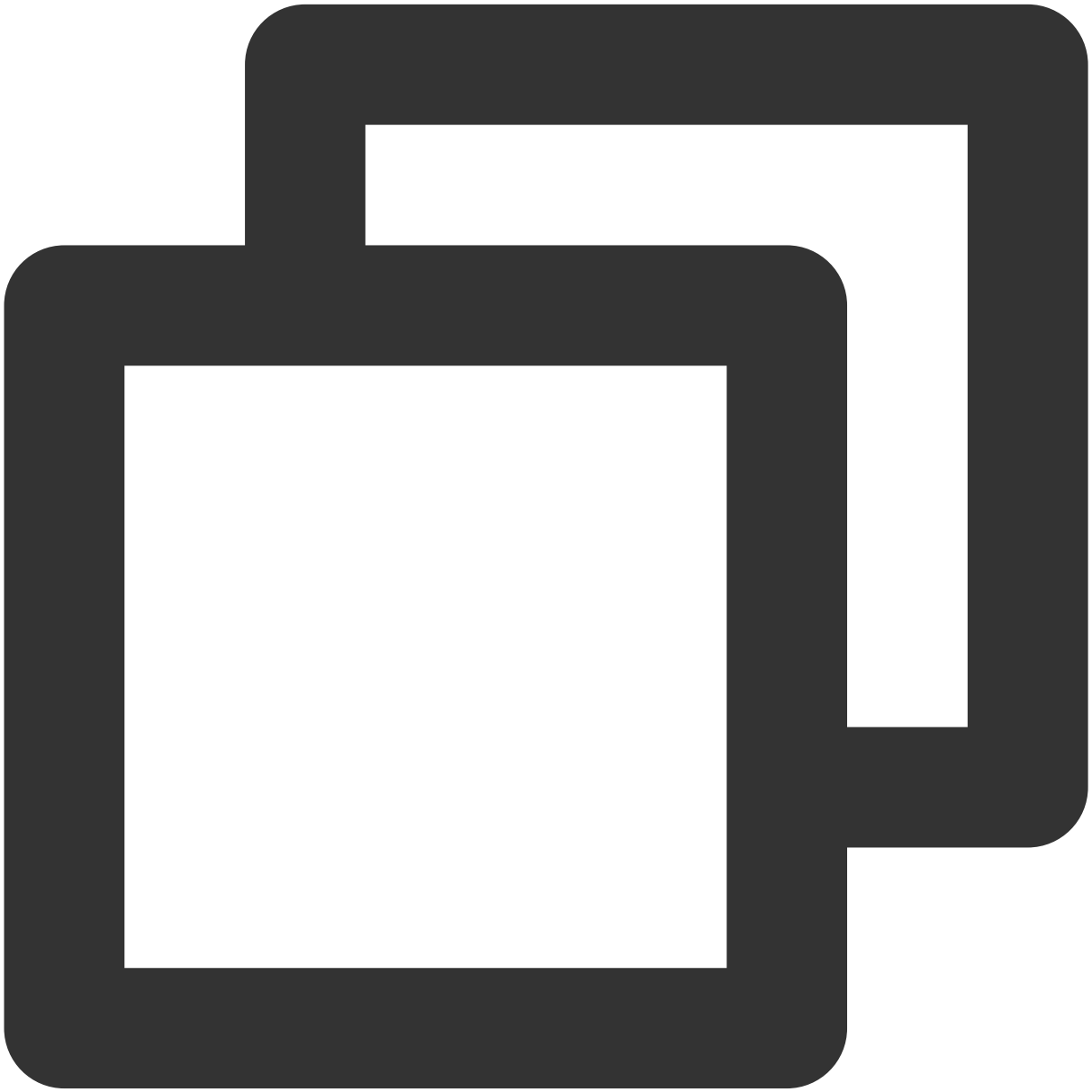
注意：

Tencent Cloud COSのキー値インデックスはUTF-8バイナリーシーケンスをインデックスとしているため、オブジェクトのリストアップ（GET Bucket）操作を実行する際、元の完全な20170701プレフィックス構造を得るため

に65536回のオブジェクトリストアップ操作が必要となる可能性があります。

列挙値プレフィックスを追加する

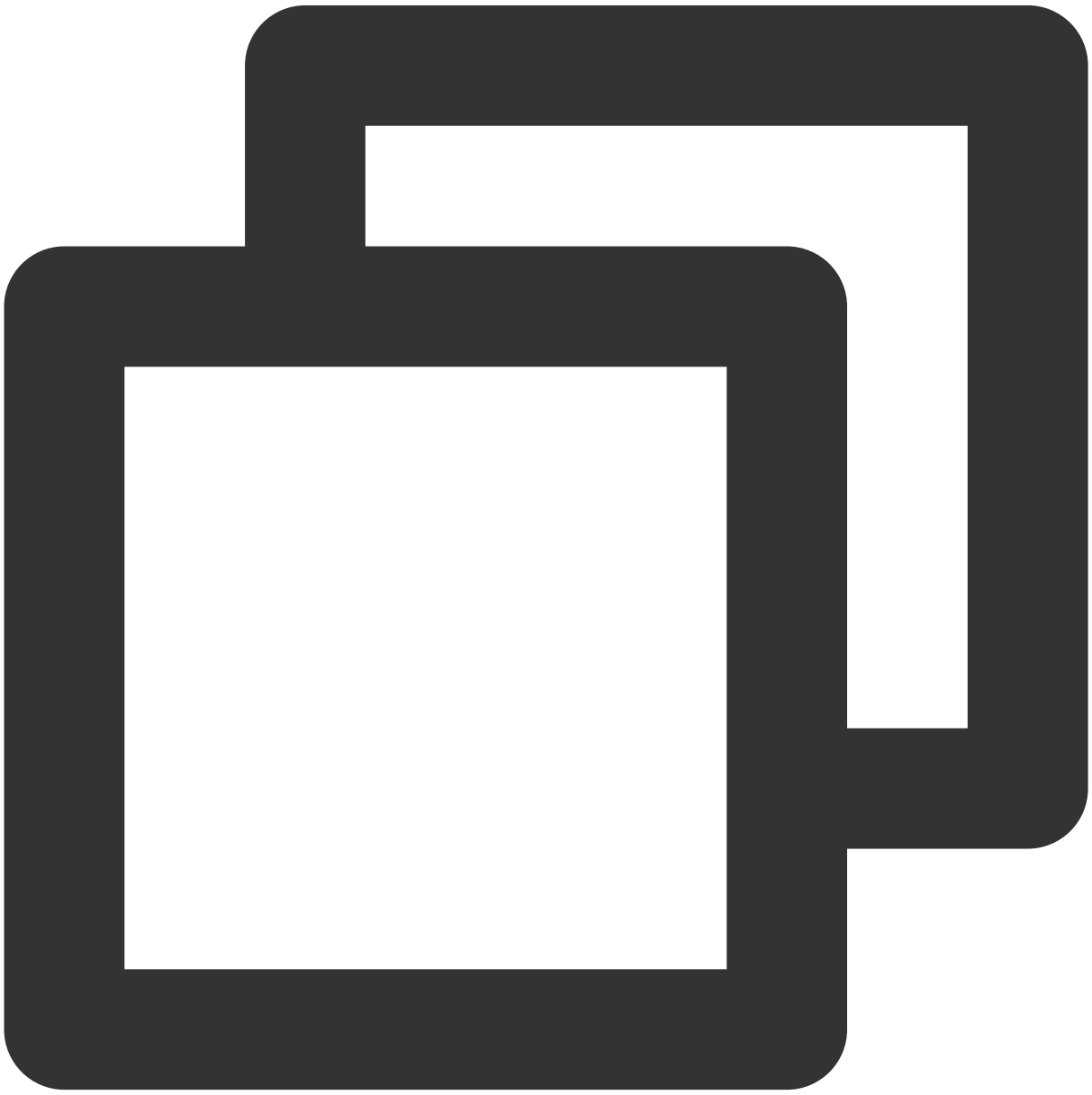
オブジェクトキーの検索のしやすさを維持したい場合は、ファイルタイプについていくつかのプレフィックスを列挙してオブジェクトのグループ化を行い、同一の列挙値のプレフィックスが、存在するインデックスパーティションのパフォーマンスを共有できるようにすることができます。



```
logs/20170701/log120000.tar.gz
logs/20170701/log120500.tar.gz
logs/20170701/log121000.tar.gz
```

```
...
images/image001/indexpage1.jpg
images/image002/indexpage2.jpg
images/image003/indexpage3.jpg
...
```

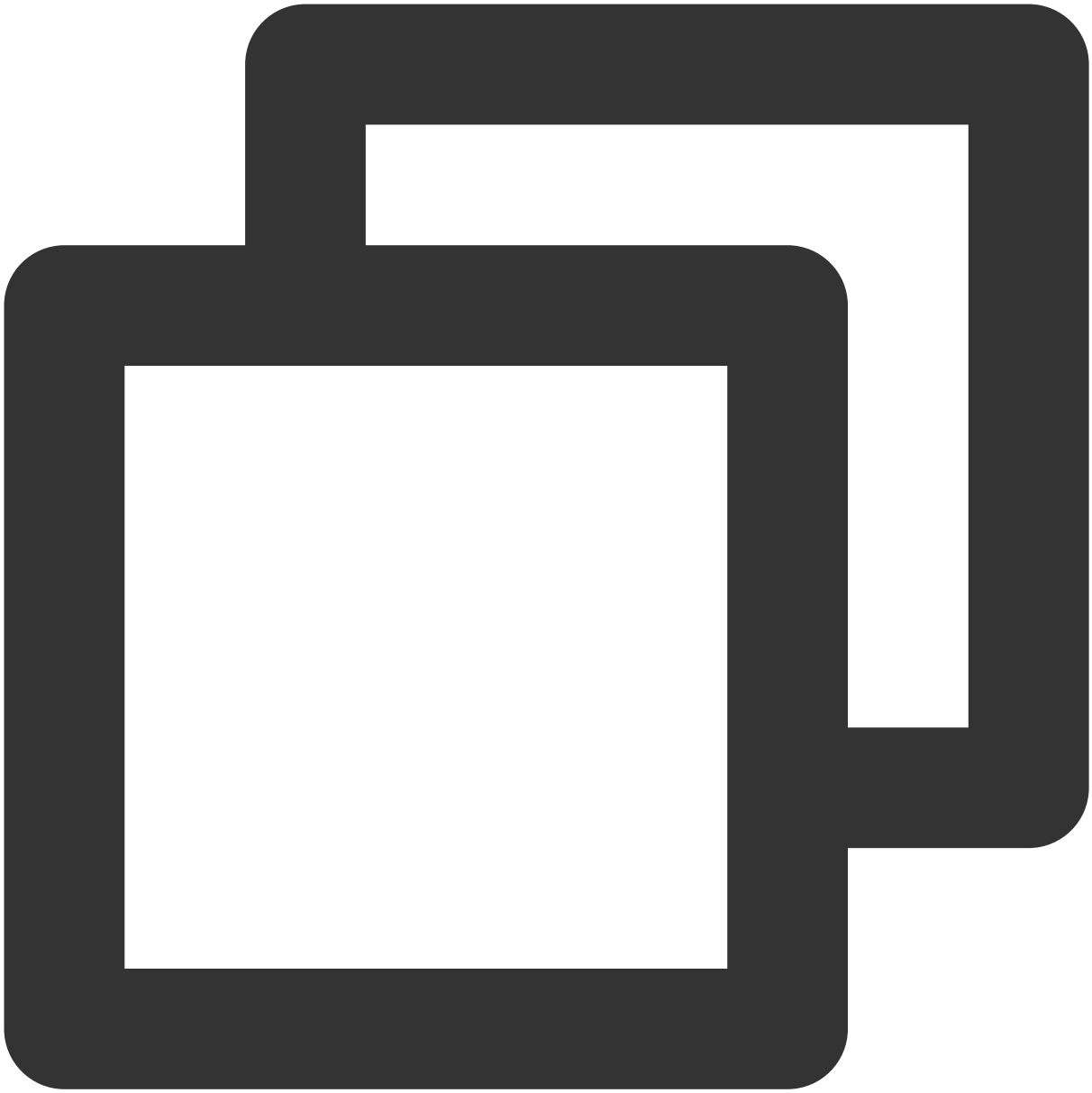
同一の列挙プレフィックスにおいて、依然として高いアクセスロードがあり、1秒あたり30000リクエストを継続的に超える場合は、上記の16進数のハッシュプレフィックスを追加するメソッドを参照し、列挙値の後に続けてハッシュプレフィックスを追加し、複数のインデックスのパーティションを実行することで、より高い読み取り書き込みパフォーマンスを実現することができます。



```
logs/faf1
-20170701/log120000.tar.gzlogs/
e073-20170701/log120500.tar.gzlogs/
333c-20170701/log121000.tar.gz...images/
0165-image001/indexpage1.jpgimages/
a349-image002/indexpage2.jpgimages/
ac00-image003/indexpage3.jpg...
```

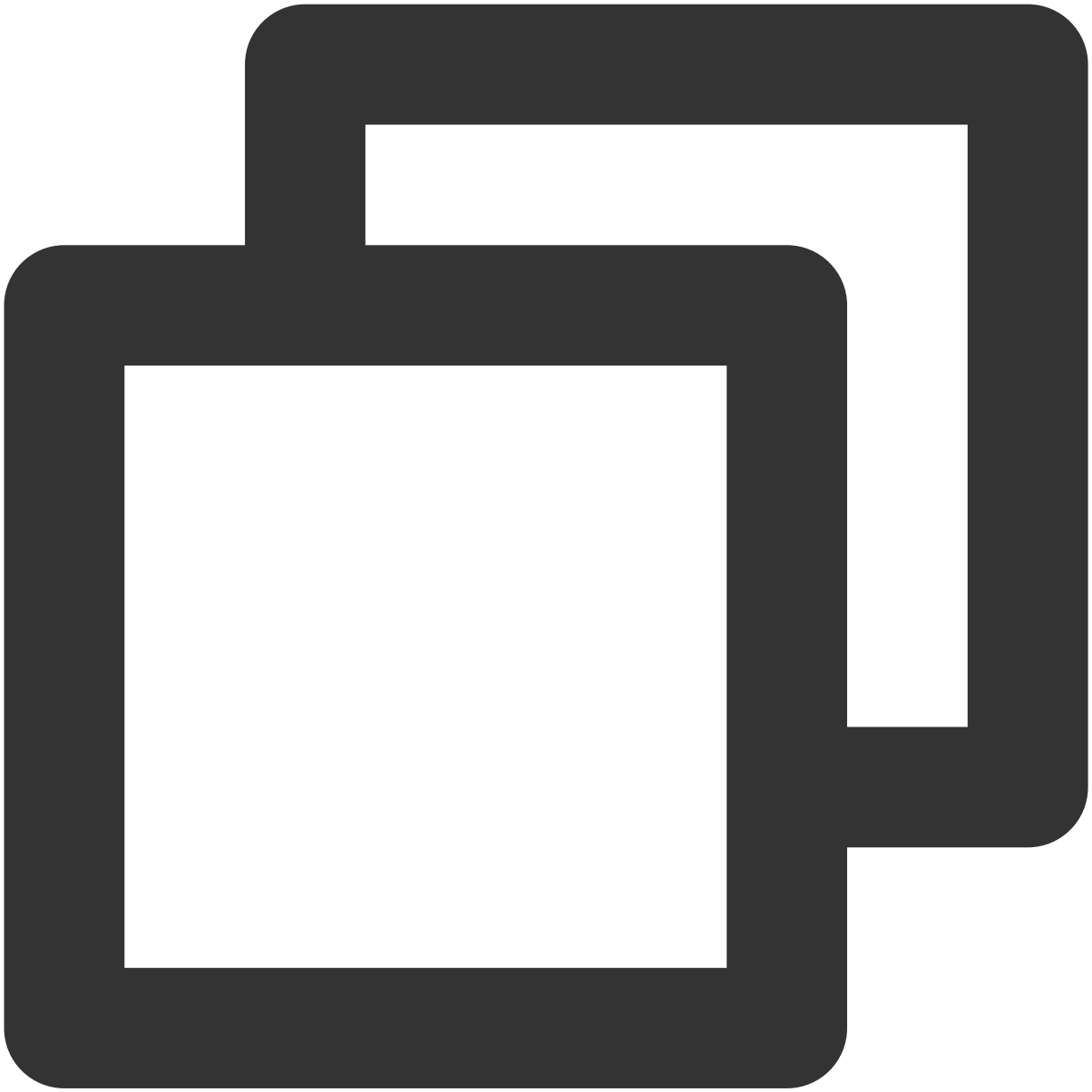
キー値名の文字列を反転する

業務上、インクリメントするIDまたは日付を使用せざるを得ない場合、または連続したプレフィックスのオブジェクトを一度に大量にアップロードする必要がある場合があります。代表的な使用法は次のようなものです。



```
20170701/log0701A.tar.gz
20170701/log0701B.tar.gz
20170702/log0702A.tar.gz
20170702/log0702B.tar.gz
...
id16777216/album/hongkong/img20170701121314.jpg
id16777216/music/artist/tony/anythinggoes.mp3
id16777217/video/record20170701121314.mov
id16777218/live/show/date/20170701121314.mp4
...
```

上記のようなキー値の命名方法では、2017とIDをプレフィックスとするキー値が存在するインデックスパーティションが容易に上限に達します。この場合、キー値のプレフィックスの一部を反転し、一定のランダム性を実現することができます。



```
10707102/log0701A.tar.gz
10707102/log0701B.tar.gz
20707102/log0702A.tar.gz
20707102/log0702B.tar.gz
...
61277761di/album/hongkong/img20170701121314.jpg
61277761di/music/artist/tony/anythinggoes.mp3
```

```
71277761di/video/record20170701121314.mov
81277761di/live/show/date/20170701121314.mp4
...
```

高GETリクエストロード

主な業務上の負荷がGETリクエスト（すなわちダウンロードリクエスト）である場合は、上記の原則に従うことに加えて、Tencent Cloud CDNサービスの併用もお勧めします。
Tencent Cloud CDNは全国および全世界に分布するエッジアクセラレーションノードを利用して、ユーザーへのコンテンツ配信時に遅延を低減して速度を上昇させるものです。ホットファイルに対するプリフェッチ方式によるキャッシュをサポートすることで、 COSへのback-to-origin GETリクエスト数を減少させます。詳細については、ドメイン名管理のドキュメントをご参照ください。

COS負荷テストガイド

最終更新日：：2024-06-26 10:42:27

COSBenchの紹介

COSBenchはIntel社が開発した、オブジェクトストレージ向けの負荷テストツールです。S3プロトコルと互換性を持つオブジェクトストレージとして、Tencent CloudのCloud Object Storage (COS)は、このツールを使用して、読み取り/書き込みの負荷テストを実施できます。

システム環境

実行環境はCentOS 7.0以降のバージョンを推奨します。ubuntu環境では、予期せぬ問題が発生することがあります。

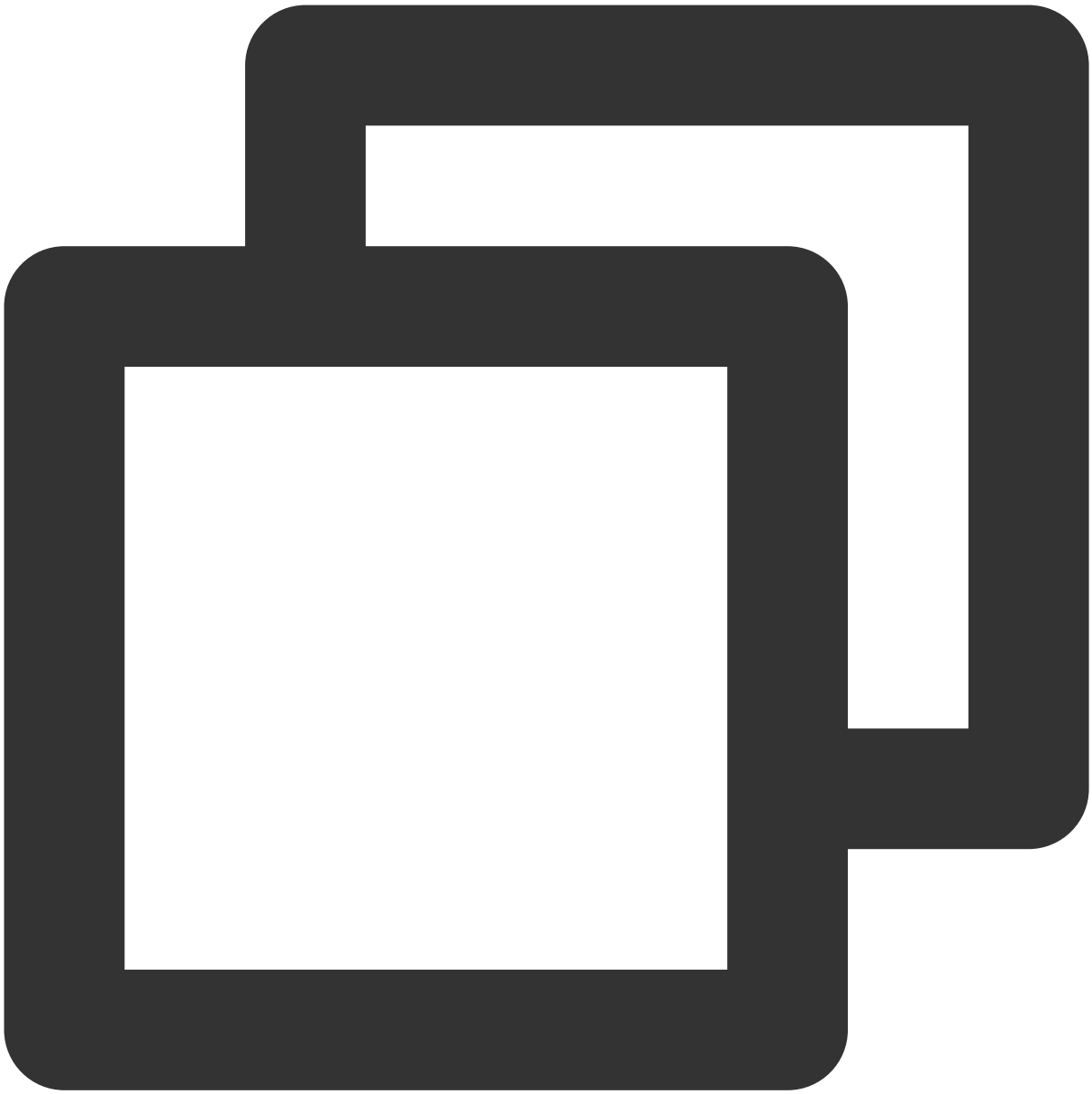
性能に影響する要因

コア数：コアが少なく、有効になっているworkerが多い場合、コンテキストの切替でオーバーヘッドが大量に発生しがちなため、負荷テストに32コアか64コアを推奨します。
NIC：出力トラフィックはNICに制限されるため、大きいファイルのトラフィック負荷テストには10GB以上のNICを推奨します。
リクエストするネットワークリンク：パブリックネットワークのリンクは質にばらつきがあります。また、パブリックネットワークからダウンロードするとき、パブリックネットワーク下りトラフィック料金が発生します。そのため、同じリージョンではプライベートネットワークアクセスを推奨します。
テスト時間：性能テストを実施するとき、比較的安定する値を測定できるように、テスト時間を長めにすることを推奨します。
テスト環境：プログラムが実行するJDKのバージョンは、性能に影響を与えます。例えば、HTTPSのテストで、古いバージョンのクライアントには暗号化アルゴリズムのGCM BUG、乱数ジェネレータにはロックなどの問題が発生することがあります。

COSBench 実行手順

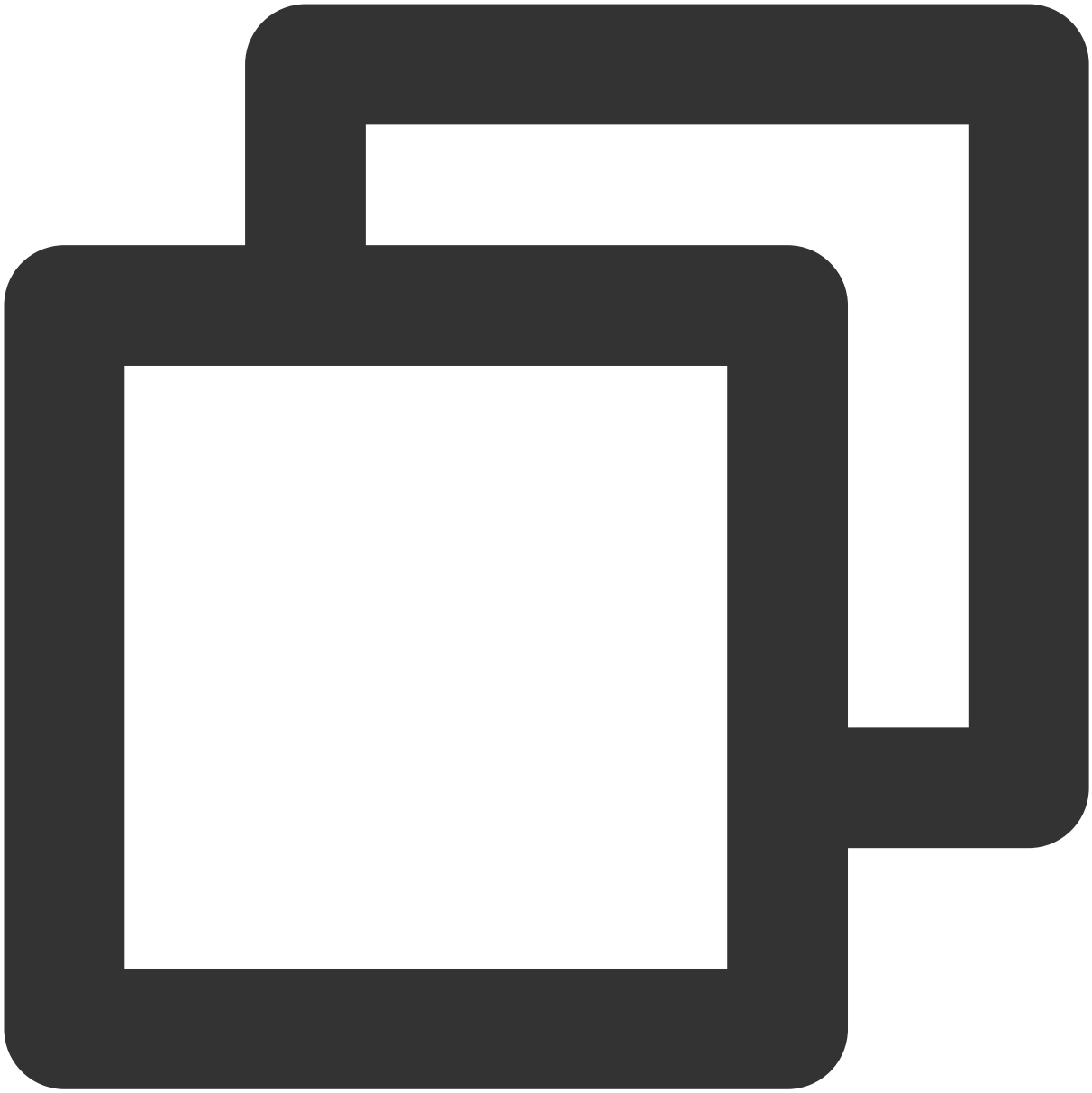
1. COSBench GitHubウェブサイトからCOSBench 0.4.2.c4.zip圧縮パッケージをダウンロードし、サーバー上で解凍します。
2. 以下のコマンドを実行して、COSBenchの依存パッケージをインストールします。

centosの場合、以下のコマンドを実行して、依存パッケージをインストールします：



```
sudo yum install nmap-ncat java curl java-1.8.0-openjdk-devel -y
```

-ubuntuの場合、以下のコマンドを実行して、依存パッケージをインストールします



```
sudo apt install nmap openjdk-8-jdk
```

3. s3-config-sample.xmlファイルを編集し、タスクの設定情報を追加します。タスク設定は以下の5段階に分けられています：

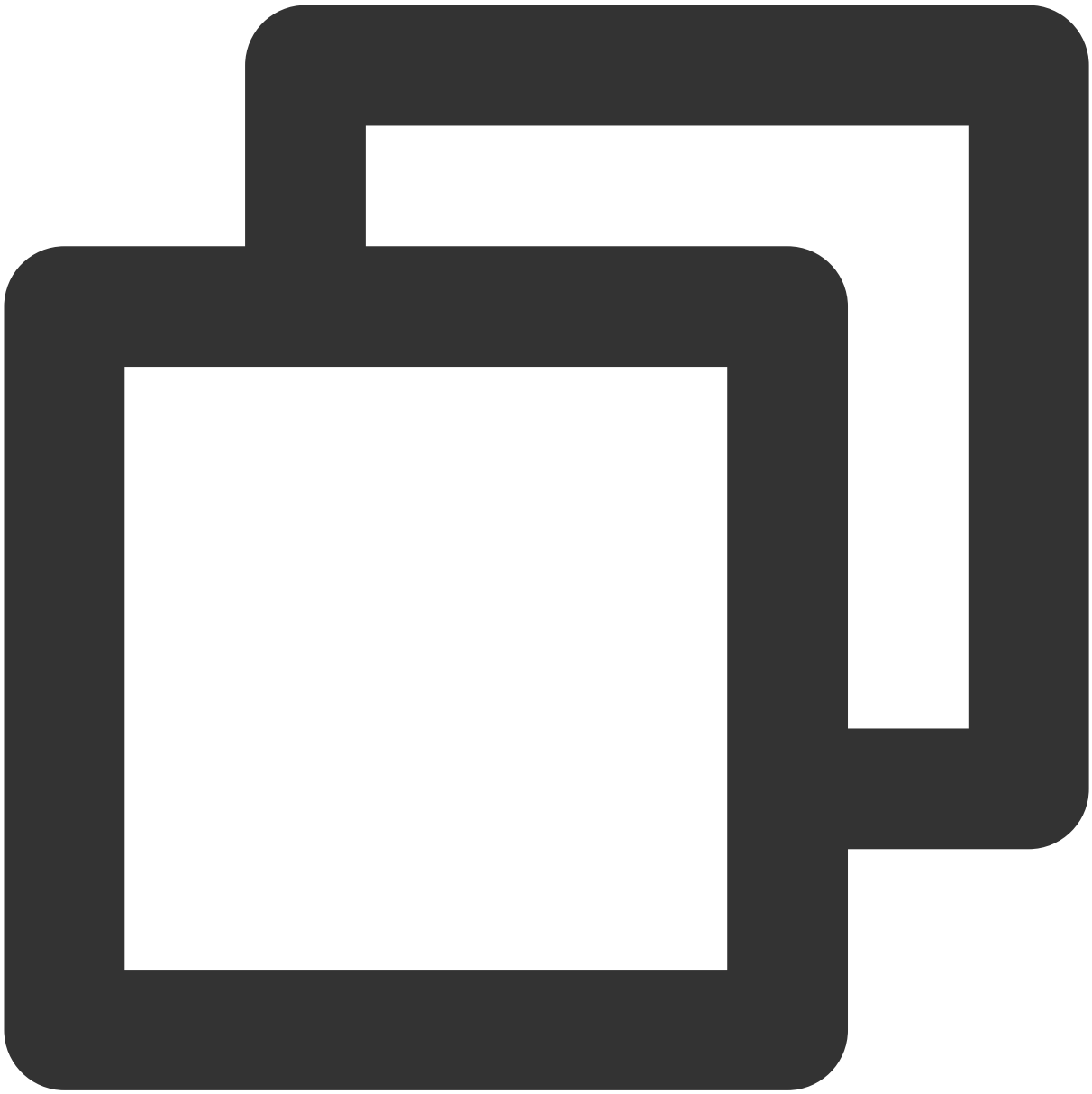
3.1 init段階：バケットを作成します。

3.2 prepare段階：main段階の準備として、workerスレッドを使用し、PUT操作で指定したサイズのオブジェクトをアップロードします。

3.3 main段階：workerスレッドがオブジェクトを指定した時間で読み書きします。

3.4 cleanup段階：生成されたオブジェクトを削除します。

3.5 dispose段階：バケットを削除します。
設定例を以下に示します：



```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="s3-50M-sample" description="sample benchmark for s3">

  <storage type="s3" config="accesskey=AKIDHZRLB9Ibhdp7Y7gyQq6BOk1997xxxxxx;secretk

  <workflow>

    <workstage name="init">
```

```
        <work type="init" workers="10" config="cprefix=examplebucket;csuffix=-1250000
      </workstage>

      <workstage name="prepare">
        <work type="prepare" workers="100" config="cprefix=examplebucket;csuffix=-125
      </workstage>

      <workstage name="main">
        <work name="main" workers="100" runtime="300">
          <operation type="read" ratio="50" config="cprefix=examplebucket;csuffix=-12
          <operation type="write" ratio="50" config="cprefix=examplebucket;csuffix=-1
        </work>
      </workstage>

      <workstage name="cleanup">
        <work type="cleanup" workers="10" config="cprefix=examplebucket;csuffix=-1250
      </workstage>

      <workstage name="dispose">
        <work type="dispose" workers="10" config="cprefix=examplebucket;csuffix=-1250
      </workstage>

    </workflow>

  </workload>
```

パラメータの説明

パラメータ	説明
accesskey、secretkey	キー情報。サブアカウントキーを使用し、最小権限ガイドに従うことで、使用上のリスクを低減させることをお勧めします。サブアカウントキーの取得については、サブアカウントのアクセスキー管理をご参照ください
cprefix	バケット名のプレフィックス。例えば、examplebucket
containers	パケット名中の数字の範囲。最後のパケット名はcprefixとcontainersから構成されます。例えば、examplebucket1、examplebucket2
csuffix	ユーザーのAPPID。APPIDの前に-の記号を付けることに注意してください（例：-1250000000）
runtime	負荷テストの実行時間
ratio	読み取りと書き込みの比率
workers	負荷テストのスレッド数

4. cosbench-start.shファイルを編集します。Java起動行に以下のパラメータを追加して、s3のmd5チェックサム機能を無効にします。```plaintext-Dcom.amazonaws.services.s3.disableGetObjectMD5Validation=true```5. cosbenchサービスを起動します。 - centosの場合、以下のコマンドを実行します：```plaintextsudo bash start-all.sh``` - ubuntuの場合、以下のコマンドを実行します：```plaintextsudo bash start-driver.sh &sudo bash start-controller.sh &```6. 以下のコマンドを実行してタスクをサブミットします。```plaintextsudo bash cli.sh submit conf/s3-config-sample.xml```さらに、このURL`http://ip:19088/controller/index.html`（ipはユーザーの負荷テストマシンのIPに置き換えます）によって実行状態を確認します。下図に示すように、5段階が表示されます：7. 次の例は、所属リージョンが北京リージョン、32コア、プライベートネットワーク帯域幅が17GbpsのCVMで行うアップロードおよびダウンロードパフォーマンステストです。次の2段階が含まれます。 1. prepare段階：100workerスレッドが50MBのオブジェクトを1000個アップロードします。 2. main段階：100workerスレッドがオブジェクトの読み書きを300秒実行します。
上記の段階1と段階2の性能テストを実行した結果は以下のとおりです：

Basic Info

ID: w27 **Name**: s3-50M-sample **Current State**: finished

Submitted At: 2019-3-20 10:39:53 **Started At**: 2019-3-20 10:39:53 **Stopped At**: 2019-3-20 10:50:29

more info

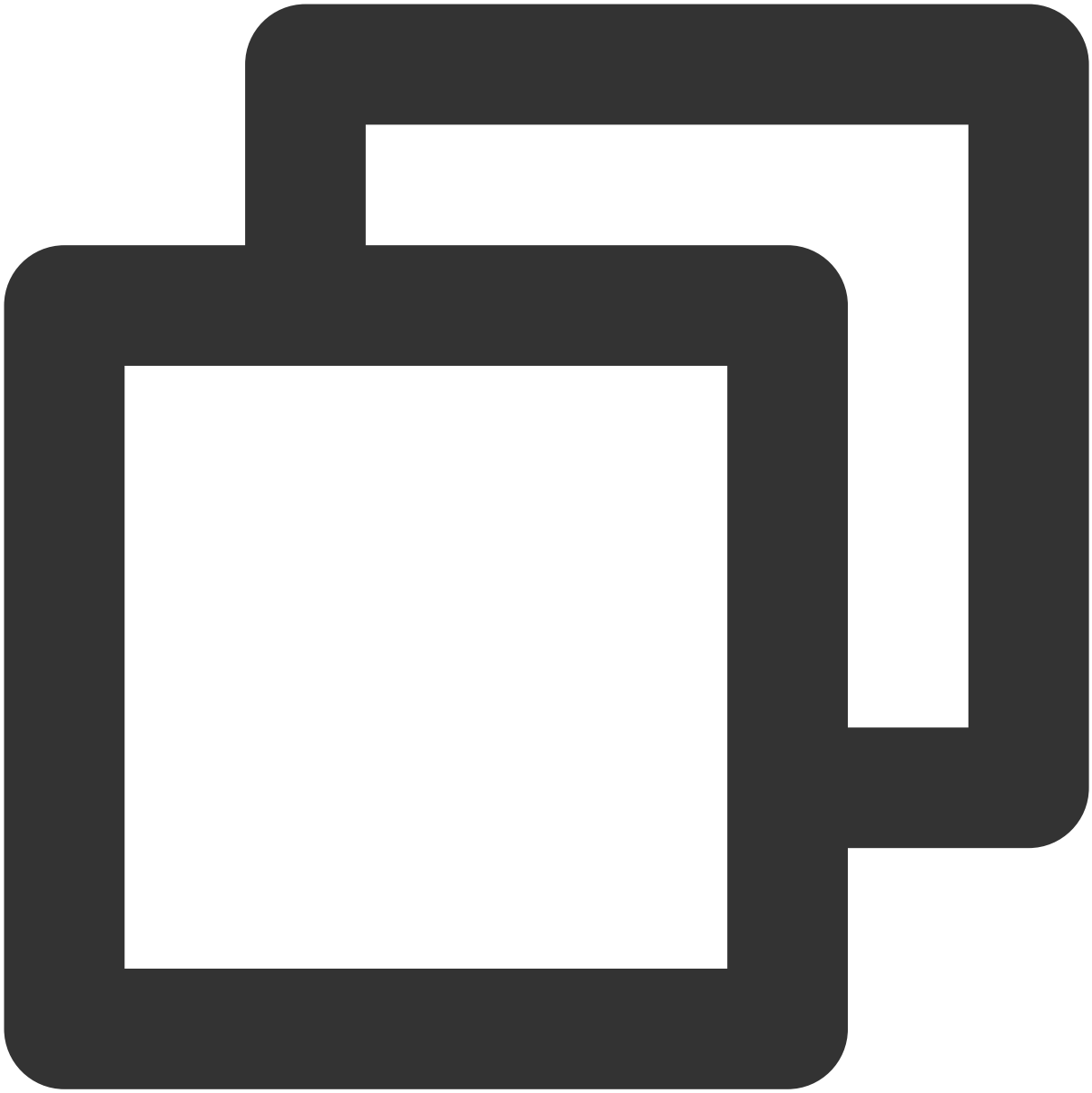
Final Result

General Report

Op-Type	Op-Count	Byte-Count	Avg-ResTime	Avg-ProcTime	Throughput	Bandwidth	Succ-Ratio
op1: init -write	0 ops	0 B	N/A	N/A	0 op/s	0 B/S	N/A
op1: prepare - write	10 kops	500 GB	2460.65 ms	181.83 ms	41.27 op/s	2.06 GB/S	100%
op1: read	10.22 kops	510.75 GB	2012.74 ms	118.33 ms	34.15 op/s	1.71 GB/S	100%
op2: write	10.28 kops	514.2 GB	908.98 ms	317.71 ms	34.38 op/s	1.72 GB/S	100%
op1: cleanup - delete	20 kops	0 B	14.19 ms	14.19 ms	704.74 op/s	0 B/S	100%
op1: dispose - delete	0 ops	0 B	N/A	N/A	0 op/s	0 B/S	N/A

show peformance details

8. 以下のコマンドを実行して、テストサービスを停止します。



```
sudo bash stop-all.sh
```

データマイグレーション サードパーティクラウドストレージのデータをCOSへ移行

最終更新日：：2024-06-26 10:42:27

プラクティスの背景

サードパーティのクラウドプラットフォームを使用しているユーザーの場合、Cloud Object Storage(COS)は、ストレージデータをサードパーティのクラウドプラットフォームからCOSに速やかに移行するときに役立ちます。

移行方法	インタラクティブ形式	ファイルサイズを区別するためのしきい値	移行の同時実行性	HTTPSセキュア送信
Migration Service Platform	ビジュアルページ操作	デフォルト設定の使用	グローバル統合	有効化

このプラットフォームは、データ移行の進捗確認、ファイルの整合性チェック、再送の失敗、中断からの再開などの機能をサポートしており、ユーザーの基本的な移行ニーズを満たすことができます。

移行のプラクティス

Migration Service Platform

Migration Service Platform(MSP)は、さまざまな移行ツールを統合し、大規模なデータ移行タスクをユーザーが手軽に監視、管理できるビジュアルインターフェースを提供するプラットフォームです。中でも「ファイルマイグレーションツール」は、ユーザーがさまざまなパブリッククラウドやデータソースサイトからCOSにデータを移行する際に役立つツールです。

移行操作の手順は次のとおりです。

1. Migration Service Platformコンソールにログインします。
2. 左のナビゲーションバーにある**COS移行**をクリックし、COS移行ページに進みます。
3. **タスクの新規作成**をクリックして、移行タスクを新規作成し、タスク情報を設定します。
4. タスクを起動します。

具体的な操作については、以下の移行チュートリアルをご参照ください。

Alibaba Cloud OSSの移行

Huawei Cloud OBSの移行

Qiniu Cloud KODOの移行

UCLOUD UFileの移行
Kingsoft Cloud KS3の移行
Baidu Cloud BOSの移行
AWS S3の移行

操作テクニック

データ移行プロセスでは、ネットワーク環境によってデータソースの読み込み速度が異なる場合がありますが、お客様は「ファイル移行タスクの新規作成」時に高いQPS同時実行性を選択することで、移行速度を向上させることが可能です。

AWS S3 SDKを使用したCOSアクセス

最終更新日：：2024-06-26 10:42:27

概要

Cloud Object Storage（COS）はAWS S3と互換性のあるAPIを提供しているため、データをS3からCOSに移行した後、簡単な設定変更を行うだけで、クライアントアプリケーションに簡単にCOSサービスとの互換性を持たせることができます。ここでは主に、各開発プラットフォームにおけるS3 SDKの適用の手順についてご説明します。適用手順の追加が完了すると、S3 SDKのインターフェースを使用してCOS上のファイルにアクセスできるようになります。

準備作業

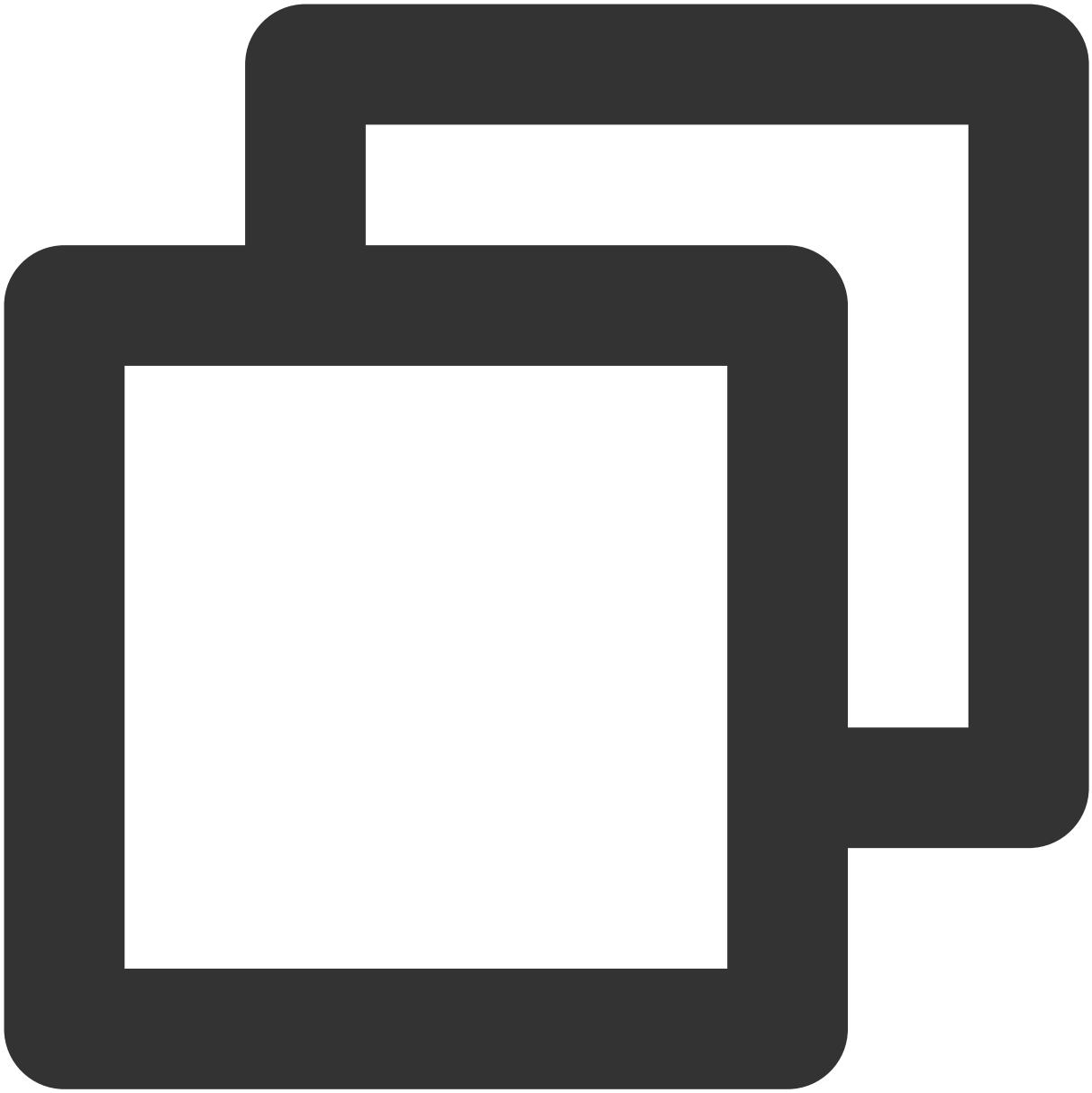
1. Tencent Cloudアカウントの登録が完了しており、なおかつCAMコンソール上でTencent CloudキーのSecretIDとSecretKeyを取得済みであることとします。
2. S3 SDKを統合済みで、正常に実行可能なクライアントアプリケーションがすでにあることとします。

Android

以下ではAWS Android SDK 2.14.2バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。端末からCOSへのアクセスについては、パーマネントキーをクライアントコード内に埋め込むと、開示されるリスクが極めて大きいため、STSサービスにアクセスして一時キーを取得することをお勧めします。詳細については、一時キーの生成および使用ガイドをご参照ください。

初期化

インスタンスを初期化する際、一時キープロバイダおよびEndpointを設定する必要があります。バケットの所在リージョンが `ap-guangzhou` の場合を例にとります。



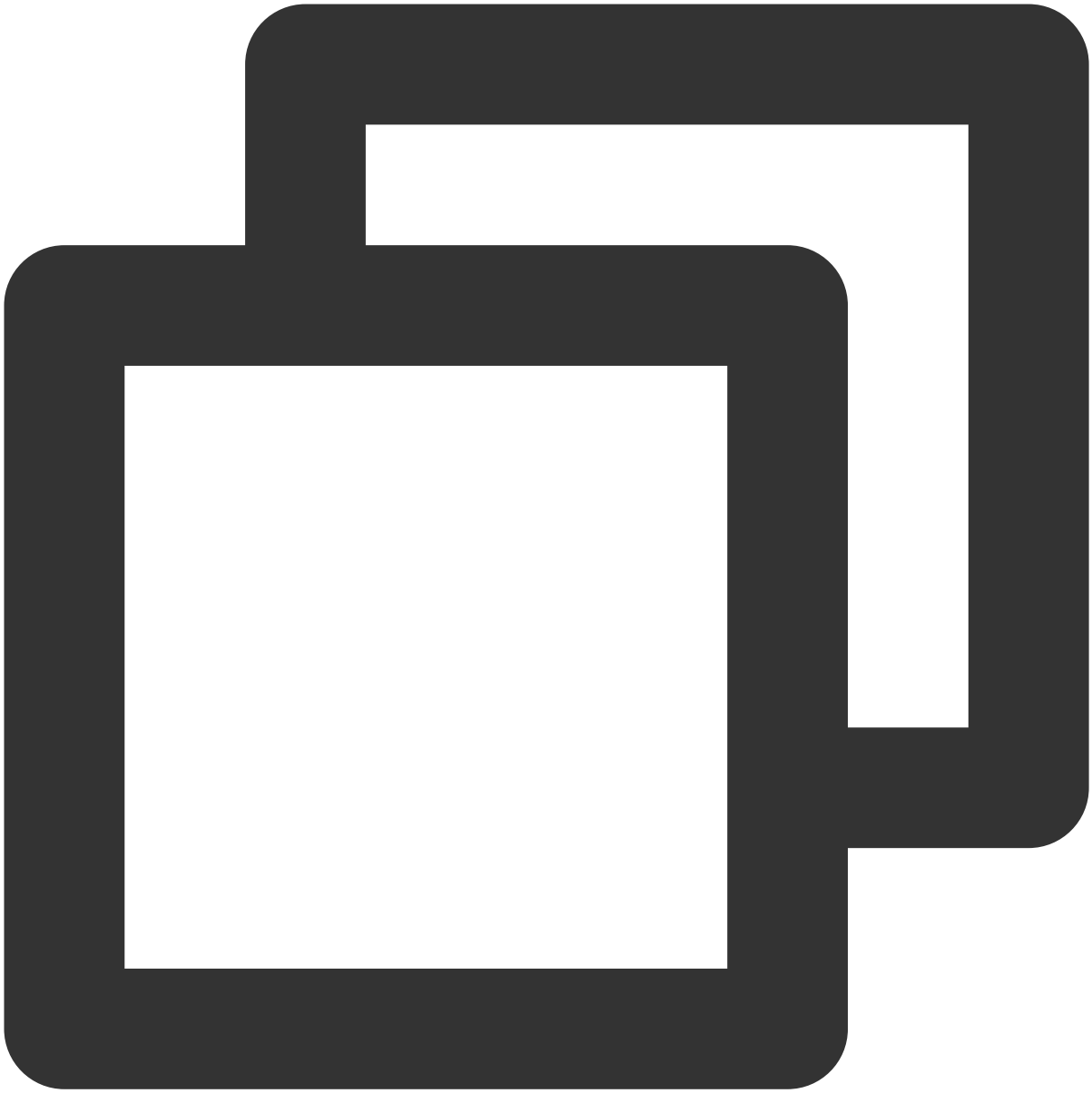
```
AmazonS3Client s3 = new AmazonS3Client(new AWSCredentialsProvider() {
    @Override
    public AWSCredentials getCredentials() {
         // ここではバックエンドがSTSにリクエストし、一時キー情報を取得します
        return new BasicSessionCredentials(
                "<TempSecretID>", "<TempSecretKey>", "<STSSessionToken>"
        );
    }

    @Override
    public void refresh() {
```

```
            //
        }
});

s3.setEndpoint("cos.ap-guangzhou.myqcloud.com");
```

iOS

AWS iOS SDK 2.10.2バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。端末からCOSへのアクセスについては、パーマネントキーをクライアントコード内に埋め込むと、開示されるリスクが極めて大きいため、STSサービスにアクセスして一時キーを取得することをお勧めします。詳細については、一時キーの生成および使用ガイドをご参照ください。

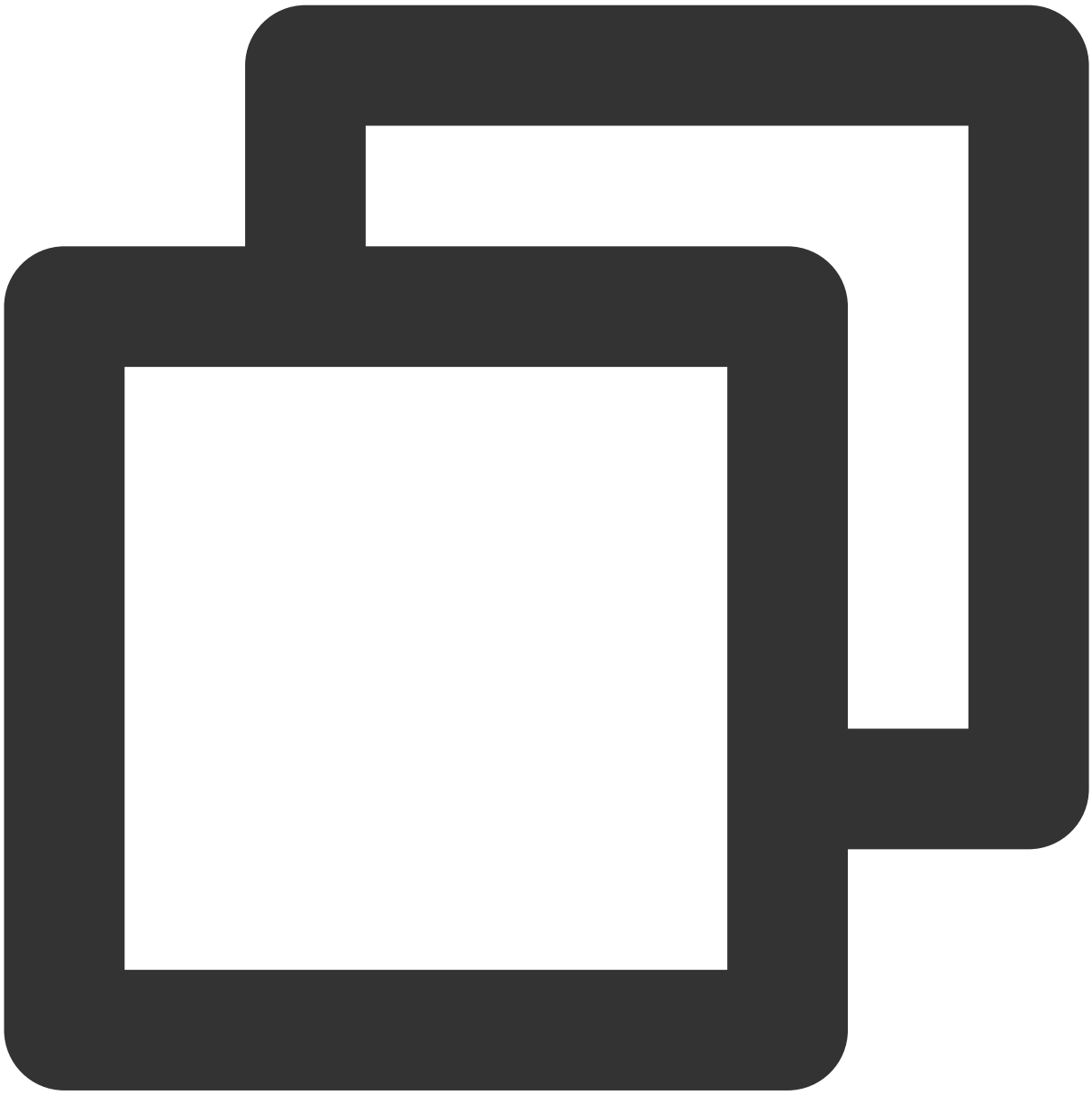
1. AWSCredentialsProviderプロトコルの実装



```
-(AWSTask<AWSCredentials *> *)credentials{
    // ここではバックエンドがSTSにリクエストし、一時キー情報を取得します
    AWSCredentials *credential = [[AWSCredentials alloc]initWithAccessKey:@"<TempSe

    return [AWSTask taskWithResult:credential];

}

- (void)invalidateCachedTemporaryCredentials{

}
```

2. 一時キープロバイダおよびEndpointの提供

バケットの所在リージョンが `ap-guangzhou` の場合を例にとります。



```
NSURL* bucketURL = [NSURL URLWithString:@"http://cos.ap-guangzhou.myqcloud.com"];

AWSEndpoint* endpoint = [[AWSEndpoint alloc] initWithRegion:AWSRegionUnknown servic
AWSServiceConfiguration* configuration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast2 endpoint:endpoint
    credentialsProvider:[MyCredentialProvider new]]; // MyCredentialProviderがAWSCre
```

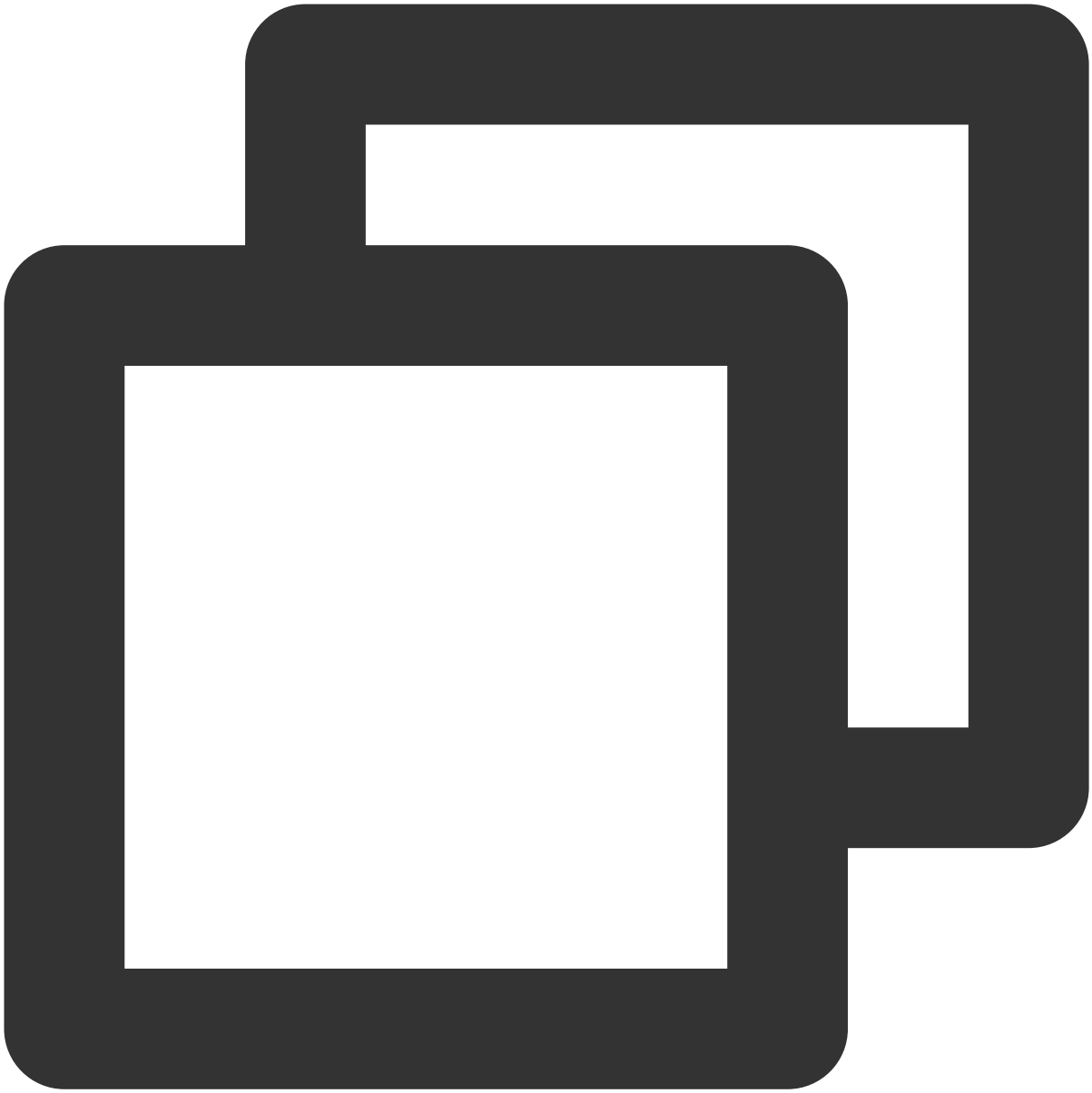
```
[[AWSServiceManager defaultServiceManager] setDefaultServiceConfiguration:configura
```

Node.js

以下ではAWS JS SDK 2.509.0バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。

初期化

インスタンスを初期化する際にTencent Cloudキーおよびを設定します。バケットの所在リージョンが `ap-guangzhou` の場合のコードの例は次のようになります。



```
var AWS = require('aws-sdk');

AWS.config.update({
    accessKeyId: "COS_SECRETID",
    secretAccessKey: "COS_SECRETKEY",
    region: "ap-guangzhou",
    endpoint: 'https://cos.ap-guangzhou.myqcloud.com',
});

s3 = new AWS.S3({apiVersion: '2006-03-01'});
```

Java

以下ではAWS Java SDK 1.11.609バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。

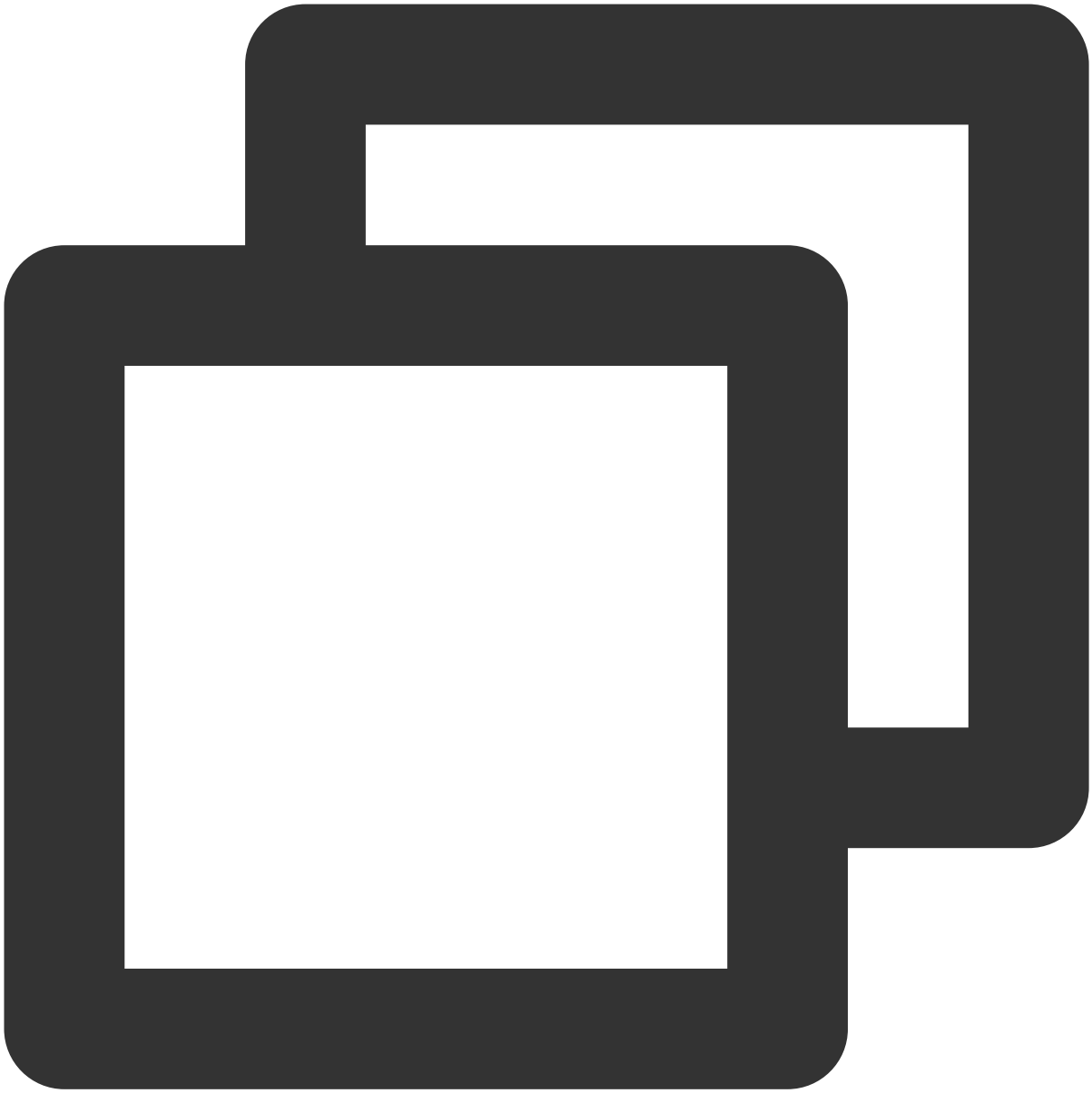
1. AWSの設定および証明書ファイルを変更する

説明：

以下ではLinuxを例に、AWSの設定および証明書ファイルの変更を行います。

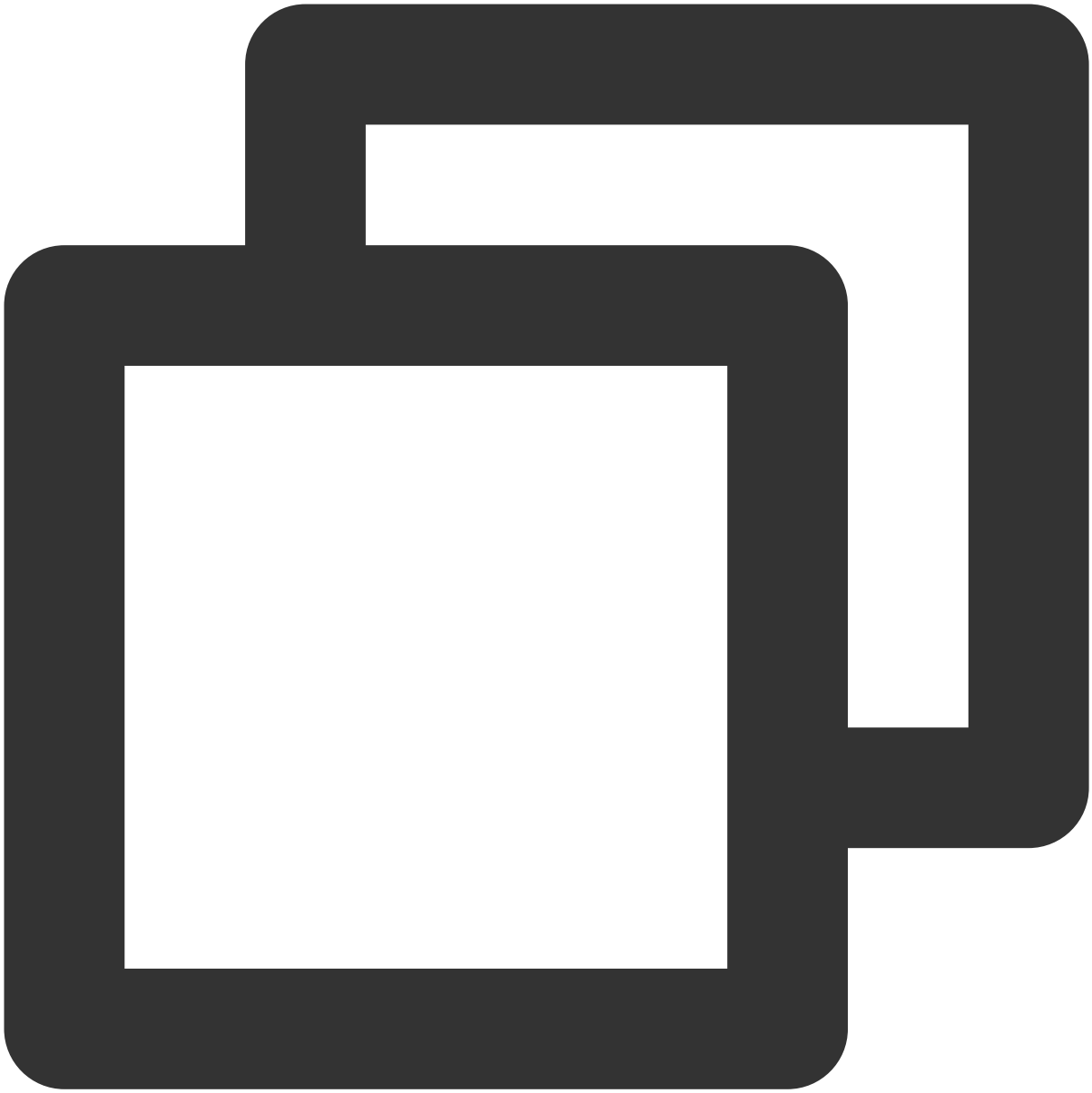
AWS SDKのデフォルトの設定ファイルは通常、ユーザーディレクトリ下にあります。設定および証明書ファイルをご参照ください。

設定ファイル（ファイルの位置は `~/.aws/config` ）に次の設定情報を追加します。



```
[default]
s3 =
addressing_style = virtual
```

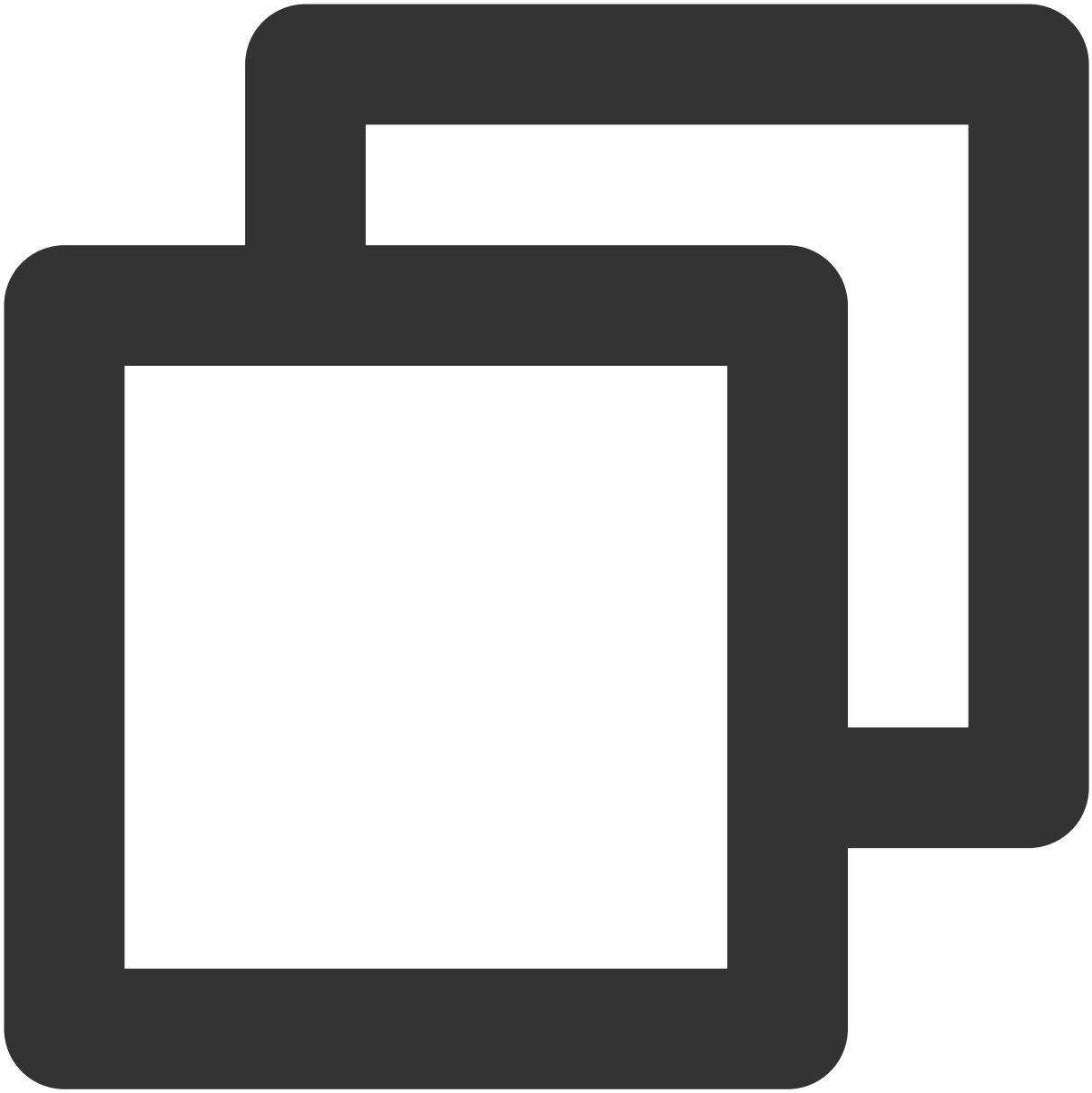
証明書ファイル（ファイルの位置は `~/.aws/credentials` ）にTencent Cloudのキーを設定します。



```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. コードにEndpointを設定する

バケットの所在リージョンが `ap-guangzhou` の場合のコードの例は次のようになります。



```
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration(
            "http://cos.ap-guangzhou.myqcloud.com",
            "ap-guangzhou"))
    .build();
```

Python

以下ではAWS Python SDK 1.9.205バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。

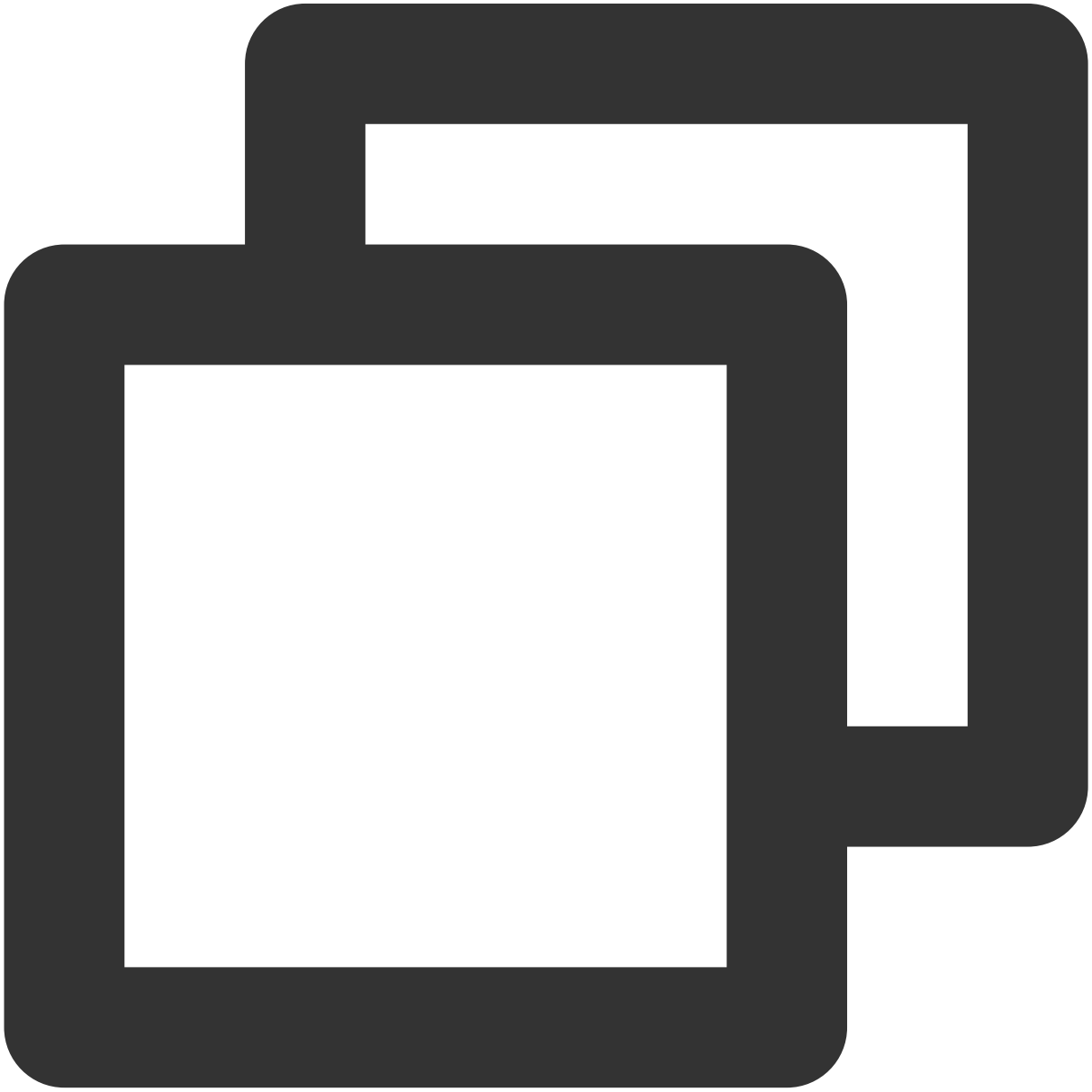
1. AWSの設定および証明書ファイルを変更する

説明：

以下ではLinuxを例に、AWSの設定および証明書ファイルの変更を行います。

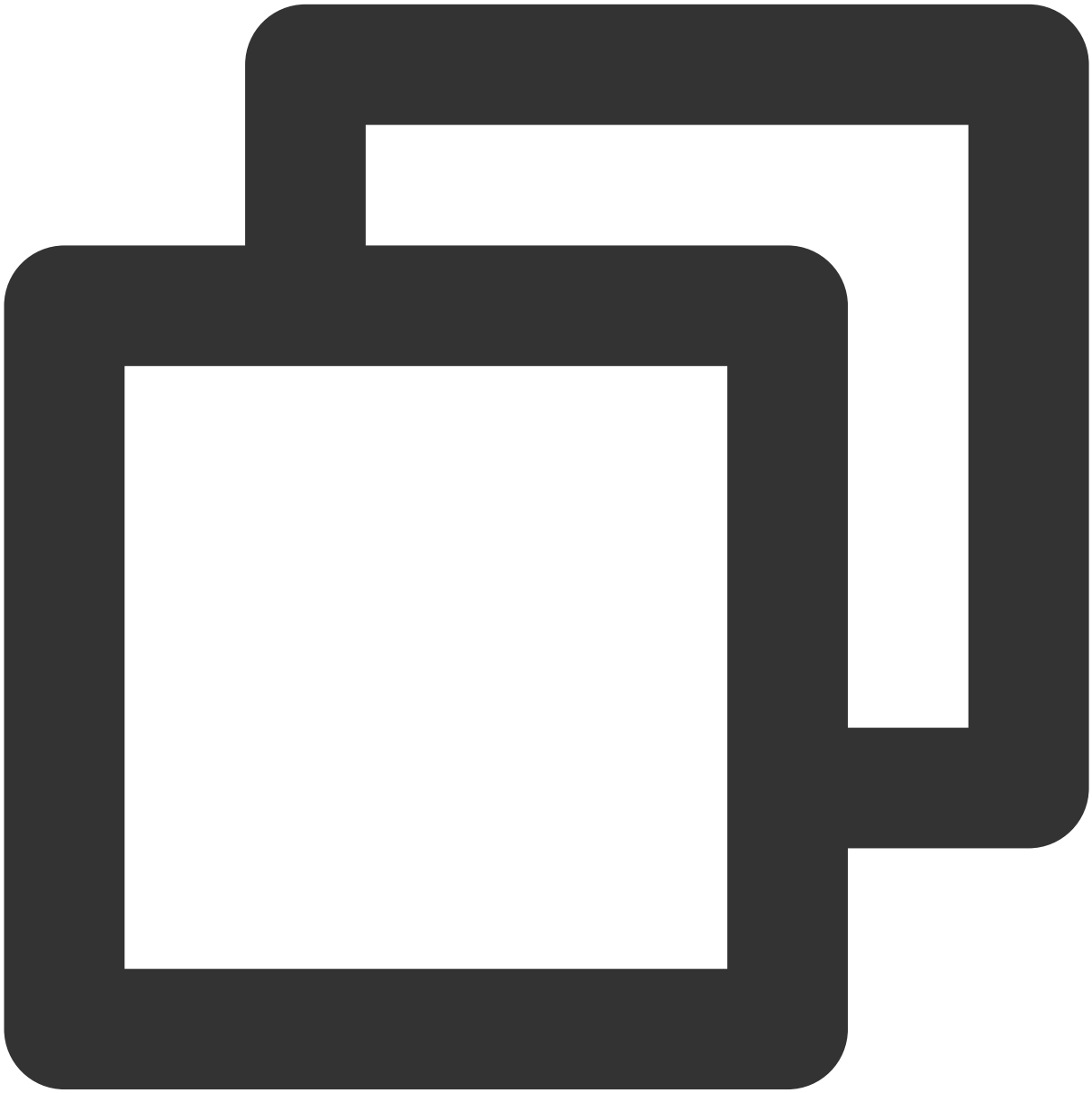
AWS SDKのデフォルトの設定ファイルは通常、ユーザーディレクトリ下にあります。設定および証明書ファイルをご参照ください。

設定ファイル（ファイルの位置は `~/.aws/config` ）に次の設定を追加します。



```
[default]
s3 =
  signature_version = s3
  addressing_style = virtual
```

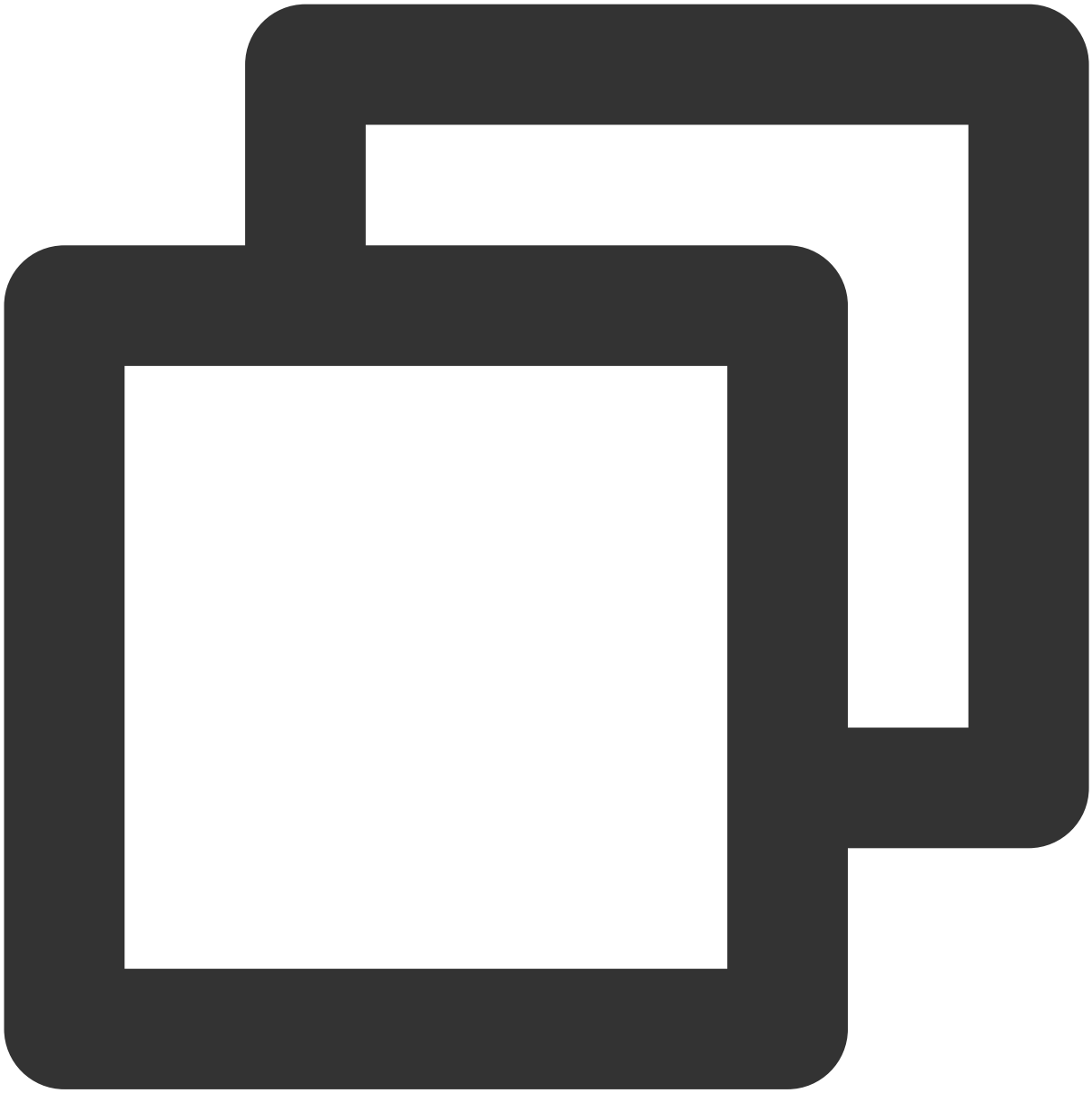
証明書ファイル（ファイルの位置は `~/.aws/credentials` ）にTencent Cloudのキーを設定します。



```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. コードにEndpointを設定する

バケットの所在リージョンが `ap-guangzhou` の場合を例にとります。



```
client = boto3.client('s3', endpoint_url='https://cos.ap-guangzhou.myqcloud.com')
```

PHP

以下ではAWS PHP SDK 3.109.3バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。

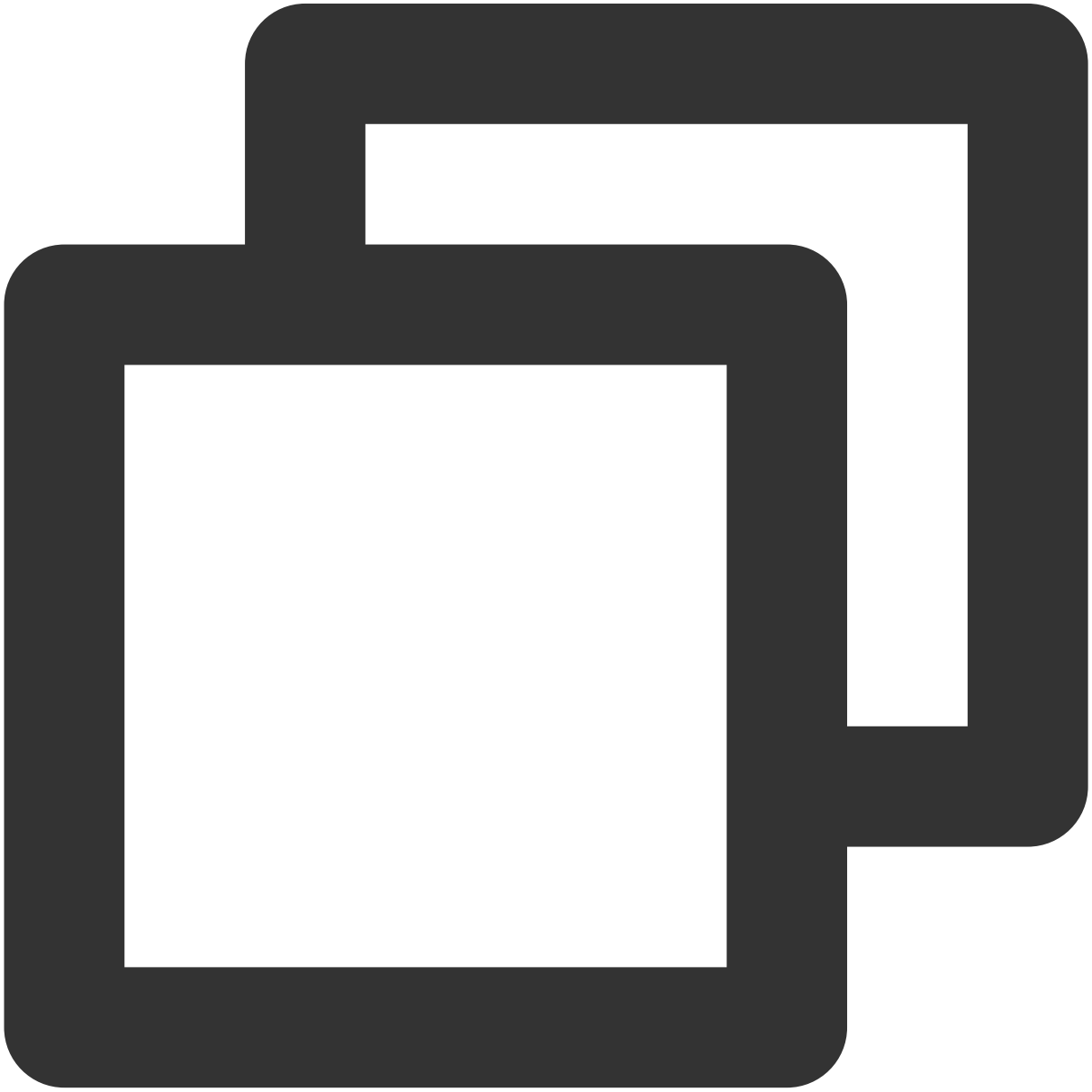
1. AWSの設定および証明書ファイルを変更する

説明：

以下ではLinuxを例に、AWSの設定および証明書ファイルの変更を行います。

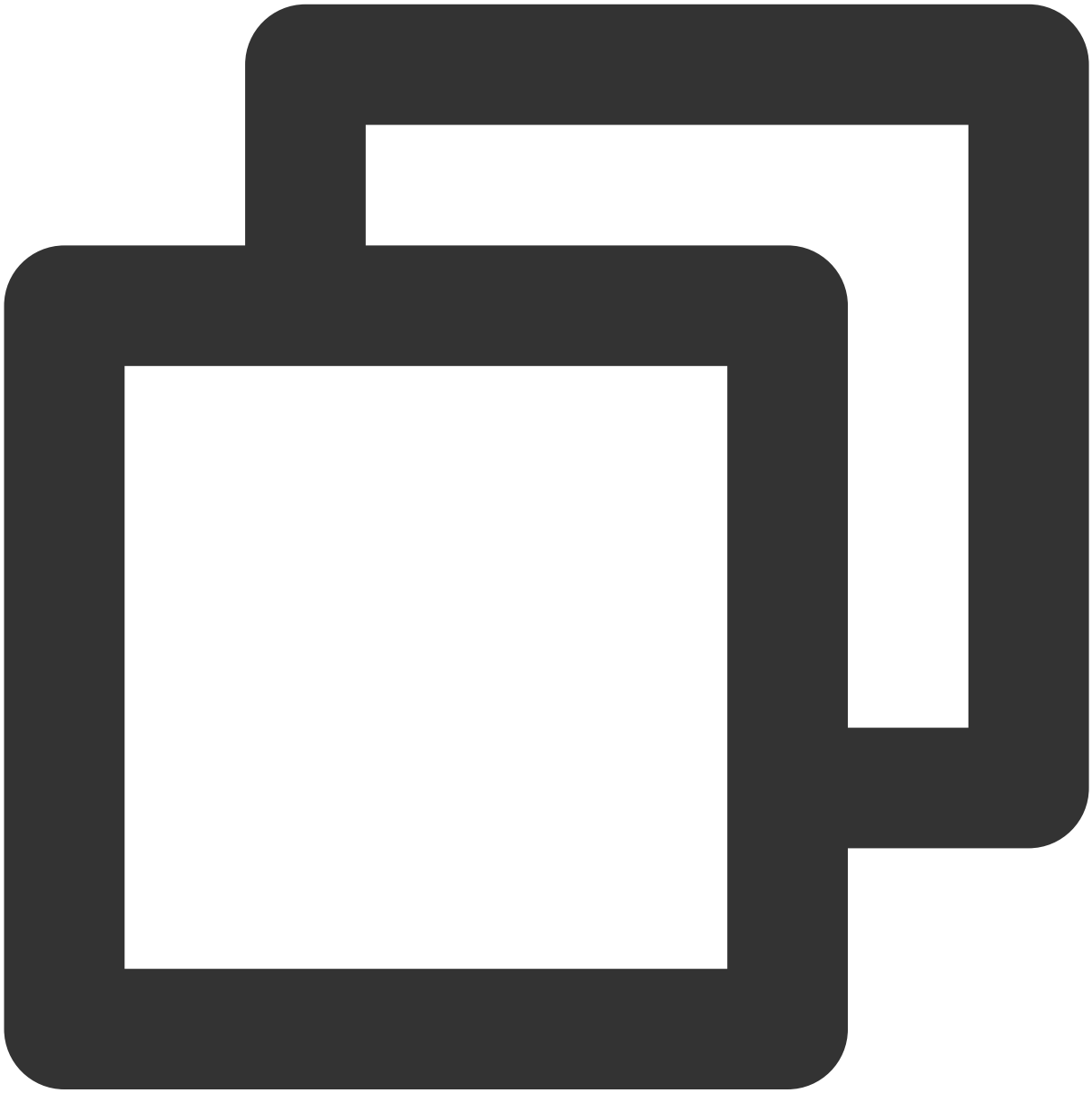
AWS SDKのデフォルトの設定ファイルは通常、ユーザーディレクトリ下にあります。設定および証明書ファイルをご参照ください。

設定ファイル（ファイルの位置は `~/.aws/config` ）に次の設定を追加します。



```
[default]
s3 =
addressing_style = virtual
```

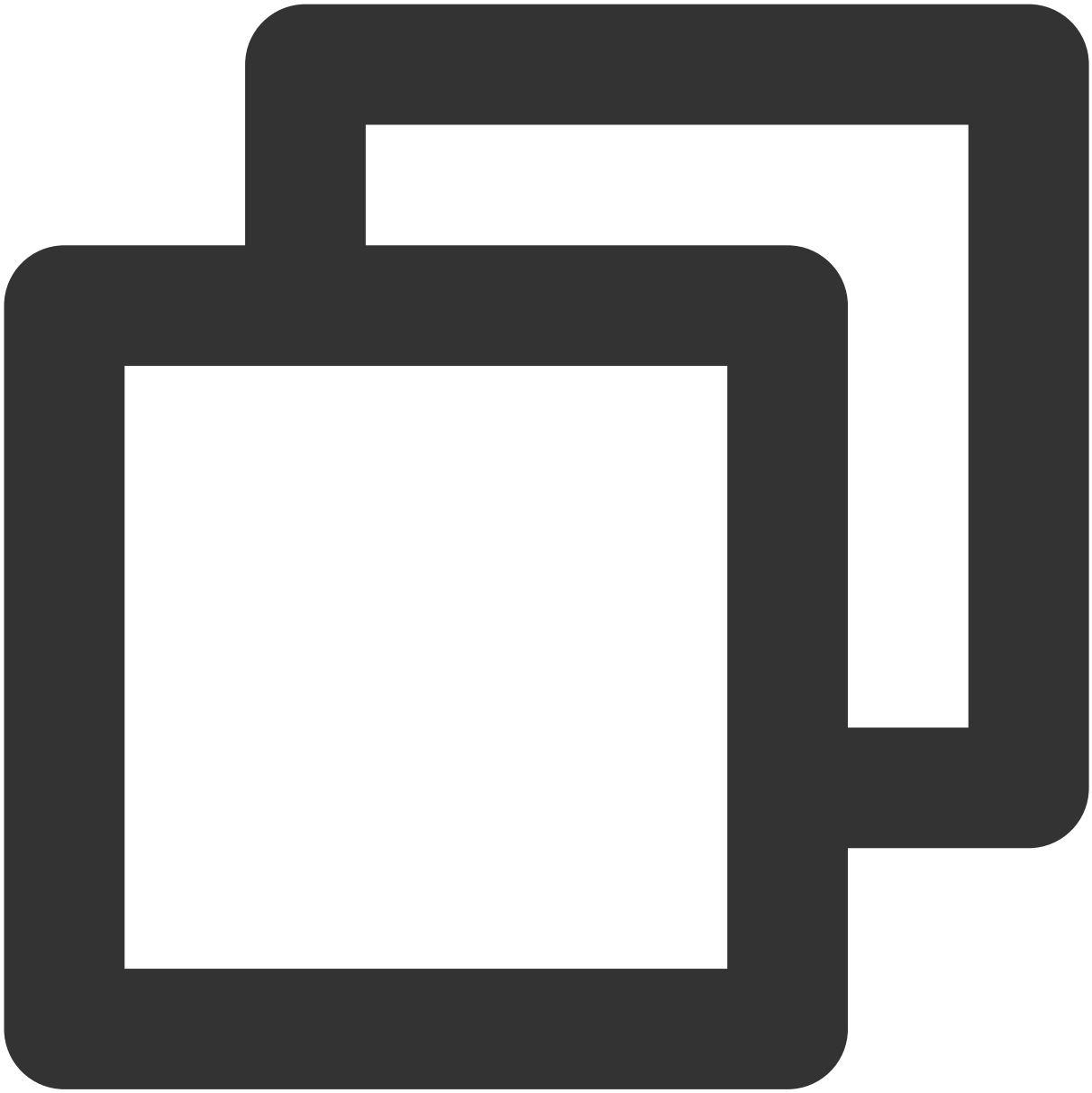
証明書ファイル（ファイルの位置は `~/.aws/credentials` ）にTencent Cloudのキーを設定します。



```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. コードにEndpointを設定する

バケットの所在リージョンが `ap-guangzhou` の場合を例にとります。



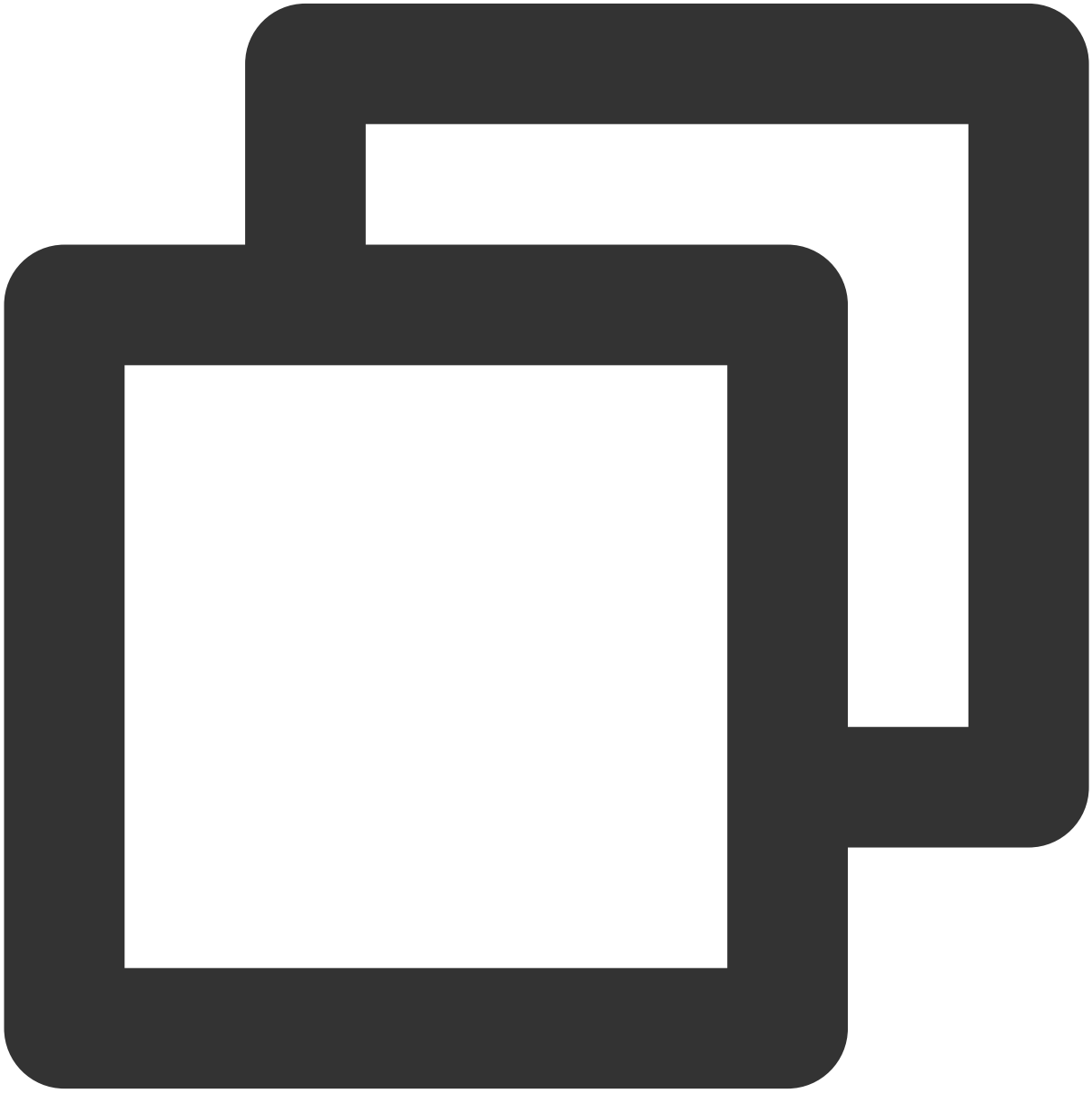
```
$S3Client = new S3Client([
  'region'          => 'ap-guangzhou',
  'version'         => '2006-03-01',
  'endpoint'        => 'https://cos.ap-guangzhou.myqcloud.com'
]);
```

.NET

以下ではAWS .NET SDK 3.3.104.12バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。

初期化

インスタンスを初期化する際にTencent CloudキーおよびEndpointを設定します。バケットの所在リージョンが `ap-guangzhou` の場合を例にとります。



```
string sAccessKeyId = "COS_SECRETID";
```

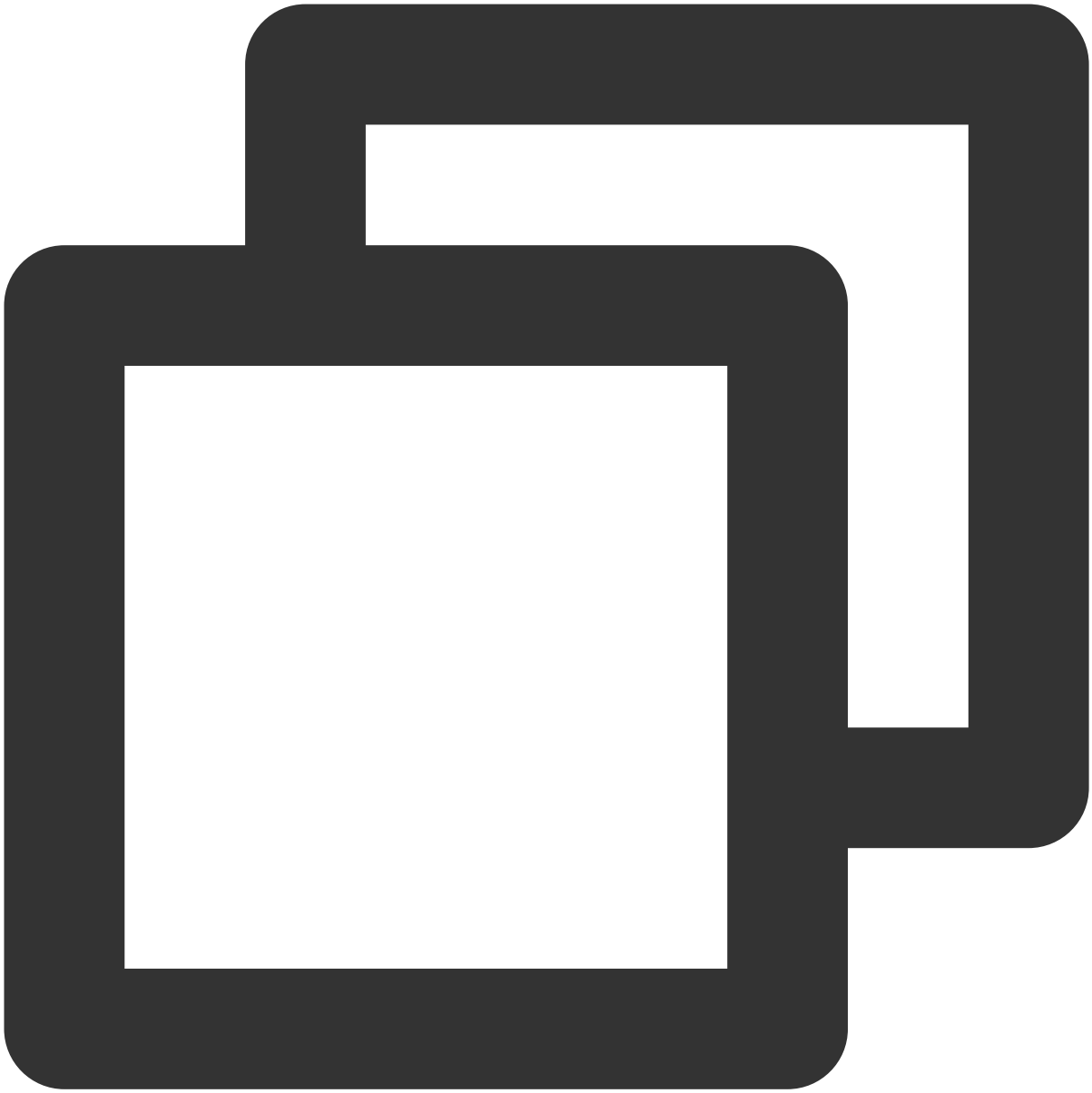
```
string sAccessKeySecret = "COS_SECRETKEY";
string region = "ap-guangzhou";

var config = new AmazonS3Config() { ServiceURL = "https://cos." + region + ".myqclo
var client = new AmazonS3Client(sAccessKeyId, sAccessKeySecret, config);
```

Go

以下ではAWS Go SDK 1.21.9バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。

1. キーによってsessionを作成する

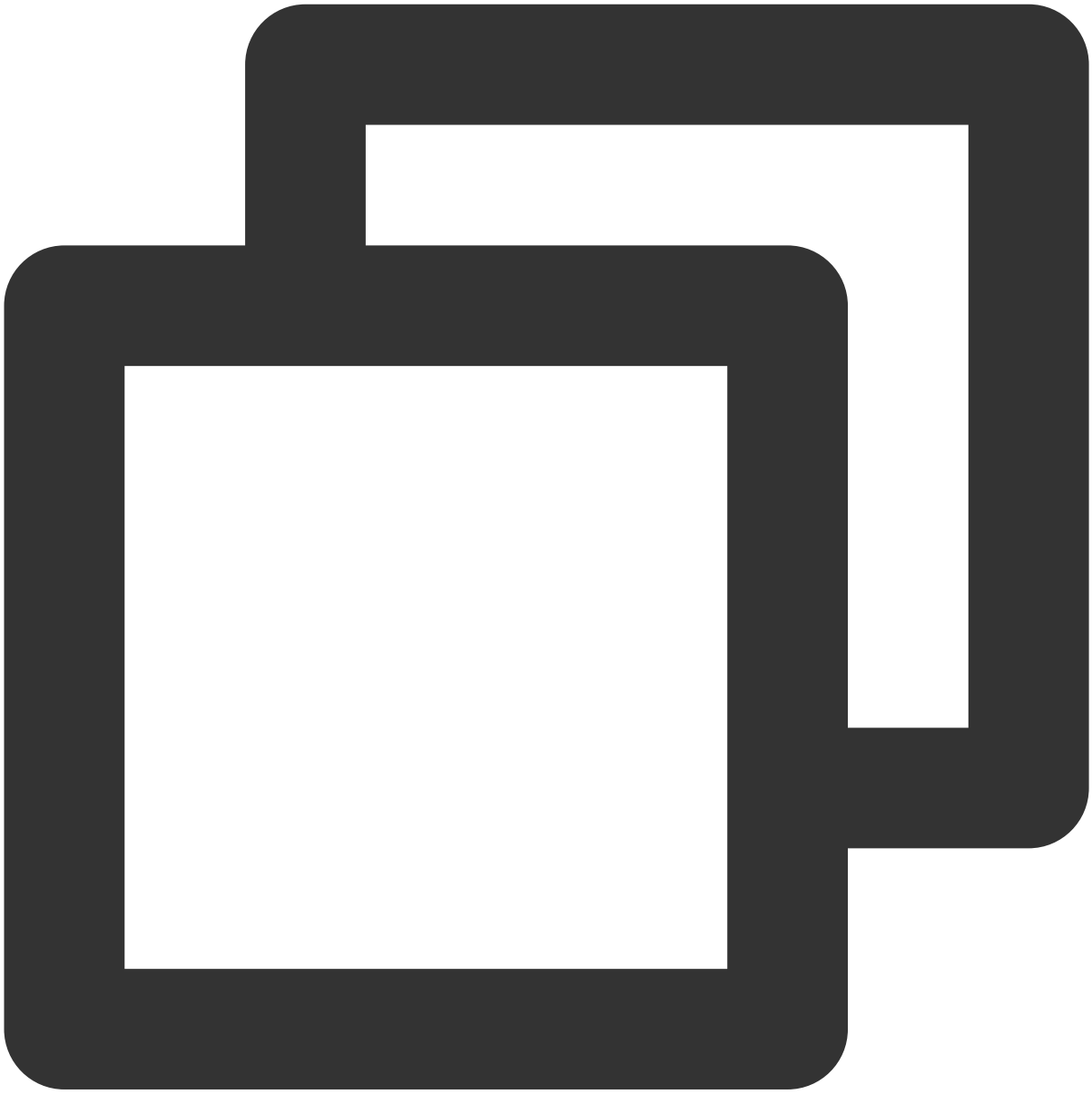
バケットの所在リージョンが `ap-guangzhou` の場合を例にとります。



```
func newSession() (*session.Session, error) {
    creds := credentials.NewStaticCredentials("COS_SECRETID", "COS_SECRETKEY", "")
    region := "ap-guangzhou"
    endpoint := "http://cos.ap-guangzhou.myqcloud.com"
    config := &aws.Config{
        Region:           aws.String(region),
        Endpoint:         &endpoint,
        S3ForcePathStyle: aws.Bool(true),
        Credentials:      creds,
        // DisableSSL:       &disableSSL,
    }
```

```
    return session.NewSession(config)
}
```

2. sessionによってserverを作成し、リクエストを送信する



```
sess, _ := newSession()
service := s3.New(sess)

// ファイルのアップロードの例
fp, _ := os.Open("yourLocalFilePath")
defer fp.Close()
```

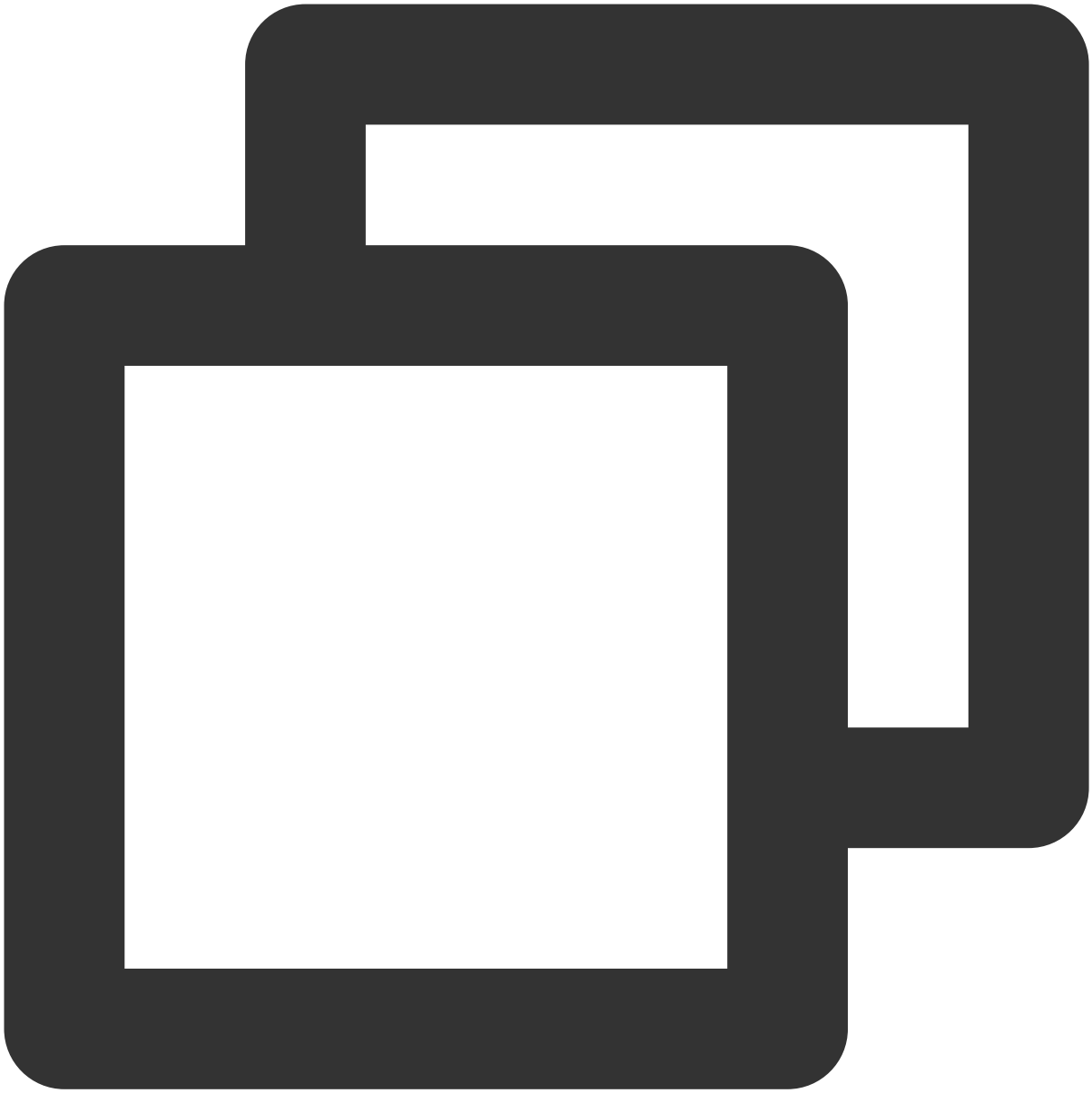
```
ctx, cancel := context.WithTimeout(context.Background(), time.Duration(30)*time.Sec
defer cancel()

service.PutObjectWithContext(ctx, &s3.PutObjectInput{
    Bucket: aws.String("examplebucket-1250000000"),
    Key:    aws.String("exampleobject"),
    Body:   fp,
})
```

C++

以下ではAWS C++ SDK 1.7.68バージョンを例に、COSサービスにアクセスできるように適用する方法についてご説明します。

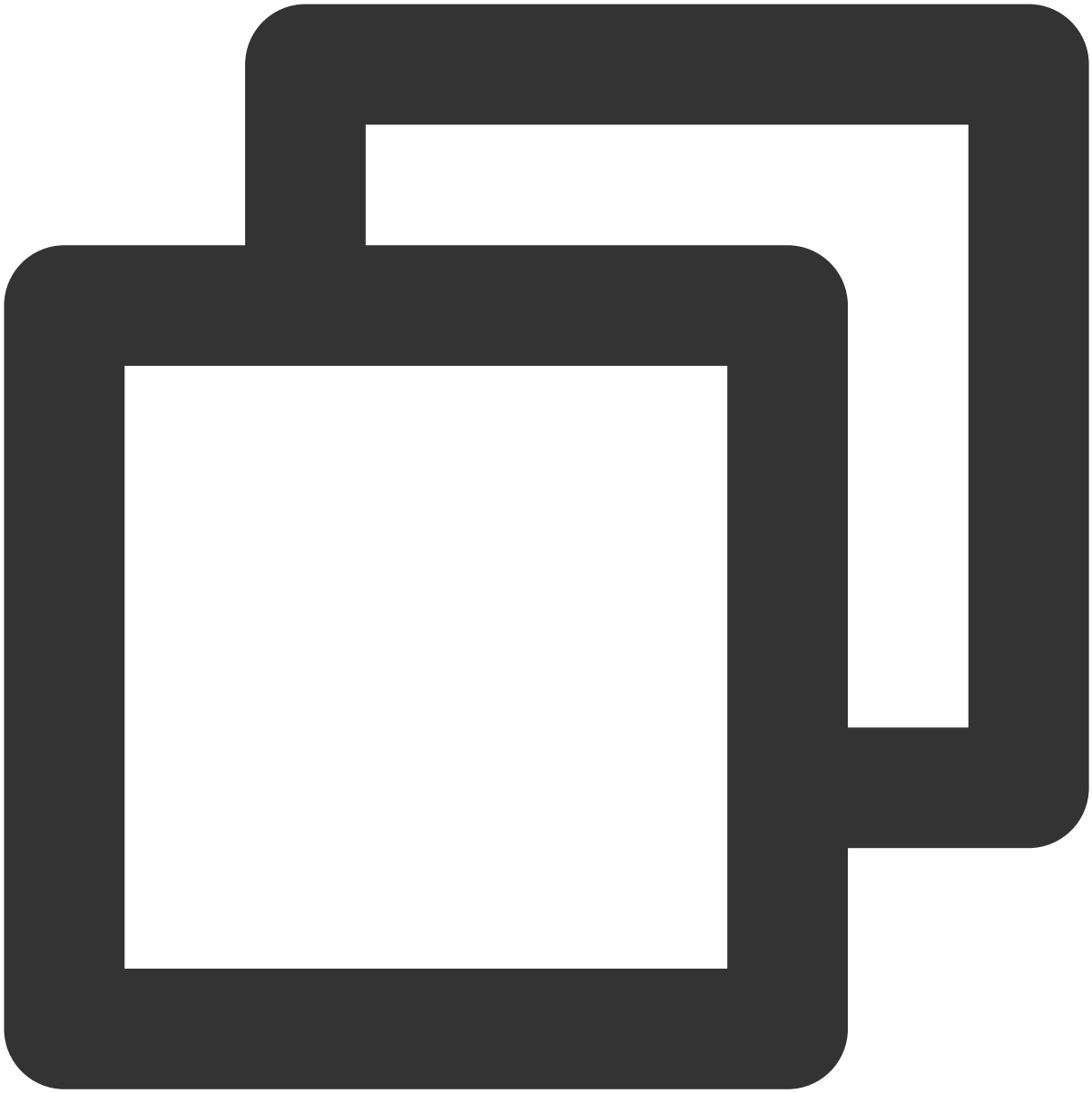
1. AWSの設定および証明書ファイルを変更する

説明：
以下ではLinuxを例に、AWSの設定および証明書ファイルの変更を行います。
AWS SDKのデフォルトの設定ファイルは通常、ユーザーディレクトリ下にあります。設定および証明書ファイルをご参照ください。
設定ファイル（ファイルの位置は `~/.aws/config` ）に次の設定を追加します。



```
[default]
s3 =
addressing_style = virtual
```

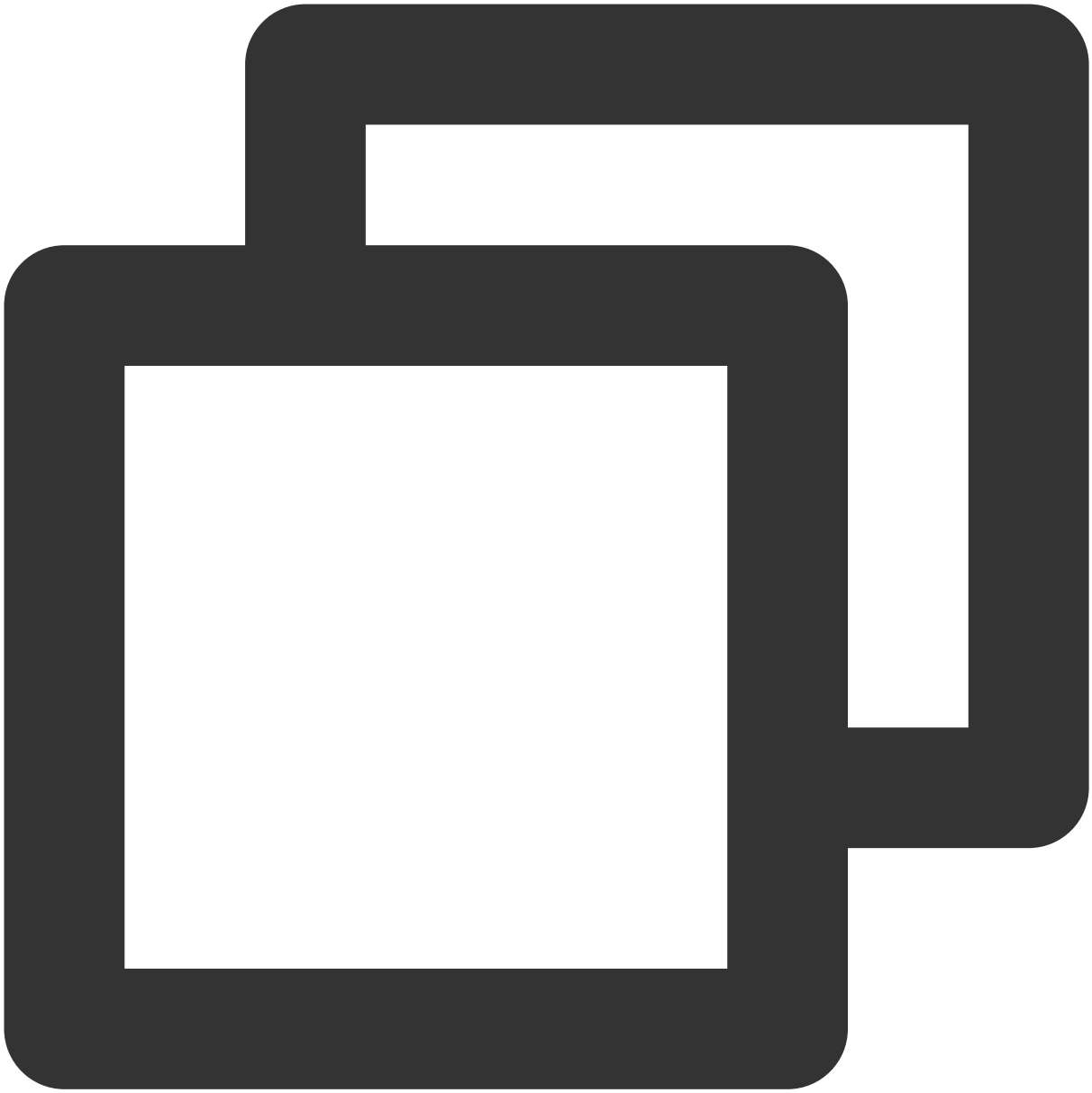
証明書ファイル（ファイルの位置は `~/.aws/credentials` ）にTencent Cloudのキーを設定します。



```
[default]
aws_access_key_id = [COS_SECRETID]
aws_secret_access_key = [COS_SECRETKEY]
```

2. コードにEndpointを設定する

バケットの所在リージョンが `ap-guangzhou` の場合のコードの例は次のようになります。



```
Aws::Client::ClientConfiguration awsCC;
awsCC.scheme = Aws::Http::Scheme::HTTP;
awsCC.region = "ap-guangzhou";
awsCC.endpointOverride = "cos.ap-guangzhou.myqcloud.com";
Aws::S3::S3Client s3_client(awsCC);
```

ドメイン名管理の実践
バケットをカスタムドメインに切り替える

最終更新日：：2024-07-05 18:48:44

背景

サービス全体の安全性と安定性を確保するため、2024 年 1 月 1 日以降に作成されたバケットについて、COS のデフォルトドメイン名を使用してオブジェクトにアクセスする場合、あらゆるタイプのファイルのプレビューはサポートされず、apk/ipa タイプのファイルのダウンロードもサポートされていません。詳細については、COS バケットドメイン名使用のセキュリティ管理に関する通知を参照してください。

2024 年 1 月 1 日以降に作成されたバケットでは、ブラウザからファイルをプレビューしたり、バケット内の apk/ipa タイプのオブジェクトを直接ダウンロードしたりする場合、ユーザーがカスタムドメイン名を使ってオブジェクトにアクセスすることが推奨されます。2024 年 1 月 1 日以前に作成されたバケットについては、バケットのデフォルトドメイン名のプレビューおよびダウンロード動作に影響はありませんが、より良いサービスの安定性を得るために、ユーザーはカスタムドメイン名を優先的に使用することが推奨されます。

この文章では、バケットにカスタムドメイン名を設定し、バケットのデフォルトドメイン名へのアクセスからカスタムドメイン名へのアクセスに切り替える方法について説明します。

ステップ 1：ドメイン名の登録と申告

まず、ユーザーは申告済みのカスタムドメイン名を準備する必要があります。
ドメイン名登録：カスタムドメイン名をお持ちでない場合は、ドメイン名登録 でドメイン名を購入してください。
ドメイン名の申告：お客様のカスタムドメイン名が中国本土のバケットに設定する場合は、ドメイン名を申告する必要があります。

ステップ 2：バケットにカスタムドメイン名を設定する

1. カスタムドメイン名の準備ができたら、COS コンソールにログインし、バケットリストに入り、設定したいバケットを選択します。

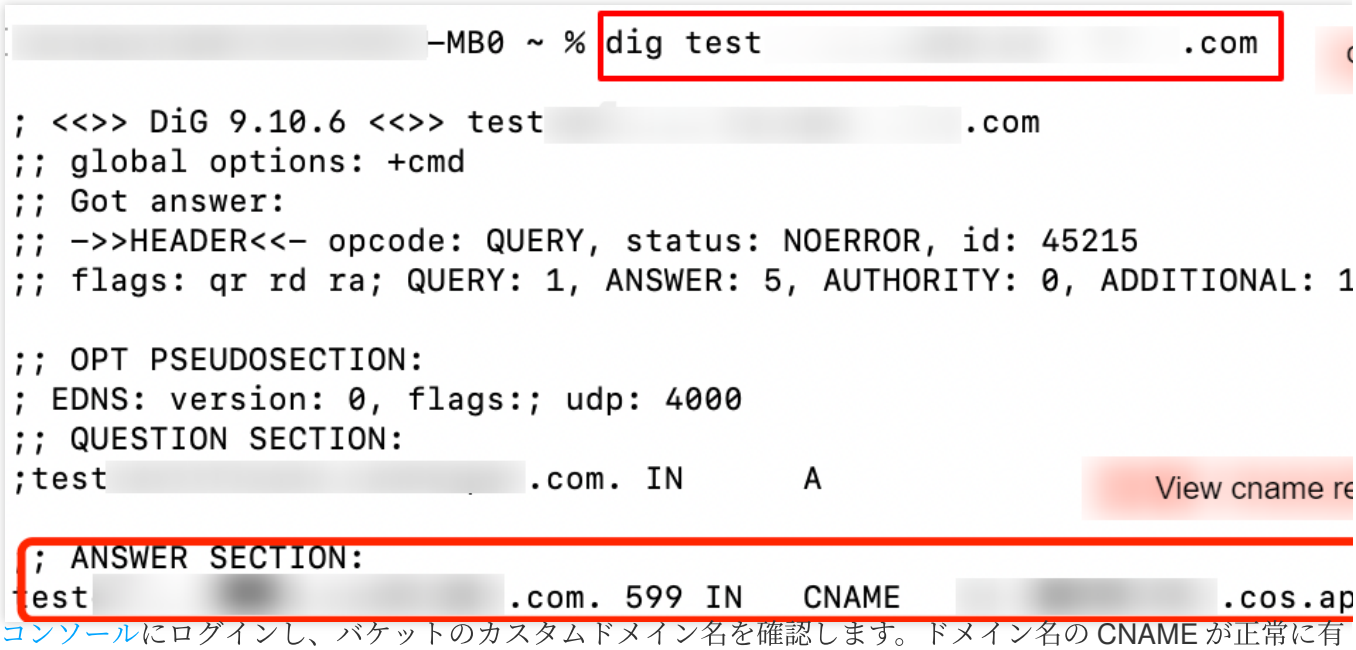
2. バケット詳細ページに移動し、**ドメイン名と伝送管理 > オリジンサーバードメイン名のカスタマイズ**を選択します。
3. **ドメイン名追加**をクリックし、ドメイン名情報を設定します。
ドメイン名：準備したカスタムドメイン名を入力します。
オリジンサーバーのタイプ：以下のタイプに分けられています。
デフォルトオリジンサーバー：カスタムドメイン名をデフォルトオリジンサーバーとして利用したい場合、デフォルトオリジンサーバーを選択してください。
静的オリジンサーバー：カスタムドメイン名を静的ウェブサイトとして利用したい場合は、まずバケットの静的ウェブサイト機能を有効にしてから、静的ウェブサイトのオリジンサーバーを選択してください。
グローバル加速オリジンサーバー：カスタムドメイン名をグローバル加速として利用したい場合は、まずバケットのグローバル加速機能を有効にしてから、グローバル加速のオリジンサーバーを選択してください。
4. HTTPS 証明書を設定します。HTTPS プロトコルでアクセスする必要がある場合は、カスタムドメイン名の証明書を設定する必要があります。
独自の証明書を使用する必要がある場合は、証明書とプライベートキーの内容を指定された入力ボックスに貼り付ける必要があります。
Tencent Cloud によって申請された証明書を使用している場合は、ポップアップウィンドウで現在のアカウントの下にある既存の Tencent Cloud 証明書を直接選択することができます。
5. カスタムドメイン名が設定されたら、CNAME 欄の情報を記録し（例えばbucket-1250000000.cos.ap-beijing.myqcloud.com）、その後のドメイン名解析の設定に使用します。

ステップ 3：ドメイン名解析の設定

Tencent Cloud のドメイン名

ドメイン名の DNS ベンダーが Tencent Cloud の場合、DNS コンソールで CNAME 解析記録を設定できます。

1. DNS コンソールで対応するドメイン名を見つけ、**解析**ボタンをクリックします。
2. **クイック解析追加**をクリックし、ドメイン名の解析記録を追加します。
3. ポップアップウィンドウで**ウェブサイト解析**を選択し、ウェブサイトのアドレスに**ドメイン名マッピング (CNAME)** を選択します。ステップ 2 で記録した CNAME 情報を入力します。例えば、bucket-1250000000.cos.ap-beijing.myqcloud.com です。
4. 解析記録が有効になるには時間がかかり、dig コマンドまたはCOS コンソール で解析が成功したか、有効になったかを確認できます。検証方法は以下の通りです。
コマンドラインウィンドウでコマンド： `dig mydomain.com` を入力し、CNAME 記録が正しく有効になっているかどうかを確認します。（使用する場合は `mydomain.com` をカスタムドメイン名に置き換えてください）



COS コンソールにログインし、バケットのカスタムドメイン名を確認します。ドメイン名の CNAME が正常に有効になっていない場合、対応するプロンプトが表示されます。

その他のベンダーのドメイン名

ドメイン名の DNS ベンダーが Tencent Cloud でない場合は、対応する DNS サービスに移動して CNAME 解析記録を設定する必要があります。

ステップ 4：カスタムドメイン名へのアクセス

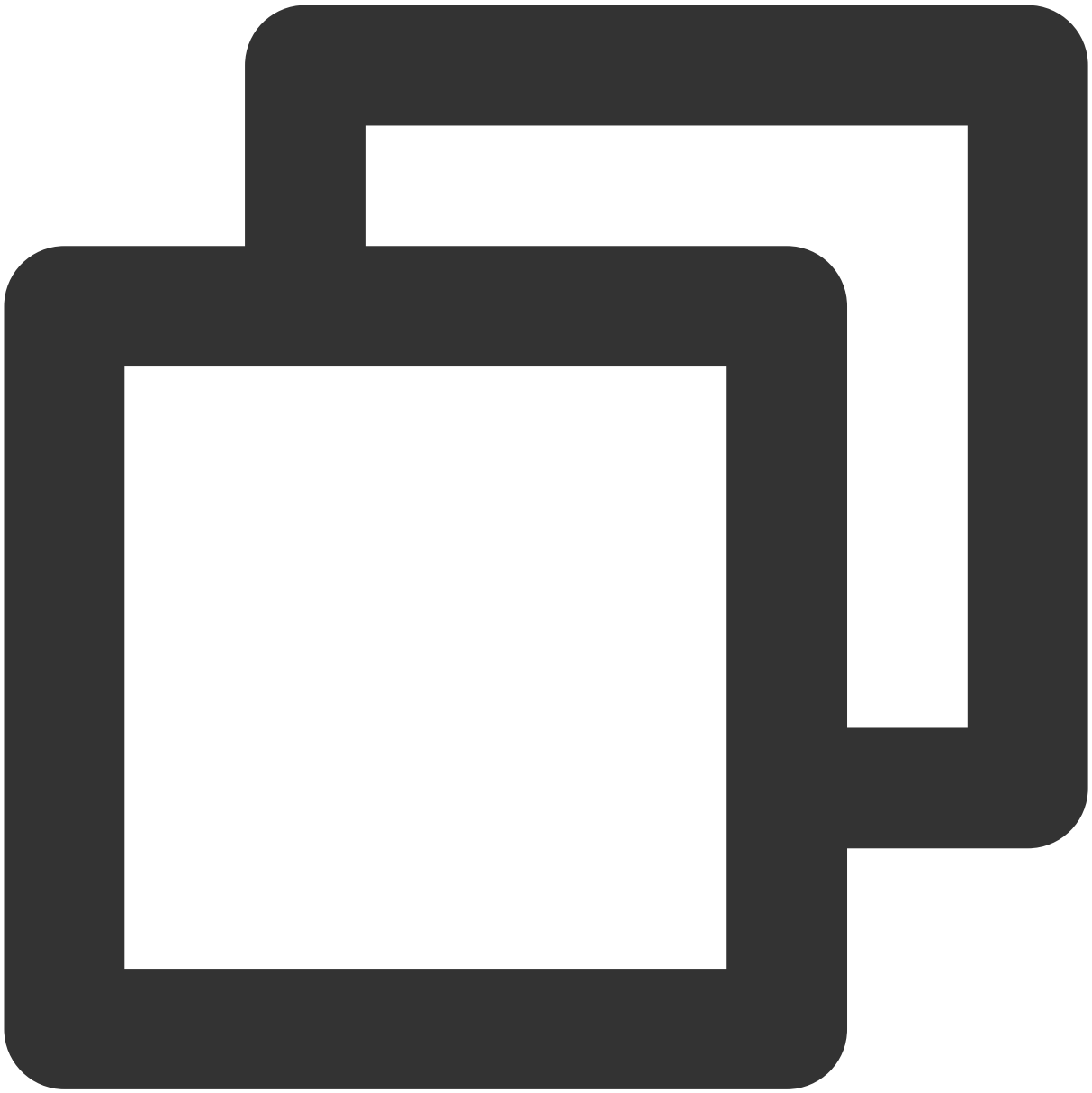
上記のステップを経て、カスタムドメイン名の設定は完了しました。以下では、カスタムドメイン名を使って COS にアクセスする方法について説明します。

オブジェクトのアクセスリンクの表示

1. COS コンソール にログインし、カスタムドメイン名が設定されたバケットを見つけ、クリックして**ファイルリスト**に入ります。オブジェクトを選択し、オブジェクトの詳細に入ります。操作ガイドについては、オブジェクト情報の表示を参照してください。
2. 指定されたドメイン名を**カスタムオリジンサーバードメイン名**に切り替えると、指定されたドメイン名をカスタムソースドメイン名に切り替えると、下部のオブジェクトアドレス、テンポラリリンクがカスタムドメイン名のリンクに切り替わり、パブリックリードオブジェクトへのアクセスにはオブジェクトアドレス（署名なし）を使用し、プライベートリードオブジェクトへのアクセスにはテンポラリリンク（署名あり）を使用することができます。

API アクセスはカスタムドメイン名に切り替える

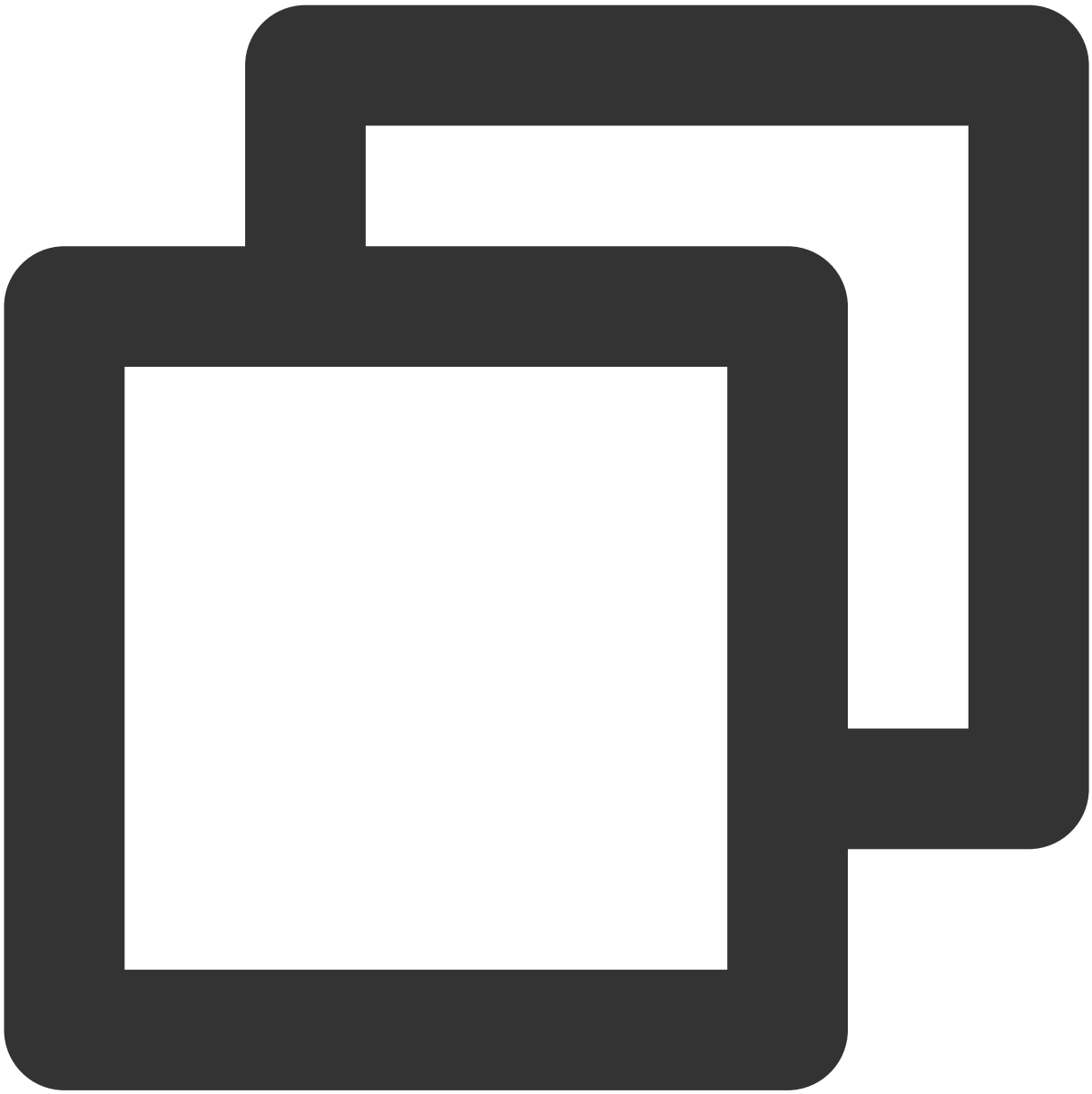
API を直接使用して COS にアクセスするユーザーの場合は、アクセス時にリクエスト Host をカスタムドメイン名に変更するだけで済みます。



```
GET /\\<ObjectKey> HTTP/1.1
Host：<BucketName-APPID>.cos.<Region>.myqcloud.com          #  ユーザーのカスタムドメイ
Date：GMT Date
Authorization：Auth String
```

SDK アクセスはカスタムドメイン名に切り替える

SDK を使用するユーザーの場合は、Client の初期化時に domain パラメータをカスタムドメイン名に設定するだけで済みます。Python SDK を例にすると、コード例は以下のとおりです。



```
domain = 'user-define.example.com' # ユーザーのカスタムドメイン名
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

各言語 COS SDK のカスタムドメイン名切り替えのコード例は、以下のドキュメンテーションを参照してください。

Go SDK

JAVA SDK
Python SDK
PHP SDK
Android SDK
iOS SDK
JS SDK
Node.js SDK
.NET SDK
C++ SDK
C 言語 SDK
ミニプログラム SDK

クロスドメインアクセスの設定

最終更新日：：2024-06-26 10:42:27

同一オリジンポリシー

同一オリジンポリシーは、1つのオリジンからロードしたドキュメントまたはスクリプトと、別のオリジンからのリソースの間でのインタラクションを制限する方法で、潜在的な悪意あるファイルの分離に用いられる重要なセキュリティメカニズムです。同一のプロトコル、同一のドメイン名（またはIP）、および同一のポートは同一のドメインとみなされ、1つのドメイン内のスクリプトはそのドメイン内の権限のみを持つ、すなわちそのドメインのスクリプトは同じドメイン内のリソースのみ読み取り/書き込みを行うことができ、他のドメインのリソースにはアクセスできません。このようなセキュリティ上の制限を同一オリジンポリシーと呼びます。

同一オリジンの定義

2つのページのプロトコル、ドメイン名およびポート（ポートを指定している場合）が同一であれば、同一オリジンとみなされます。下の表に、 `http://www.example.com/dir/page.html` に対する同一オリジンチェックの例を挙げます。

URL	結果	理由
`http://www.example.com/dir2/other.html`	成功	プロトコル、ドメイン名、ポートが同一
`http://www.example.com/dir/inner/another.html`	成功	プロトコル、ドメイン名、ポートが同一
`https://www.example.com/secure.html`	失敗	プロトコルが異なる (HTTPS)
`http://www.example.com:81/dir/etc.html`	失敗	ポートが異なる (81)
`http://news.example.com/dir/other.html`	失敗	ドメイン名が異なる

クロスドメインアクセス

クロスドメインリソース共有（Cross-Origin Resource Sharing，CORS）メカニズムは、単にクロスドメインアクセスとも呼ばれ、Webアプリケーションサーバーにクロスドメインアクセス制御を許可することで、ドメイン間のデータ伝送を安全に実行するものです。CORSにはブラウザとサーバーの両方のサポートが必要です。現在はすべてのブラウザがこの機能をサポートしています。IEブラウザのバージョンはIE10またはそれ以上が必要です。

CORSの通信プロセス全体はすべてブラウザで自動的に完了し、ユーザーが関与する必要がありません。開発者にとっては、CORS通信と同一オリジンのAJAX通信に違いはなく、コードはまったく同じです。ブラウザはAJAXリクエストがクロスオリジンであることを検出すると、自動的にヘッダー情報を追加します。追加のリクエストを1回多く送信する場合もありますが、ユーザーには感知されません。
このため、CORS通信の実現で重要なのはサーバーです。サーバーにCORSインターフェースを実装すれば、クロスオリジン通信が可能となります。

CORSの主なユースケース

ユーザーがブラウザを使用する状況であればCORSを使用できるため、アクセス権限の制御を行うのはサーバーではなくブラウザになります。このため他のクライアントを使用する際はクロスドメインの問題を気にする必要はありません。
CORSの主なアプリケーションは、ブラウザ側でAJAXを使用してCOSのデータへのアクセスまたはデータのアップロード、ダウンロードを実現しており、ユーザー自身のアプリケーションサーバーによる中継を必要としません。COSとAJAX技術を同時に使用しているウェブサイトでは、CORSを使用してCOSとの直接通信を実現することをお勧めします。

COSのCORSに対するサポート

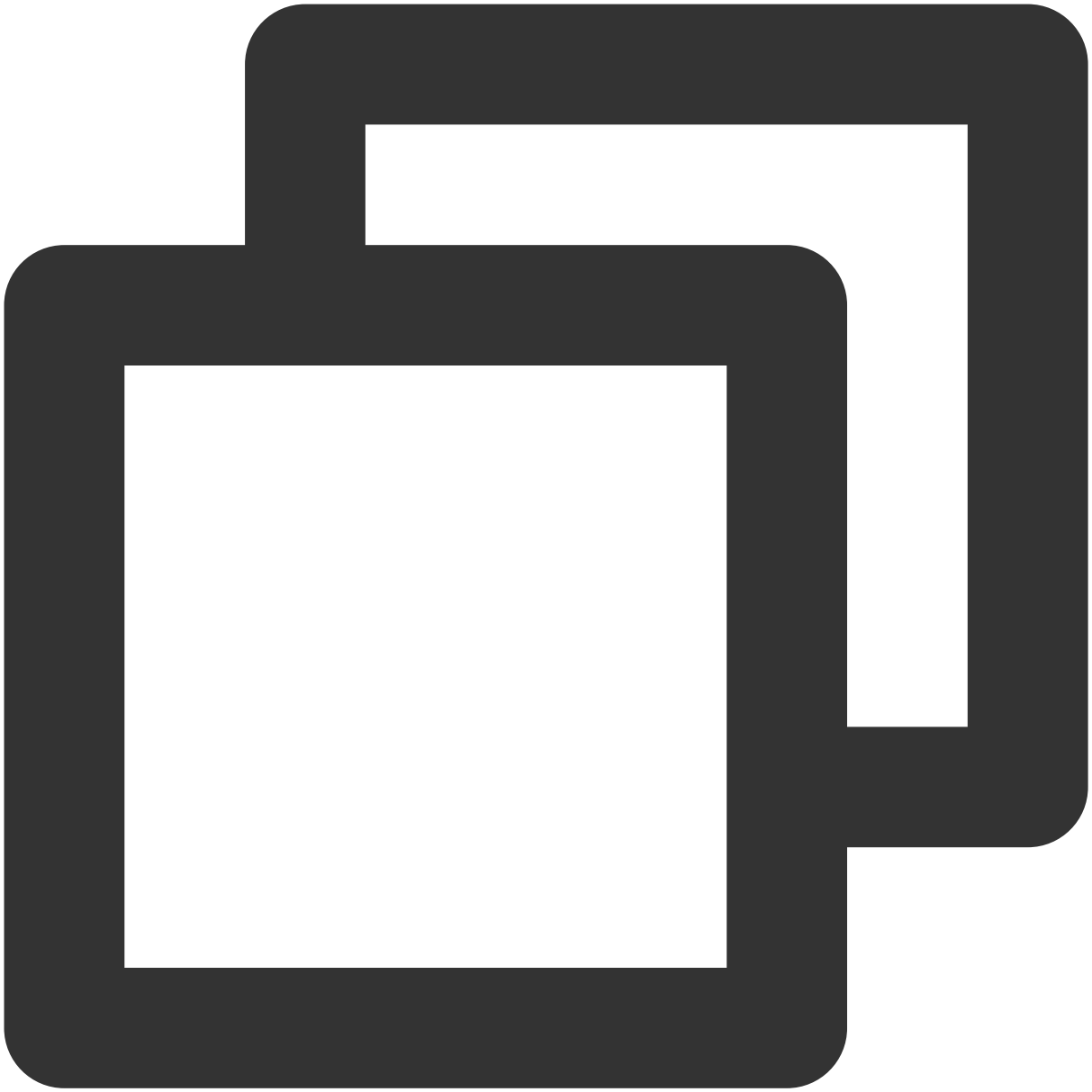
COSはCORSルールの設定をサポートし、対応するクロスドメインリクエストをニーズに応じて許可または拒否します。このCORSルール設定はバケットレベルです。
CORSリクエストの承認の可否およびCOSのID認証などは完全に独立したものであり、すなわちCOSのCORSルールはCORSに関連するHeaderを追加するかどうかの決定だけに用いられるルールです。このリクエストをブロックするかどうかは完全にブラウザによって決定されます。
現在COSのすべてのObject関連インターフェースはCORS関連の検証を行えるようになっています。またMultipart関連のインターフェースもCORS検証をすでに完全にサポートしています。
説明：
同一のブラウザで、 `www.a.com` と `www.b.com` という2つのページからそれぞれ同一のクロスドメインリソースを同時にリクエストされ、 `www.a.com` のリクエストが先にサーバーに到着した場合、サーバーはリソースにAccess-Control-Allow-OriginのHeaderをつけて `www.a.com` のユーザーに返します。このとき `www.b.com` もリクエストを送信しており、ブラウザはCacheした前回のリクエストレスポンスをユーザーに返しますが、Headerの内容がCORSの要求にマッチしないため、 `www.b.com` のリクエストは失敗します。

CORS設定の例

以下の簡単な例によって、AJAXを使用してCOSからデータを取得する設定の手順をご説明します。例で使用しているバケット（Bucket）の権限はパブリック（Public）に設定しています。アクセス権限がプライベートのバケット（Bucket）の場合はリクエスト内に署名を追加する必要があるだけで、その他の設定は同様です。
以下の例で使用しているバケットの名称はcorstest、バケットのアクセス権限はパブリック読み取り/プライベート書き込みです。

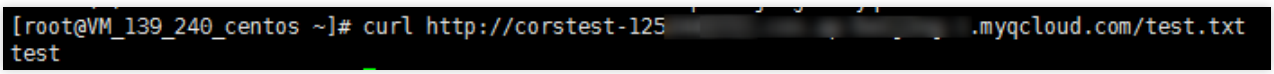
準備作業

1. ファイルに正常にアクセスできることを確認する
test.txtのテキストドキュメントをcorstestにアップロードします。test.txtのアクセスアドレスは `http://corstest-125xxxxxxx.cos.ap-beijing.myqcloud.com/test.txt` です。
curlを使用してこのテキストドキュメントに直接アクセスします。以下のアドレスをご自身のファイルのアドレスに置き換えてください。



```
curl http://corstest-125xxxxxxx.cos.ap-beijing.myqcloud.com/test.txt
```

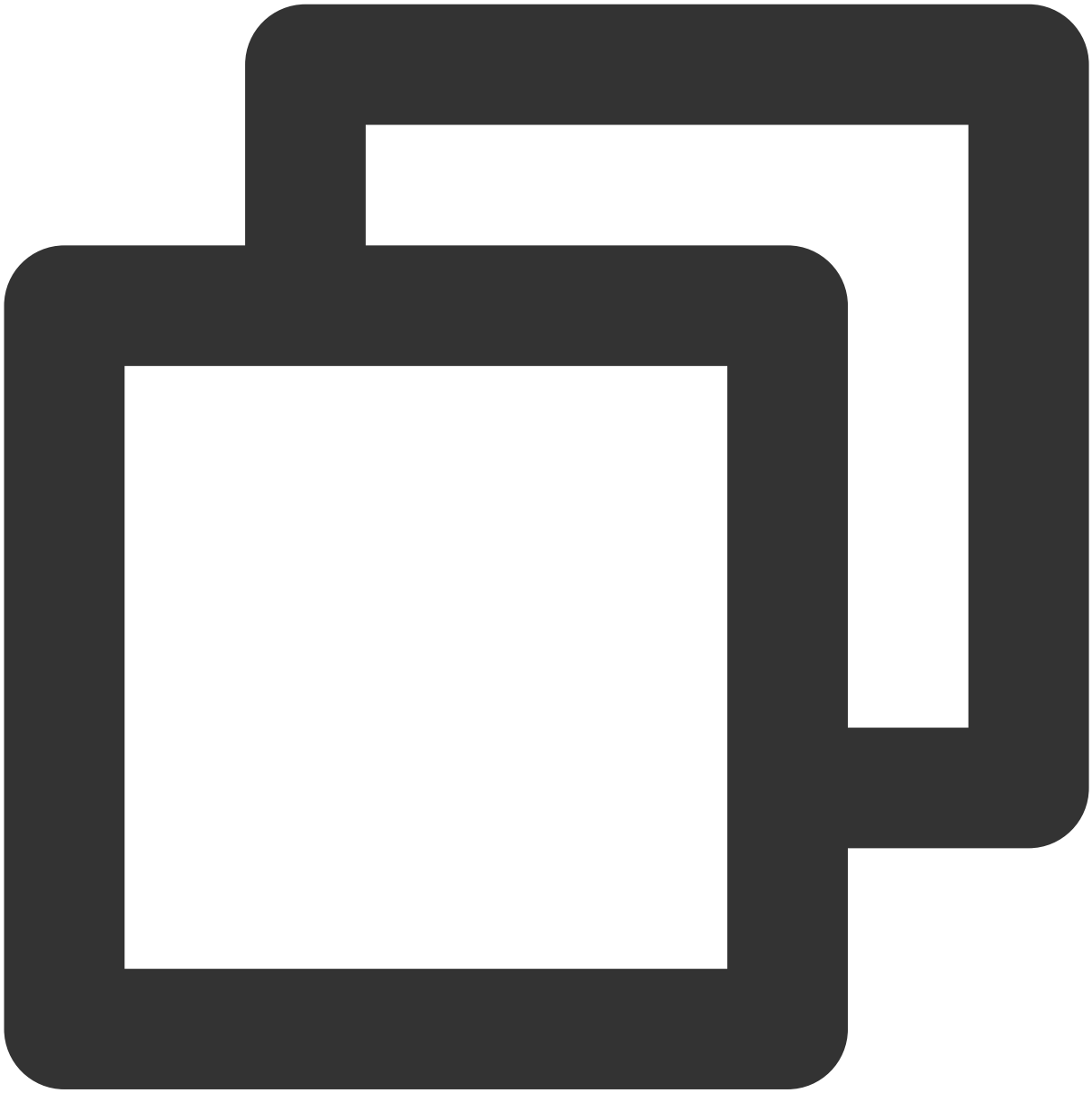
test.txtファイルに返される内容はtestとなります。このドキュメントに正常にアクセス可能であることを表します。



2. **AJAX技術を使用してファイルにアクセスする**

AJAX技術を使用してこのtest.txtファイルに直接アクセスしてみます。

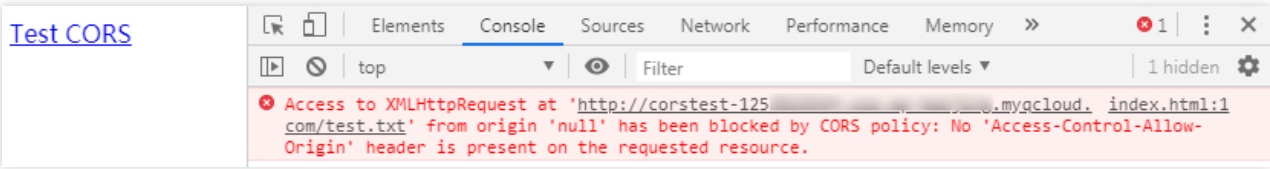
1. 簡単なHTMLファイルを作成します。以下のコードをローカルにコピーしてHTMLファイルとして保存した後、ブラウザで開きます。カスタムヘッダーを設定していないため、このリクエストの事前チェックは不要です。



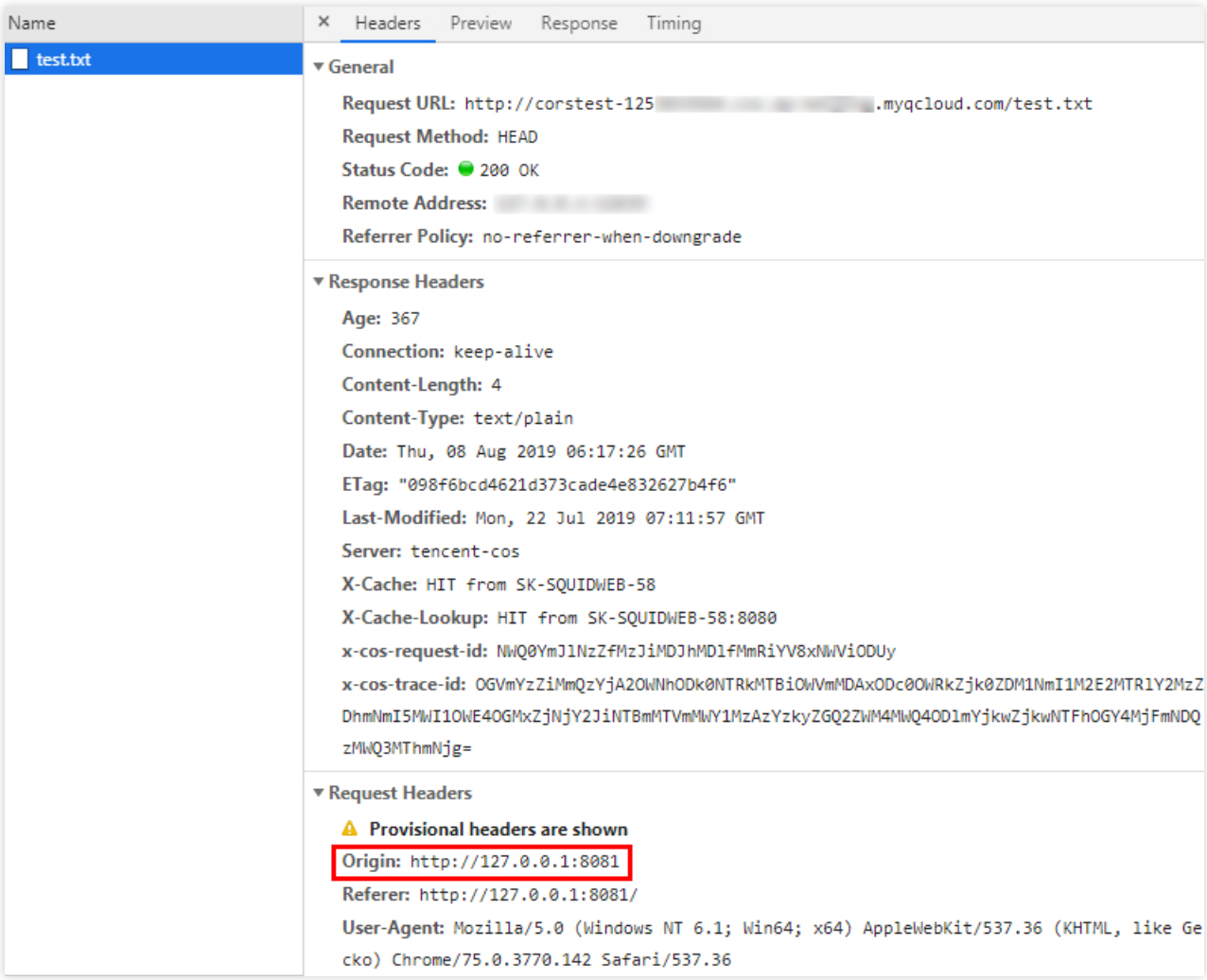
```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
</head>
<body>
```

```
<a href="javascript:test()">Test CORS</a>
<script>
function test() {
    var url = 'http://corstest-125xxxxxxx.cos.ap-beijing.myqcloud.com/test.txt';
    var xhr = new XMLHttpRequest();
    xhr.open('HEAD', url);
    xhr.onload = function () {
        var headers = xhr.getAllResponseHeaders().replace(/\\r\\n/g, '\\n');
        alert('request success, CORS allow.\\n' +
            'url: ' + url + '\\n' +
            'status: ' + xhr.status + '\\n' +
            'headers:\\n' + headers);
    };
    xhr.onerror = function () {
        alert('request error, maybe CORS error.');
    };
    xhr.send();
}
</script>
</body>
</html>
```

2. このHTMLファイルをブラウザで開き、**Test CORS**をクリックしてリクエストを送信すると、次のエラーが表示されます。エラーはアクセス権限がないことを表すもので、その原因はAccess-Control-Allow-Originというヘッダーが見つからないことによるものです。これは明らかに、サーバーにCORSが設定されていないことが原因です。



3. アクセスに失敗後、再びHeader画面で原因を確認します。ブラウザがOriginを含むRequestを送信しており、これがクロスドメインリクエストであることがわかります。



説明：

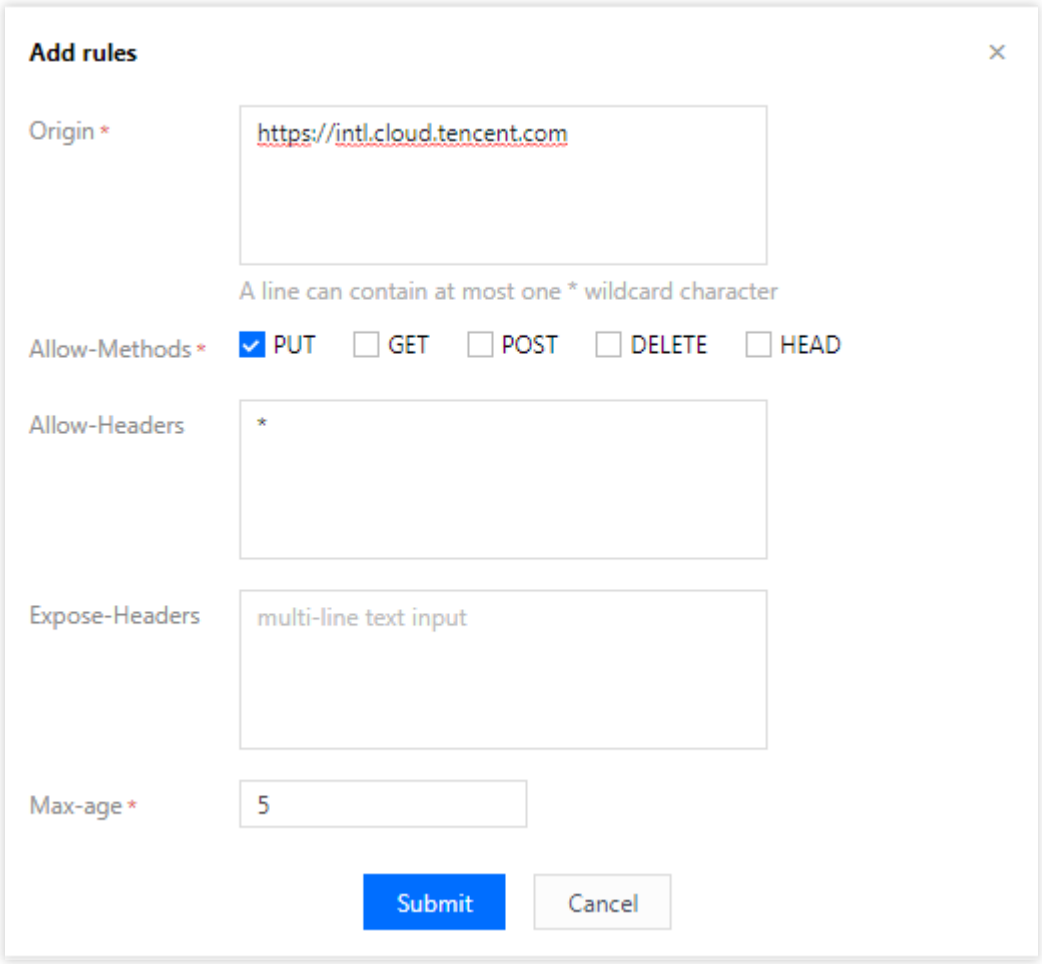
ウェブサイトはサーバー上で構築しており、アドレスは `http://127.0.0.1:8081` です。そのため、Origin は `http://127.0.0.1:8081` です。

CORSの設定

アクセス失敗の原因が確定した後は、バケットに関連するCORSを設定することで上記の問題を解決できます。CORSの設定はCOSコンソールで行うことができ、この例ではCOSコンソールを使用してCORSの設定を完了します。ご自身のCORSの設定が特に複雑なものでない場合も、 コンソールを使用してCORSの設定を行うことをお勧めします。

1. COSコンソールにログインし、**バケットリスト**をクリックして関連のバケットに進み、**セキュリティ管理**タブをクリックして、ドロップダウンしたページから「クロスドメインアクセスCORS設定」を見つけます。

2. **ルールの追加**をクリックし、1つ目のルールを追加します。最も寛容な設定を使用する場合は次のようになります。

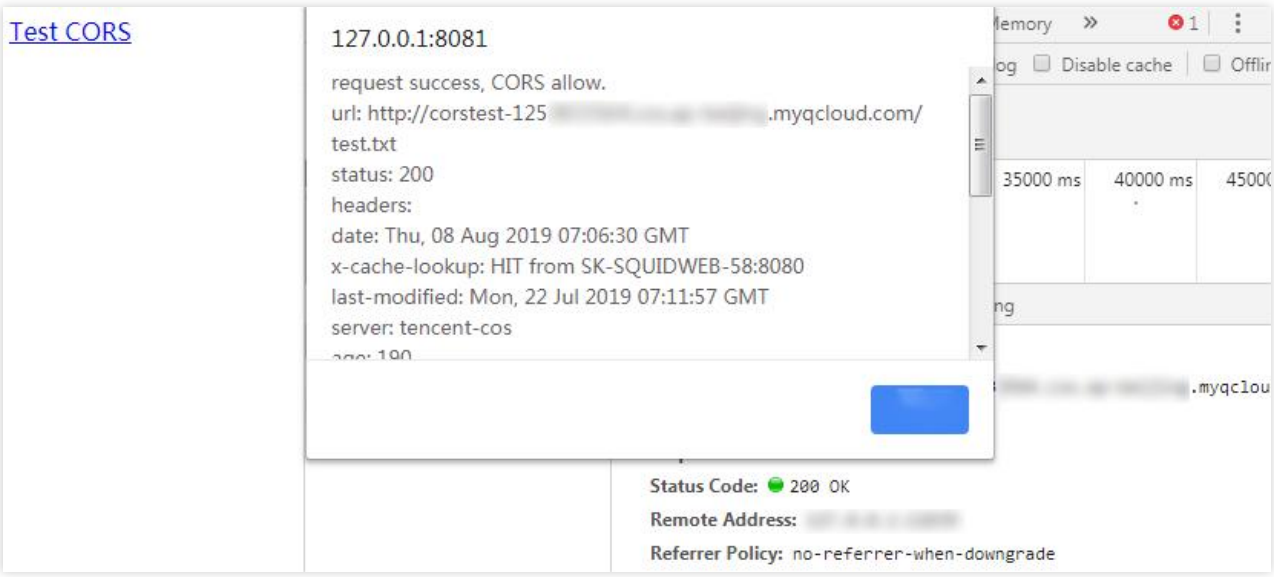


注意：

CORS設定は個々のルールで構成されたものであり、1つ目のルールから順にマッチングしていき、最初にマッチしたルールに準じます。

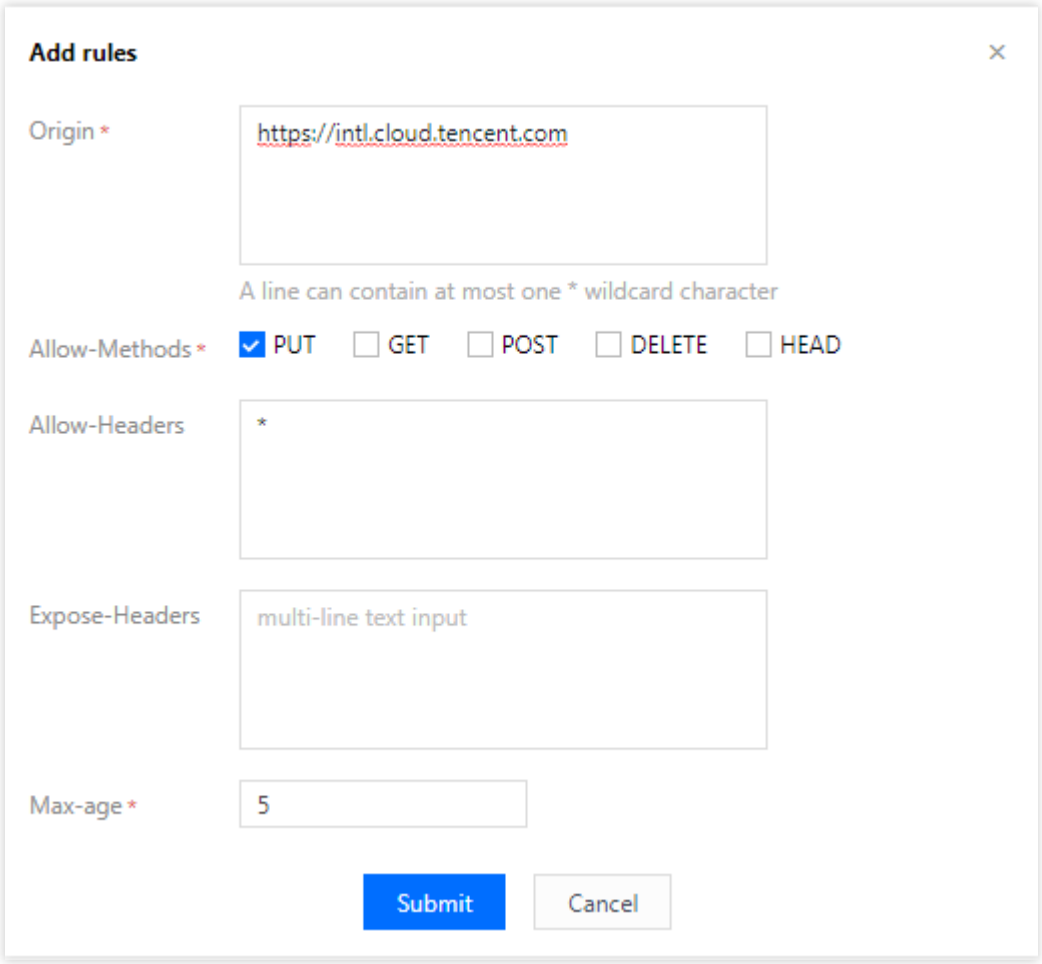
検証結果

設定完了後、test.txtテキストファイルへのアクセスを再度試みます。結果が次のとおりであれば、リクエストは正常なアクセスが可能です。



トラブルシューティングと推奨事項

クロスドメインによるアクセスの問題を解消したい場合は、CORS設定を上記の最も寛容な設定にすることができます。この設定はすべてのクロスドメインリクエストを許可するものです。この設定でもエラーが発生する場合は、エラーがCORSではなく別の部分で生じていることを意味しています。
最も寛容な設定のほか、さらに精密な制御メカニズムを設定することでピンポイントの制御を行うことも可能です。例えば、この例では、次のような最小の設定を使用してマッチングに成功することができます。



このため、大部分のケースでは、ご自身のユースケースに応じた最小の設定を使用して安全性を保証することを推奨します。

CORS設定項目の説明

CORSの設定項目には次のものがあります。

オリジンOrigin

クロスドメインリクエストを許可するオリジンです。
複数のオリジンを同時に指定できますが、1行に入力できるのは1つだけです。
設定は `*` をサポートしており、これはすべてのドメイン名が許可されることを意味しますが、推奨しません。
`http://www.abc.com` 形式など、単一のドメイン名をサポートしています。
`http://*.abc.com` など、第2レベルの汎用ドメイン名もサポートしていますが、 `*` 記号は1行に1つだけとします。

プロトコル名のHTTPまたはHTTPSを省略しないように注意します。ポートがデフォルトの80でない場合は、ポートも含める必要があります。

操作Methods

許可するクロスドメインリクエストメソッドを列挙します（1つまたは複数）。
例：GET、PUT、POST、DELETE、HEAD

Allow-Header

許可するクロスドメインリクエストHeaderです。
複数のオリジンを同時に指定できますが、1行に入力できるのは1つだけです。
Headerは省略されやすいので、特別な要件がなければ `*` に設定することをお勧めします。これはすべて許可することを意味します。
大文字と小文字は区別されません。
Access-Control-Request-Headersで指定された各Headerには、Allowed-Headerに対応する項目が必要です。

Expose-Header

ブラウザに公開されているHeaderのリスト、すなわちユーザーがアプリケーションからアクセスするレスポンスヘッダーです（例えばあるJavascriptのXMLHttpRequestオブジェクトなど）。
具体的な設定はアプリケーションのニーズに応じて決定する必要があります。デフォルトではEtagの入力を推奨します。
ワイルドカードは使用できず、大文字と小文字は区別されません。1行に入力できるのは1つだけです。

タイムアウトMax-Age

ブラウザが特定のリソースのプリフライトリクエスト（OPTIONSリクエスト）から返される結果をキャッシュする時間であり、単位は秒です。特別な場合を除き、60秒などと長めに設定することができます。この項目はオプション設定項目です。

COS静的ウェブサイト機能を使用したフロントエンドSPAの構築

最終更新日：：2024-06-26 10:42:27

single-page applicationとは何ですか。

single-page application（SPA）とは、ネットワークアプリケーションプログラムまたはウェブサイトのモデルの一種であり、従来のように新しいページ全体をサーバーからリロードするのではなく、現在のページを動的にリライトすることでユーザーとのインタラクションを行うものです。この方式では、ページ間の切り替えによってユーザーエクスペリエンスが中断されることがなく、アプリケーションをデスクトップアプリケーションにより近いものにすることができます。single-page applicationでは、必要なコード（HTML、JavaScript、CSS）はすべて単一のページからロードして検索するか、または必要に応じて（通常はユーザーの操作に対する応答として）適切なリソースを動的にロードし、ページに追加します。
現在のフロントエンド開発領域でよくみられるSPA開発フレームワークには、React、Vue、Angularなどがあります。
ここでは、現在比較的よく用いられている2つのフレームワークを使用し、**Tencent CloudのCloud Object Storage（COS）の静的ウェブサイト**機能を使用してオンラインで使用できるSPAをスピーディーに構築する方法について、段階を追ってご説明するとともに、よくあるご質問とその対処方法についても記載します。

準備作業

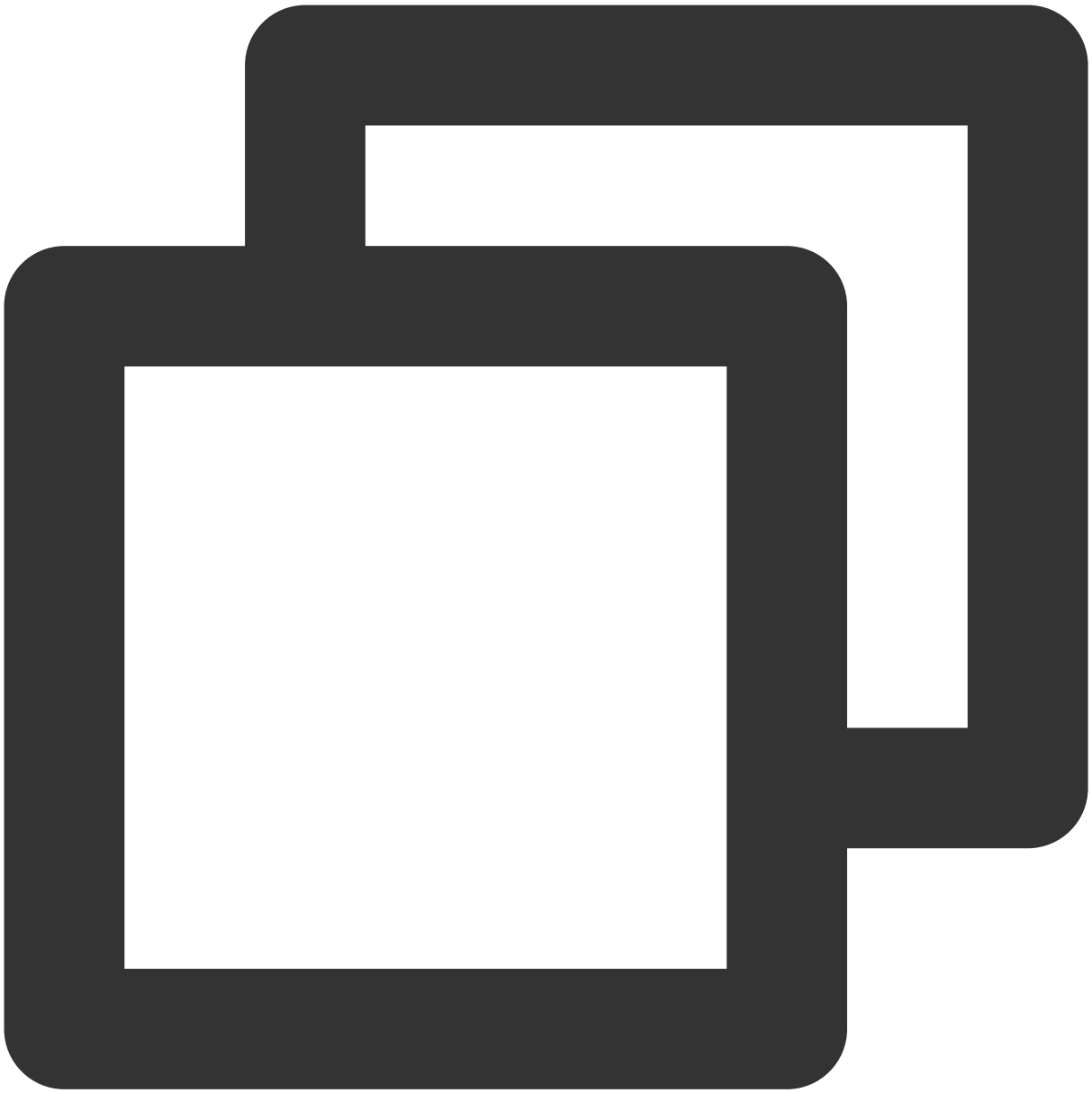
1. Node.js 環境をインストールします。
2. Tencent Cloudアカウントを登録し、実名認証を完了して、Tencent Cloud COSコンソールに正常にログインできることを確認します。
3. バケットを作成します（テストに便利なように、権限を**パブリック読み取り/プライベート書き込み**に設定します）。

フロントエンドコードを作成する

注意：
すでにご自身でコードを実装済みの場合は、バケットの静的ウェブサイト設定を有効化するの手順に直接ジャンプして確認することができます。

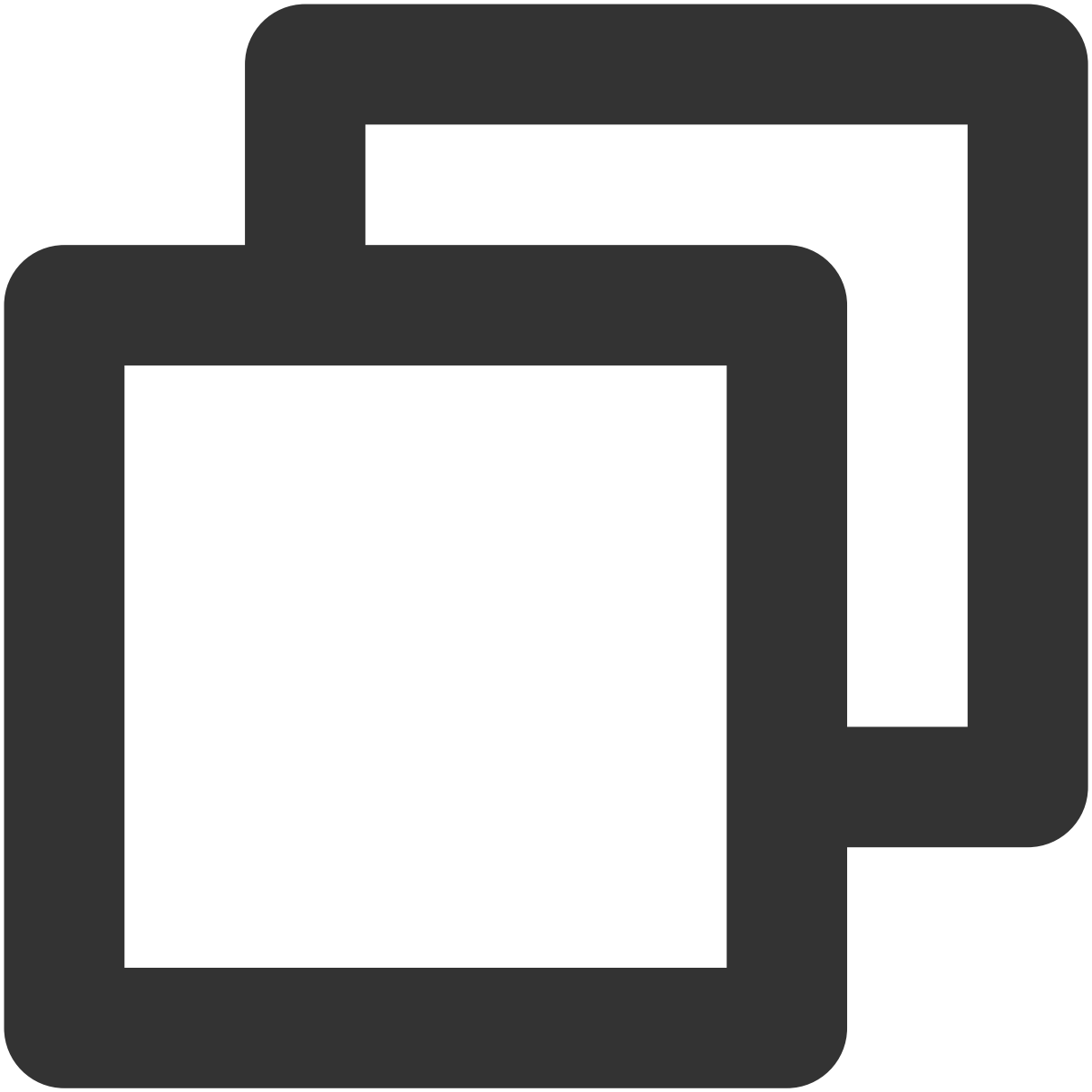
Vueを使用してSPAをスピーディーに構築する

1. 次のコマンドを実行し、vue-cliをインストールします。



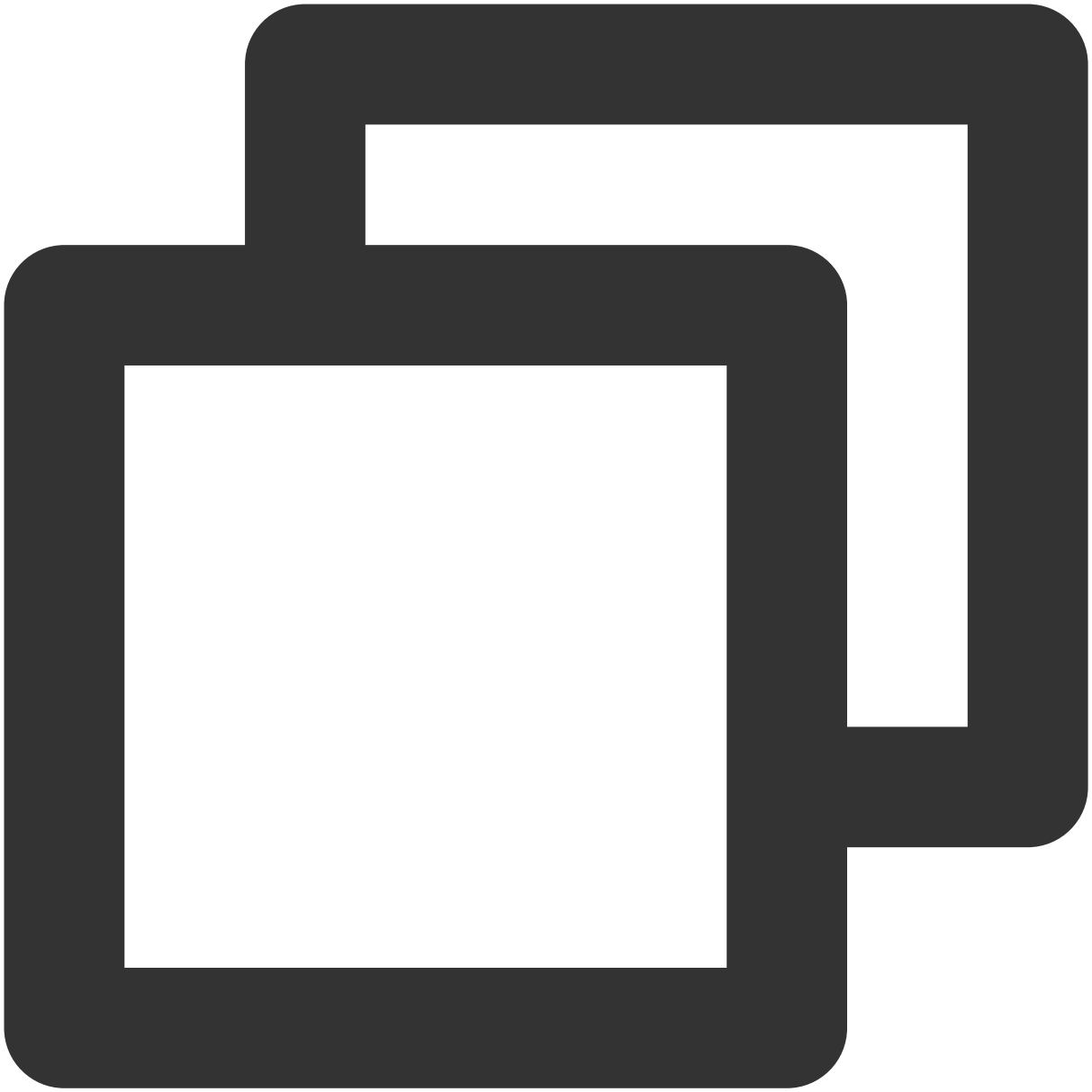
```
npm install -g @vue/cli
```

2. 次のコマンドを実行し、vue-cliを使用してvueプロジェクトをクイック生成します。公式ドキュメントをご参照ください。



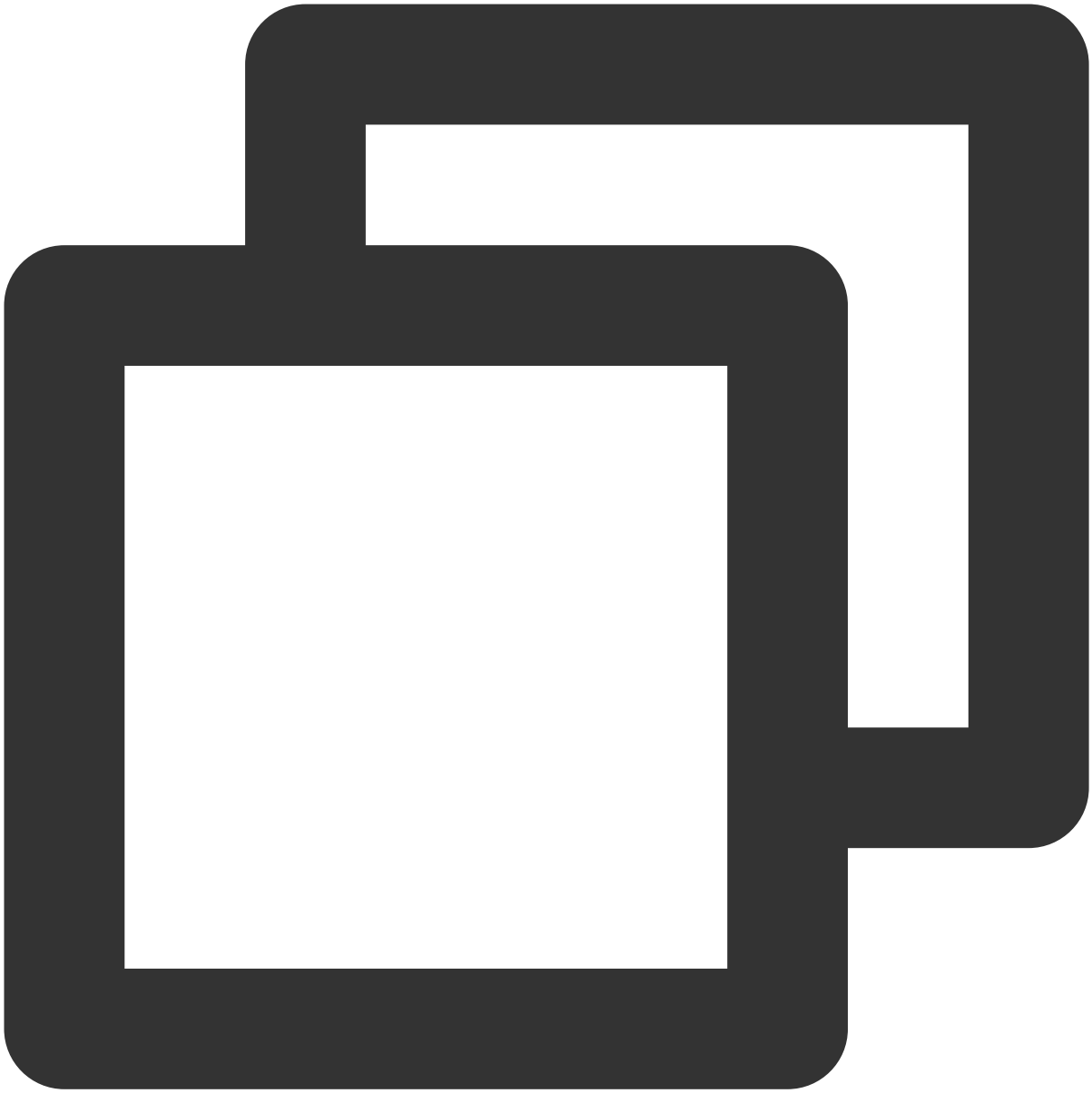
```
vue create vue-spa
```

3. 次のコマンドを実行し、プロジェクトのルートディレクトリにvue-routerをインストールします。



```
npm install vue-router -S  (Vue2.x)
```

または



```
npm install vue-router@4 -S  (Vue3.x)
```

4. プロジェクト内のmain.jsおよびApp.vueファイルを変更します。

main.jsは以下のようにします。

```
import { createApp } from 'vue'
import { createRouter, createWebHistory } from 'vue-router'
import App from './App.vue'
import Home from './components/Home.vue'
import Foo from './components/Foo.vue'
import Bar from './components/Bar.vue'
import Default from './components/404.vue'

const routes = [
    { path: '/', component: Home },
    { path: '/foo', component: Foo },
    { path: '/bar', component: Bar },
    { path: '/:w+', component: Default }
]

const router = createRouter({
    history: createWebHistory(),
    routes
})

const app = createApp(App)
app.use(router)
app.mount('#app')
```

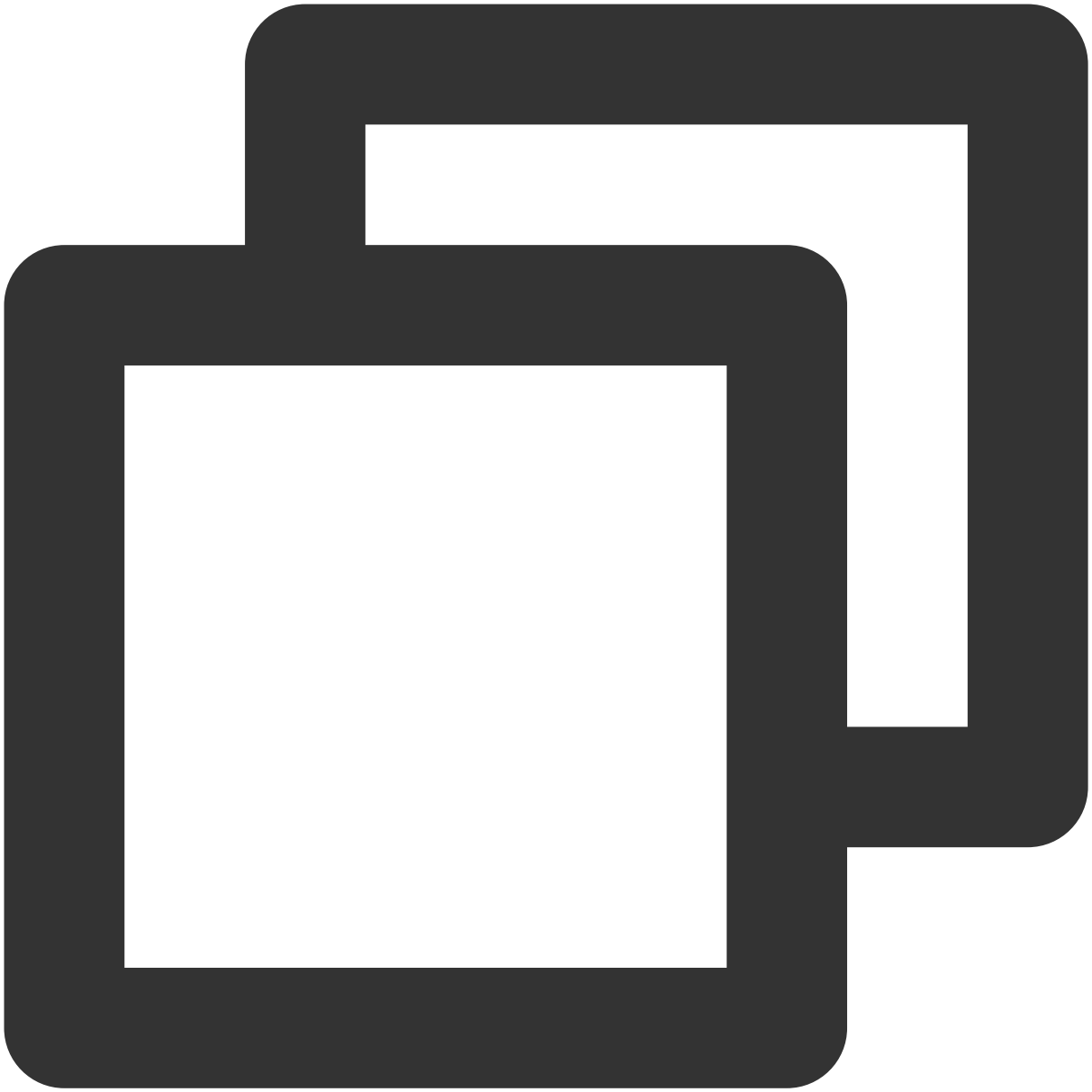
App.vueは主にコンポーネントのtemplateを以下のように変更します。

```
<template>
  <div>
    <ul>
      <li>
        <router-link to="/">Home</router-link>
      </li>
      <li>
        <router-link to="/foo">Foo</router-link>
      </li>
      <li>
        <router-link to="/bar">Bar</router-link>
      </li>
    </ul>
    <img alt="Vue logo" src="./assets/logo.png">
    <router-view></router-view>
  </div>
</template>
```

説明：

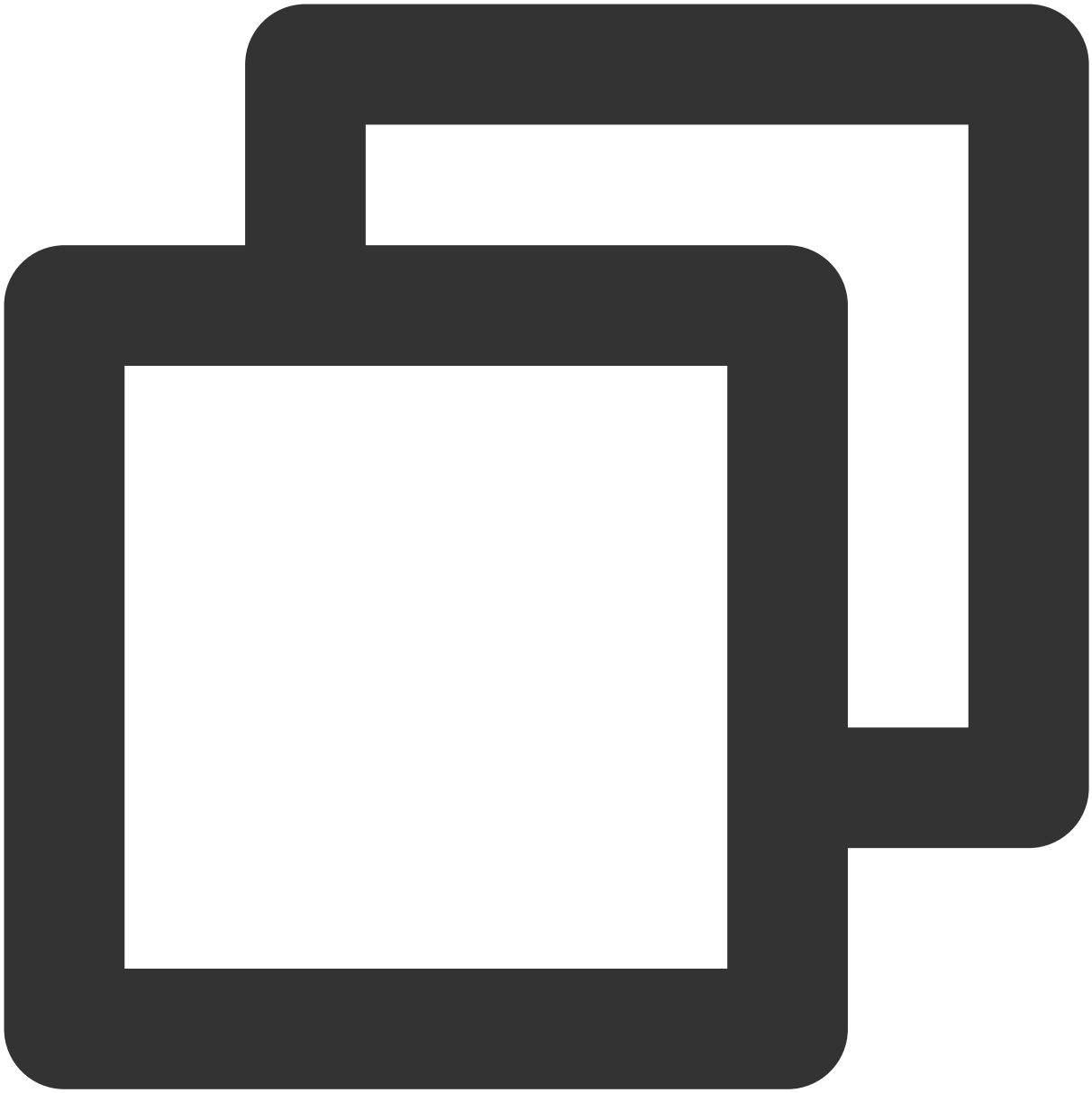
スペースの都合上、ここでは一部の重要コードのみを抜粋します。完全なコードはここをクリックしてダウンロードできます。

5. コードを変更した後、次のコマンドを実行してローカルプレビューを行います。



```
npm run serve
```

6. デバッグを行い、プレビューで誤りがないか確認した後、次のコマンドを実行して本番環境コードをパッケージ化します。

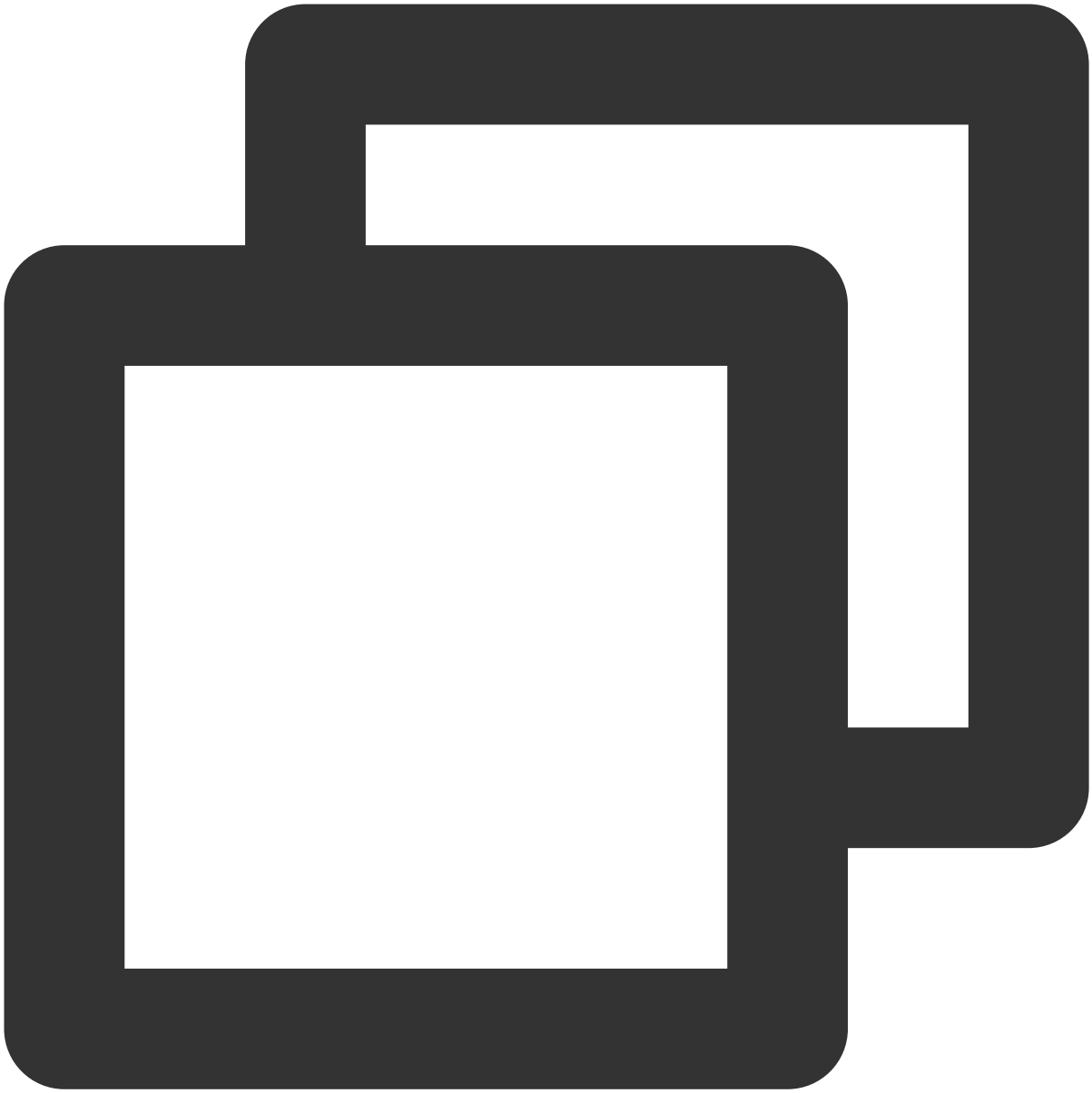


```
npm run build
```

この時点で、プロジェクトのルートディレクトリ下にdistディレクトリが生成されます。これでVueのプログラムコードの準備は完了です。

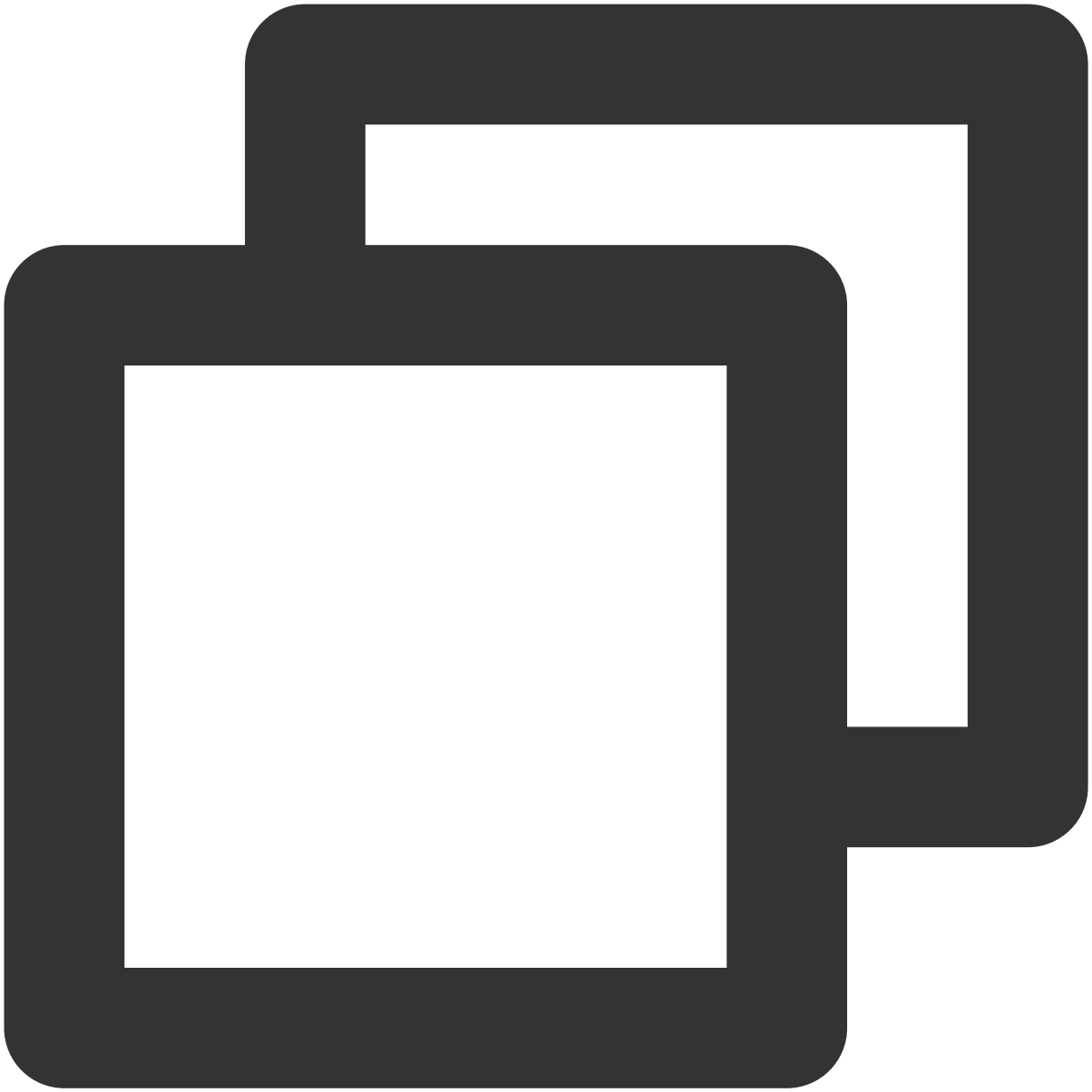
Reactを使用してSPAをスピーディーに構築する

1. 次のコマンドを実行し、create-react-appをインストールします。



```
npm install -g create-react-app
```

2. create-react-appを使用してreactプロジェクトをクイック生成します。公式ドキュメントをご参照ください。
3. 次のコマンドを実行し、プロジェクトのルートディレクトリにreact-router-domをインストールします。



```
npm install react-router-dom -S
```

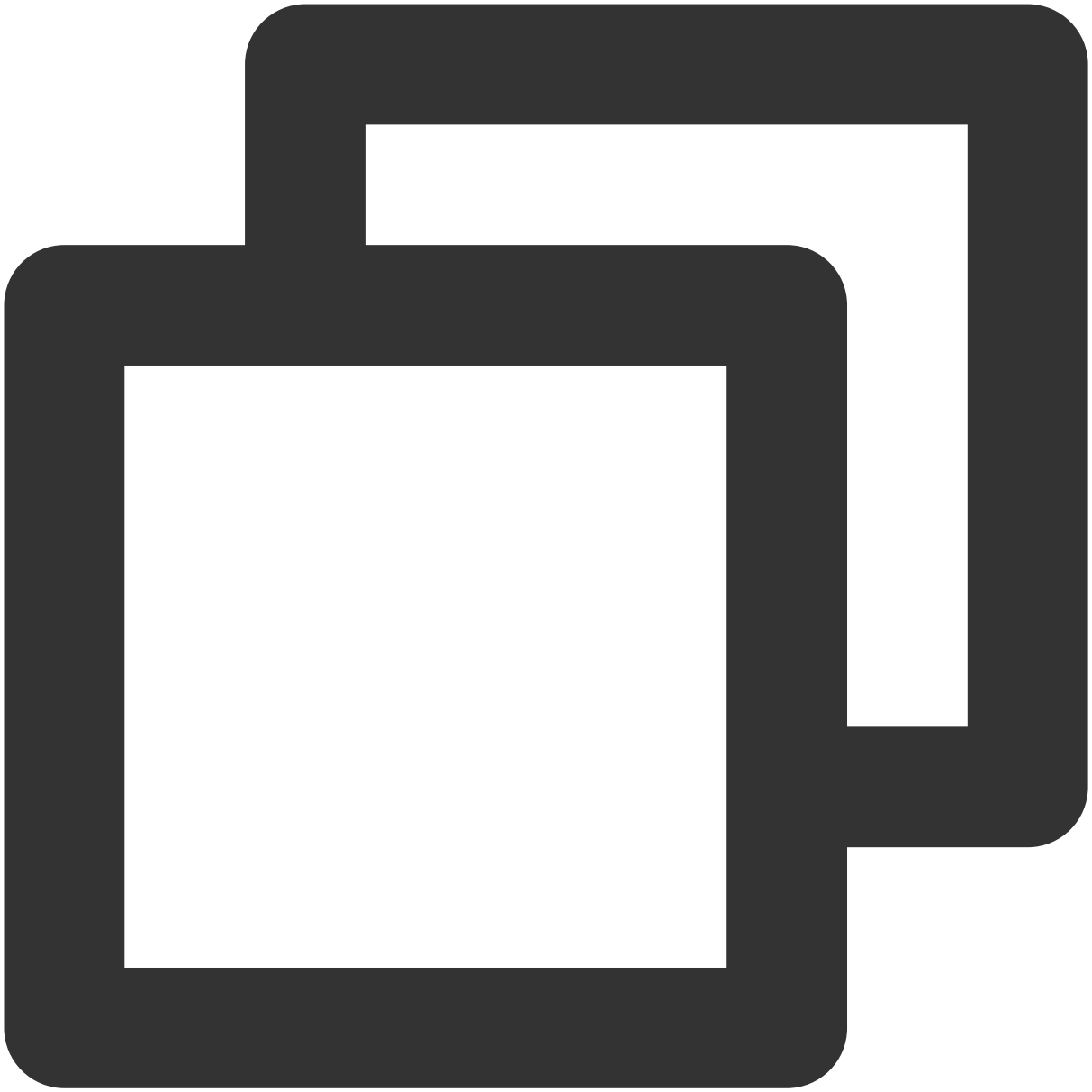
4. プロジェクト内のApp.jsファイルを変更します。

```
import React from 'react';
import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom'
import './App.css';

function Home() {
  return <h2>Home</h2>;
}

function About() {
  return <h2>About</h2>;
}

function NoMatch() {
  return <h2>404 Page</h2>
}

function App() {
  return (
    <Router>
      <div className="App">
        <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
            <li>
              <Link to="/about">About</Link>
            </li>
          </ul>
        </nav>
        <Switch>
          <Route exact path="/">
            <Home />
          </Route>
          <Route path="/about">
            <About />
          </Route>
          <Route path="*">
            <NoMatch />
          </Route>
        </Switch>
      </div>
    </Router>
  );
}

export default App;
```

説明：

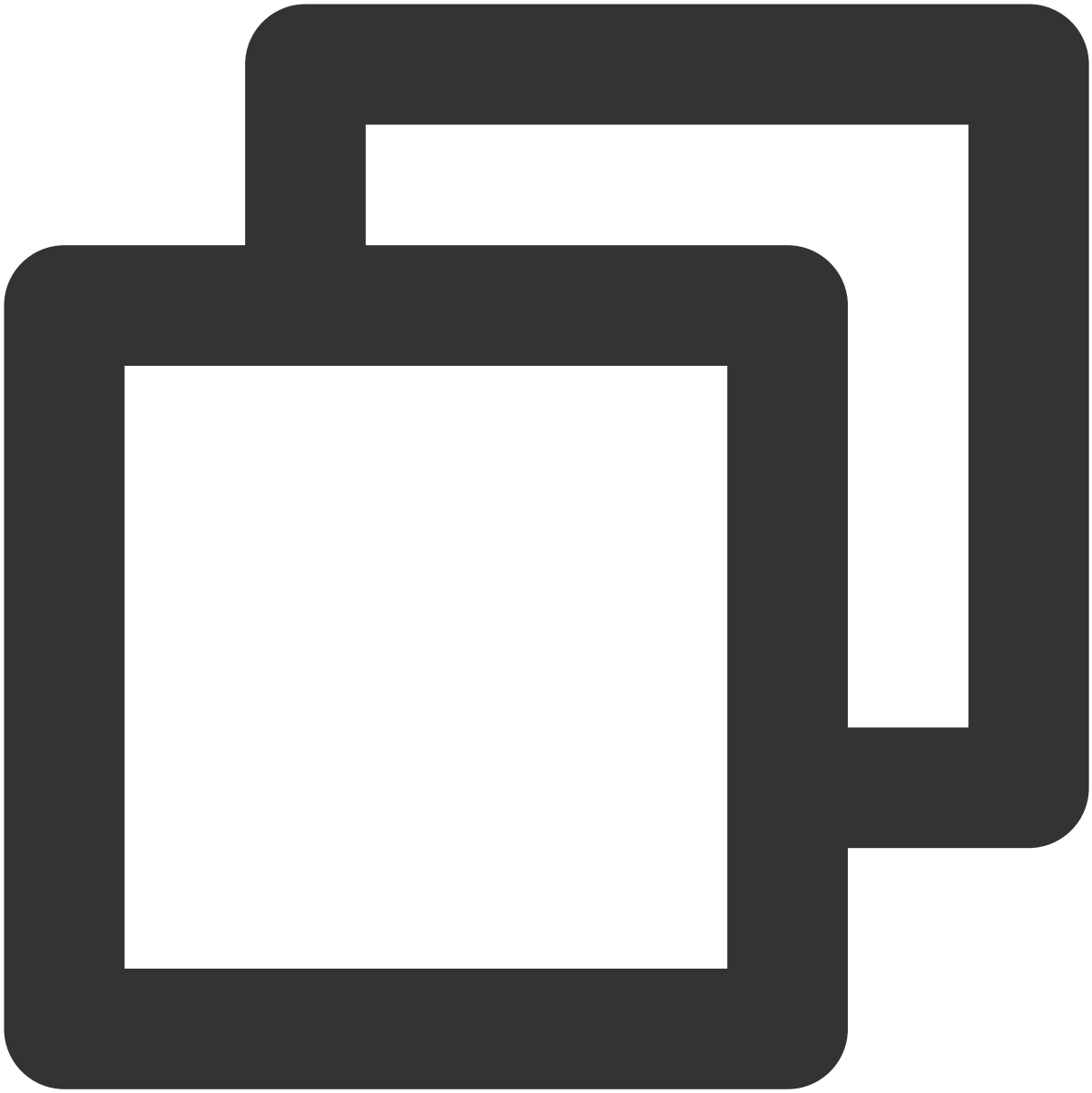
スペースの都合上、ここでは一部の重要コードのみを抜粋します。完全なコードはここをクリックしてダウンロードできます。

5. コードを変更した後、次のコマンドを実行してローカルプレビューを行います。



```
npm run start
```

6. デバッグを行い、プレビューで誤りがないか確認した後、次のコマンドを実行して本番環境コードをパッケージ化します。



```
npm run build
```

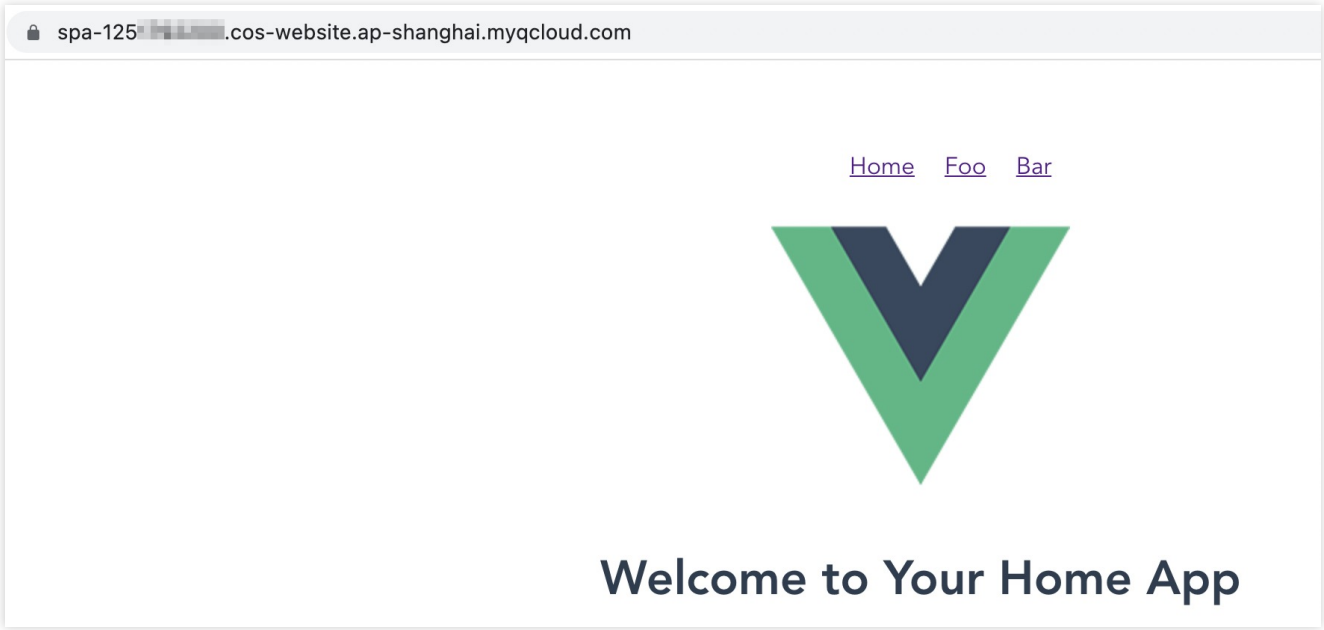
この時点で、プロジェクトのルートディレクトリ下にbuildディレクトリが生成されます。これでReactのプログラムコードの準備は完了です。

バケットの静的ウェブサイト設定を有効化する

1. 作成済みのバケットの詳細ページに進み、左側ナビゲーションバーで、**基本設定** > **静的ウェブサイト**をクリックします。
2. 静的ウェブサイト管理ページで、下図を参照して設定を行います。操作の詳細については、静的ウェブサイトの設定をご参照ください。

COSにデプロイする

1. 静的ウェブサイトを設定済みのバケットを見つけ、ファイルリストページに進みます。
2. コンパイルディレクトリ（Vueはデフォルトではdistディレクトリ、reactはデフォルトではbuildディレクトリ）下のすべてのファイルをバケットのルートディレクトリ下にアップロードします。操作の詳細については、オブジェクトのアップロードをご参照ください。
3. バケットの静的ウェブサイトドメイン名にアクセスします（下図のアクセスノード参照）。
これで、デプロイが完了したアプリケーションのメインページを確認できます。ここではVueアプリケーションの例を挙げます。



4. ルーティング（Home、Foo、Bar）を切り替えてページを更新してみて、予測したとおり（ルーティング下で更新しても404エラーが表示されない）になるかどうかを検証します。

よくあるご質問

静的ウェブサイトのデフォルトドメイン名を使用したくない場合はどうすればよいですか。自分のドメイン名を使用できますか。

上記で使用したデフォルトの静的ウェブサイトドメイン名以外に、COSはカスタムCDNアクセラレーションドメイン名、カスタムオリジンサーバードメイン名の設定もサポートしています。具体的には、ドメイン名管理の概要のドキュメントを参照して設定できます。設定が成功すると、ご自身の使用したいドメイン名でアプリケーションにアクセスできるようになります。
CDNアクセラレーションドメイン名の設定を選択した場合は、更新後のデータを入手しやすいよう、CDNノードキャッシュ期限設定を行っておくことにご注意ください。

アプリケーションをデプロイ後、ルーティングを切り替えてレンダリングに成功しましたが、ページを更新すると404エラーが表示されました。何が原因ですか。

原因は、静的ウェブサイト設定の際に設定が失われたか、または**エラードキュメント**を誤って設定したことによる可能性があります。この章の冒頭にある標準設定のスクリーンキャプチャをもう一度ご参照ください。エラードキュメントとインデックスドキュメントはどちらも `index.html` に設定されています。
SPAの特性により、どのような状況でもアプリケーションポータル（通常は `index.html` ）にアクセスできることを保証する必要があります。そうしなければ、その後のルーティングの一連の内部ロジックがトリガーされないからです。

ルーティングの切り替え後、ページは正常に表示されますが、HTTPステータスコードが404のままです。どうすれば正常に200が返されますか。

ここでの原因は、静的ウェブサイト設定の際に**エラードキュメントのレスポンスコード**を設定していなかったことによるものです。冒頭の設定スクリーンキャプチャをご参照の上、200に設定することで解決できます。

エラードキュメントを設定後、アクセスエラーのパスに404を表示する機能も必要なのですが、どうすればよいですか。

ここでは、フロントエンドコード内で404ロジックを実装することを推奨します。ルーティング設定の一番下に最下層のマッチングルールを設定し、前のすべてのルールがマッチングに失敗した場合は404コンポーネントをレンダリングするようにします。コンポーネントの内容はニーズに合わせてご自身で設計可能です。具体的には、ここでご提供するコードdemo内のルーティング設定の最後の設定をご参照ください。

アクセスしたページで403 Access Deniedエラーが表示されましたが、何が原因ですか。

原因は、バケットの権限が**プライベート読み取り/書き込み**に設定されていることによる可能性があります。**パブリック読み取り/プライベート書き込み**に変更すると解決できます。
また、CDNアクセラレーションドメイン名を使用して**プライベート読み取り/書き込み**のバケットにアクセスする場合、**back-to-origin認証**を有効化しておかなければ、CDNサービスにCOSリソースへの正常なアクセス権限を付与できないことにも注意が必要です。

画像処理の実践
画像テキスト混合ウォーターマークの実践

最終更新日：：2024-06-26 10:42:27

ユースケース

Cloud Infinite基本画像処理のウォーターマーク機能は画像上への画像ウォーターマークまたはテキストウォーターマークの追加をサポートしています。ただし、現実の業務シーンにおいては、1枚の画像上に、固定のロゴウォーターマークと動的に変化するテキストウォーターマーク（ユーザー名など）の両方がある状況がしばしば発生します。上記の状況について、参考までに以下の方法をご紹介します。実際の業務シーンに応じて、適切な統合方法を選択することができます。

方法の優位性比較

方法	メリット	デメリット
方法1	統合が簡単で、手軽に実装できます。	ウォーターマークのサイズを画像の大きさに合わせて動的に変えることができず、タイリング操作を行うことができません。
方法2	画像サイズがよく変わる場合に、方法1に比べてよりアダプティブな効果が得られます。	統合のフローがより煩雑であり、2回分の処理料金がかかります。

操作方法

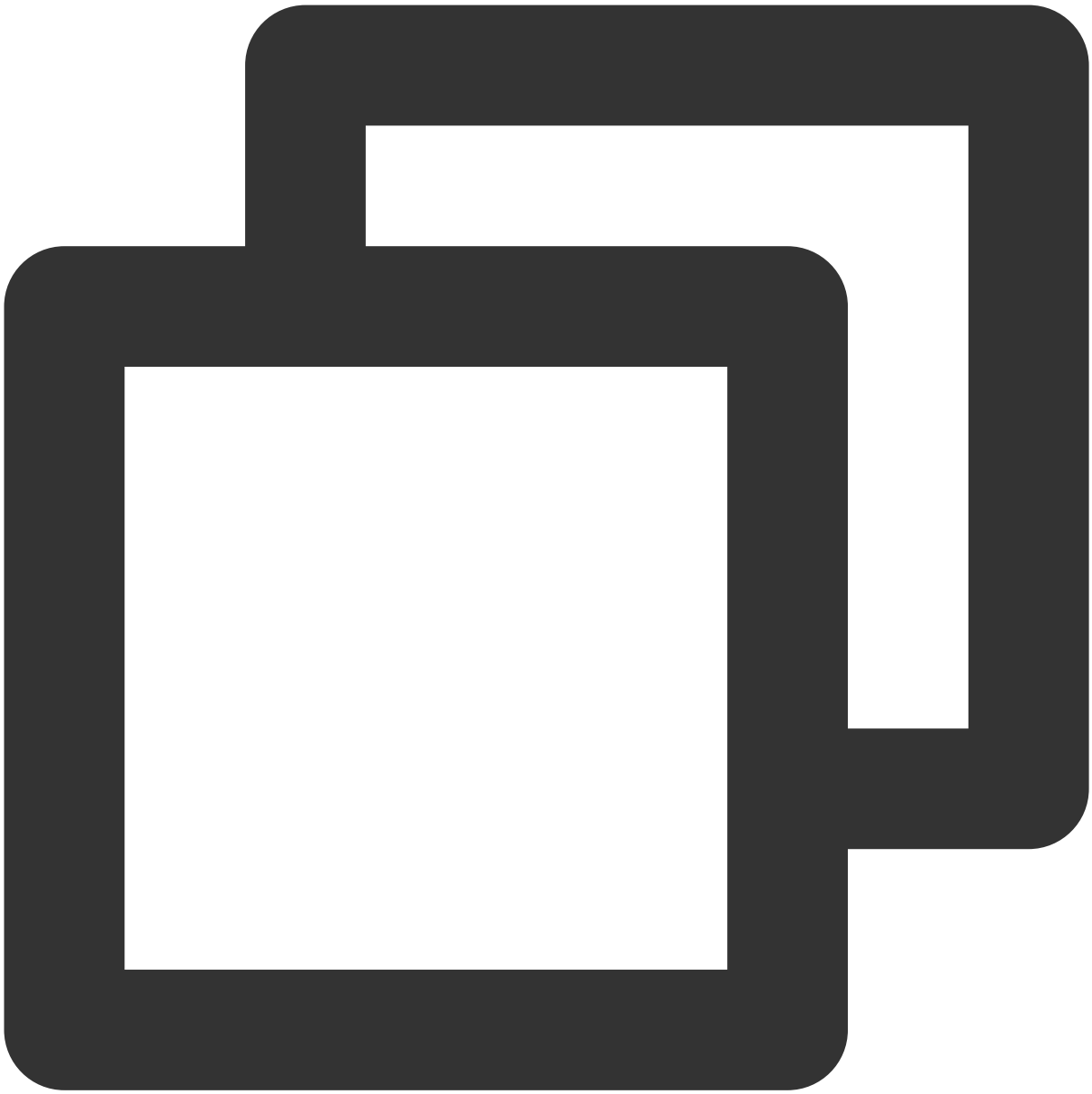
方法1：パイプラインオペレーターを使用してURLを実装すると同時に2種類のウォーターマークを付与する

Cloud Infiniteの基本画像処理機能では、パイプラインオペレーター "|"を使用して、1回のリクエストで複数回の処理を行うことができ、なおかつ処理料金とトラフィック料金は1回分として計算されます。この方式ではリクエストの繰り返しによる遅延と追加料金を大幅に減らすことができます。

操作手順

1. 画像ウォーターマークのAPIドキュメントを参照し、画像ウォーターマークのパラメータを定義します。
APIパラメータに不慣れな場合は、スタイル管理のドキュメントを参照し、コンソールでスタイルを追加してパラ

メータを生成することができます。

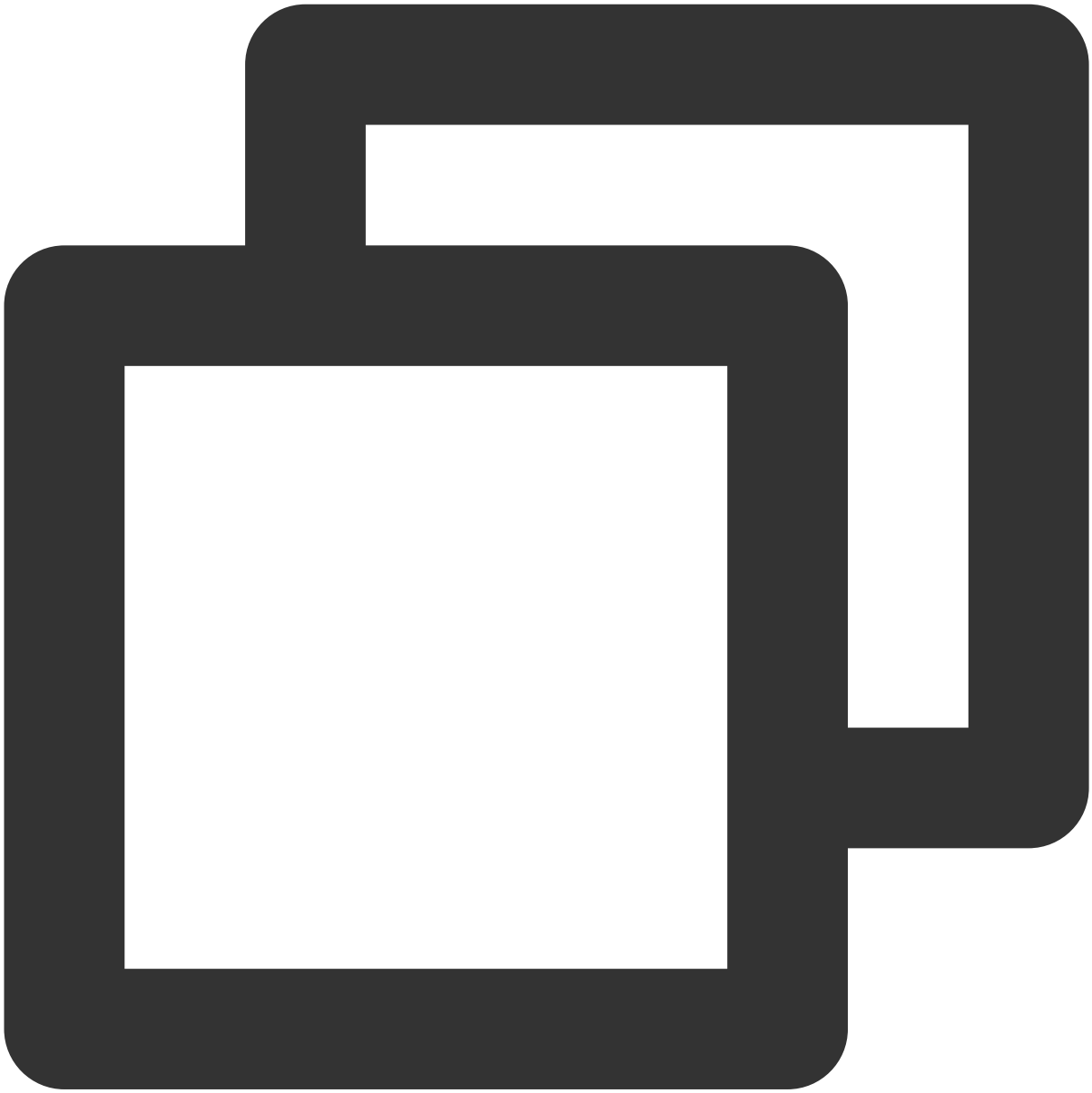


```
watermark/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTgxMjU2MzguY29zLmFwLWd1YW5nemhvdS5teXFjb
```

説明：

ここのコード `aHR0cDovL2V4YW1wbGVzLTEyNTgxMjU2MzguY29zLmFwLWd1YW5nemhvdS5teXFjbG91ZC5jb20vbG9nby5wbmc` は、ウォーターマーク画像リンク（COS bucket上に保存されている画像のリンク）にURLセーフなBase64エンコードを経て生成されたものです。

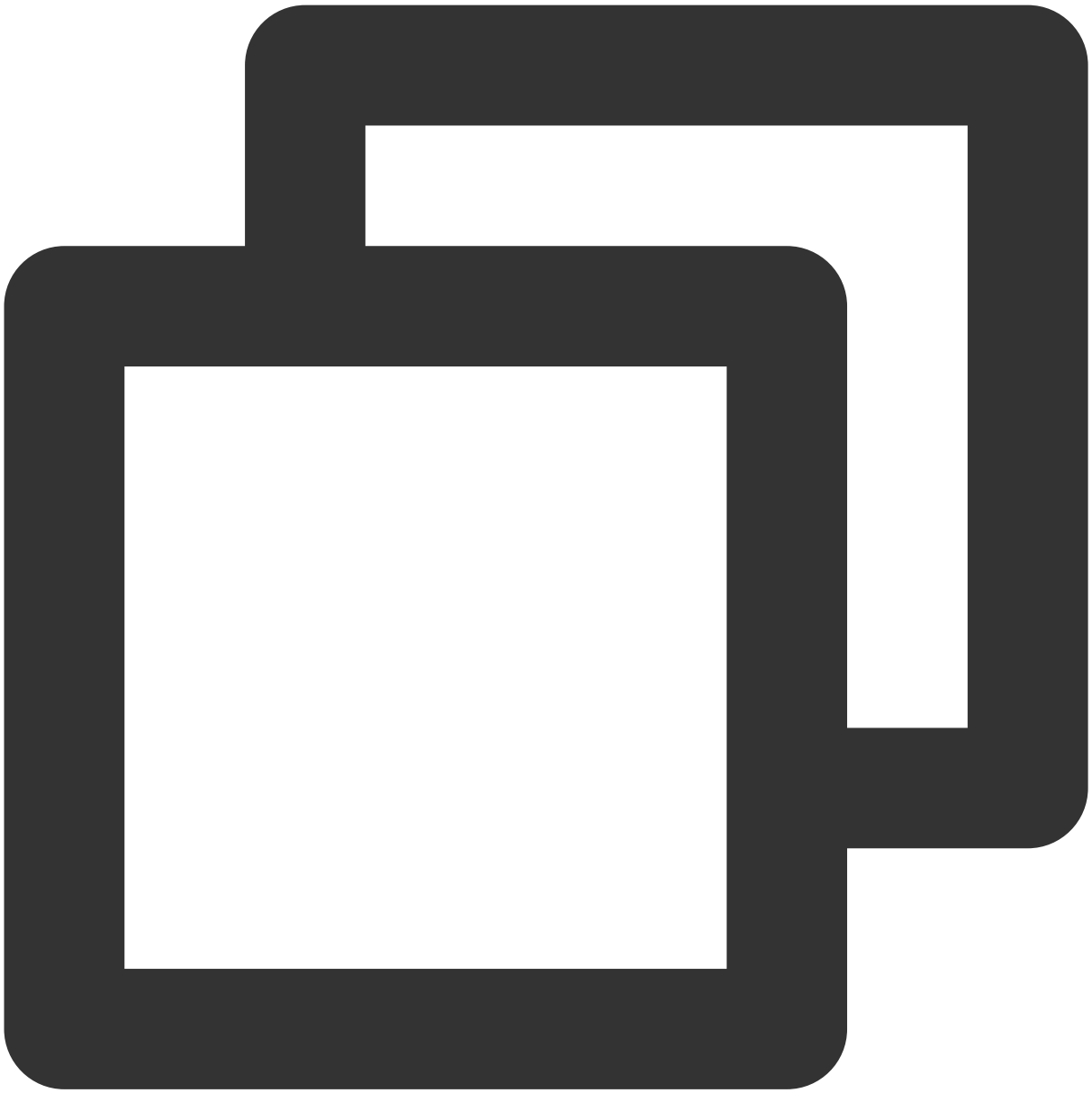
2. テキストウォーターマークのAPIドキュメントを参照し、テキストウォーターマークのパラメータを定義します。
APIパラメータに不慣れな場合は、スタイル管理を参照し、コンソールでスタイルを追加してパラメータを生成することができます。



```
watermark/2/text/VUlOOiAxMjM0NTY3OA/font/SGVsdmV0aWNhLmRmb250/fontsize/36/fill/IzAw
```

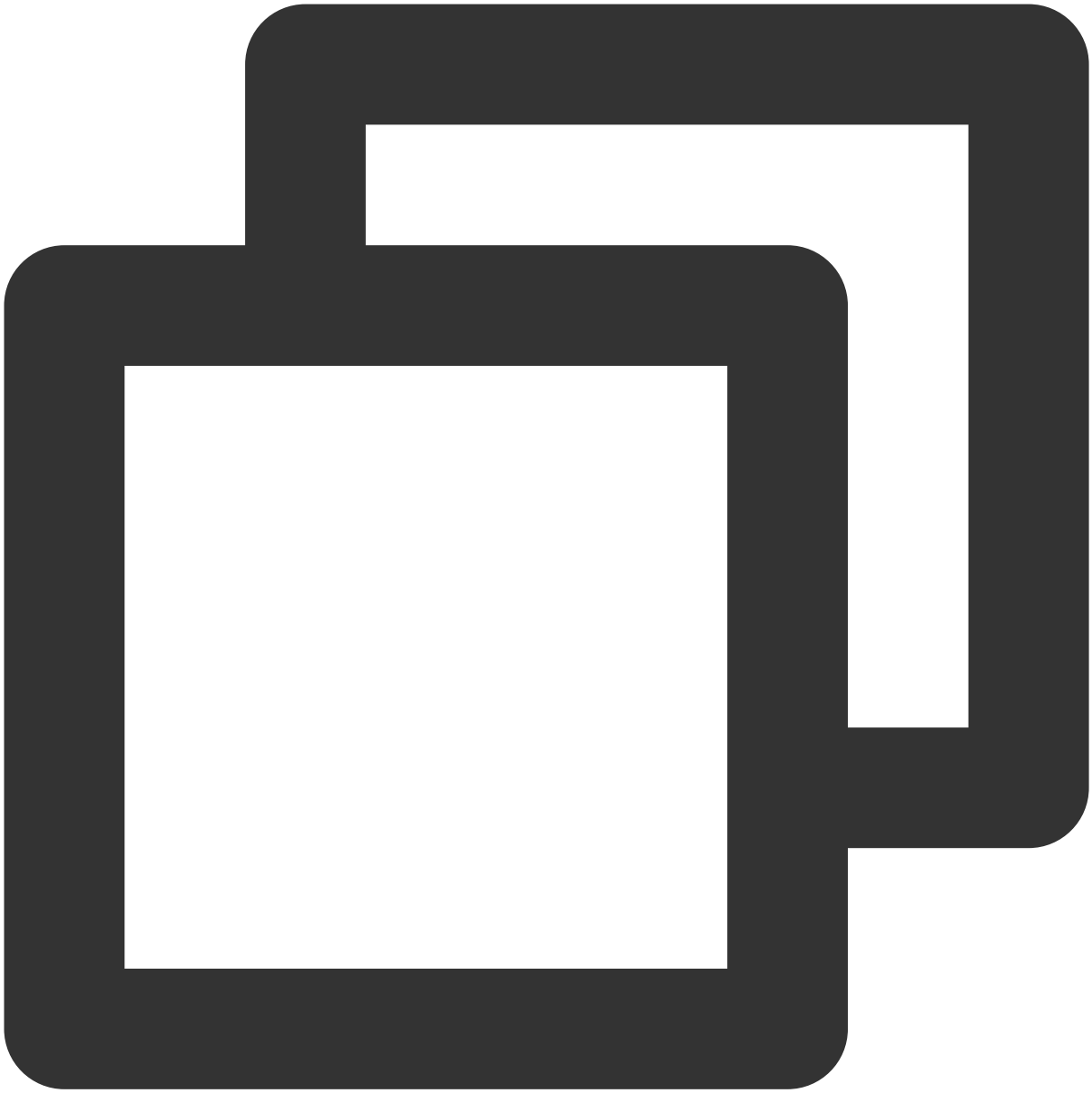
説明：
ここのコード `VUlOOiAxMjM0NTY3OA` はテキスト情報 `UIN: 12345678` にURLセーフなBase64エンコードを行って生成されたものです。

3. パイプラインオペレーターを使用して画像ウォーターマークとテキストウォーターマークの2つの部分のパラメータをスプライシングします。



```
watermark/2/text/VUlOOiAxMjM0NTY3OA/font/SGVsdmV0aWNhLmRmb250/fontsize/36/fill/IzAw
```

4. スプライシングしたパラメータを画像のダウンロードリンクの後ろにつなげます。

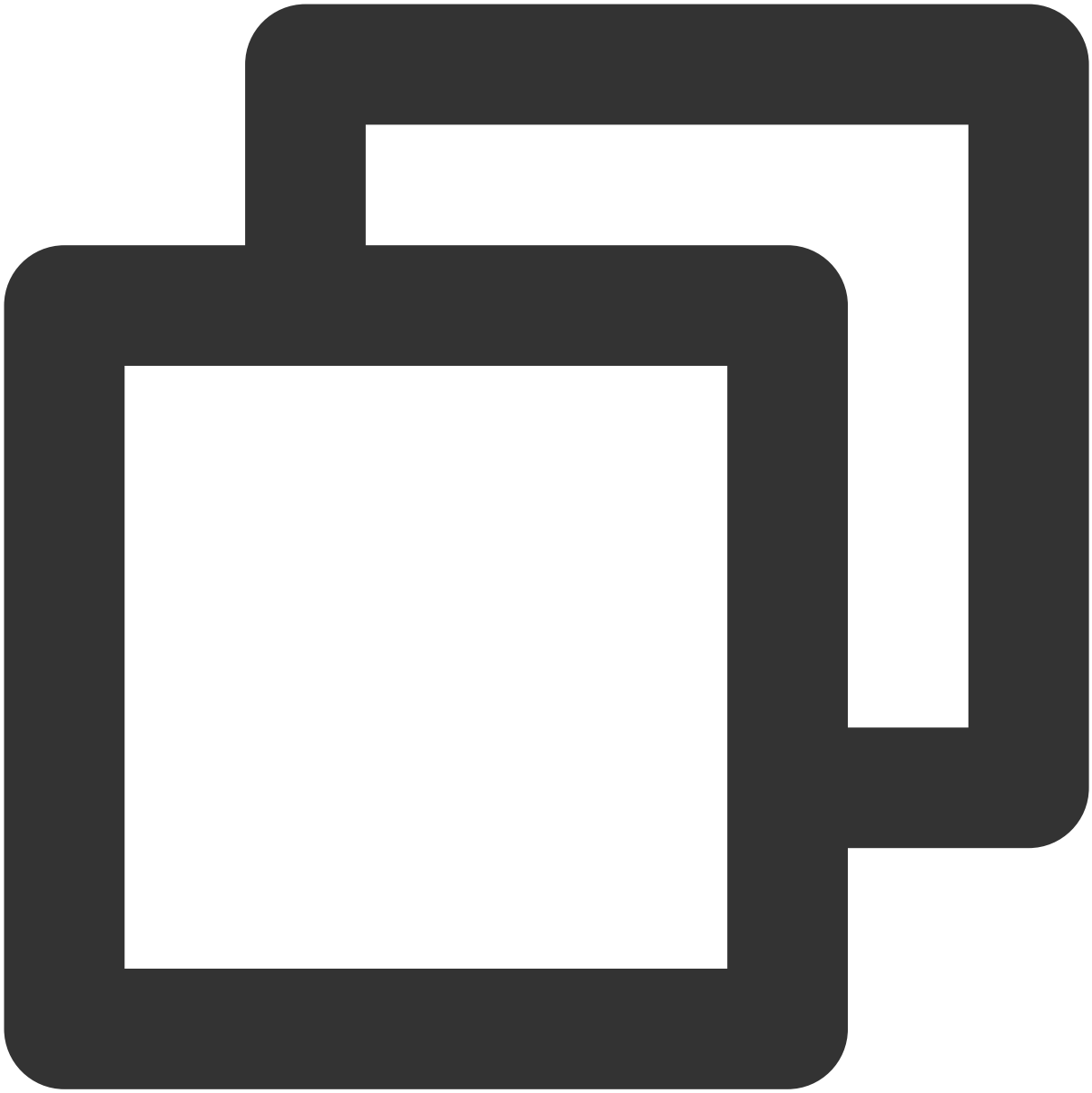


```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png?watermark/2/t
```

これで混合ウォーターマークの画像が得られます。

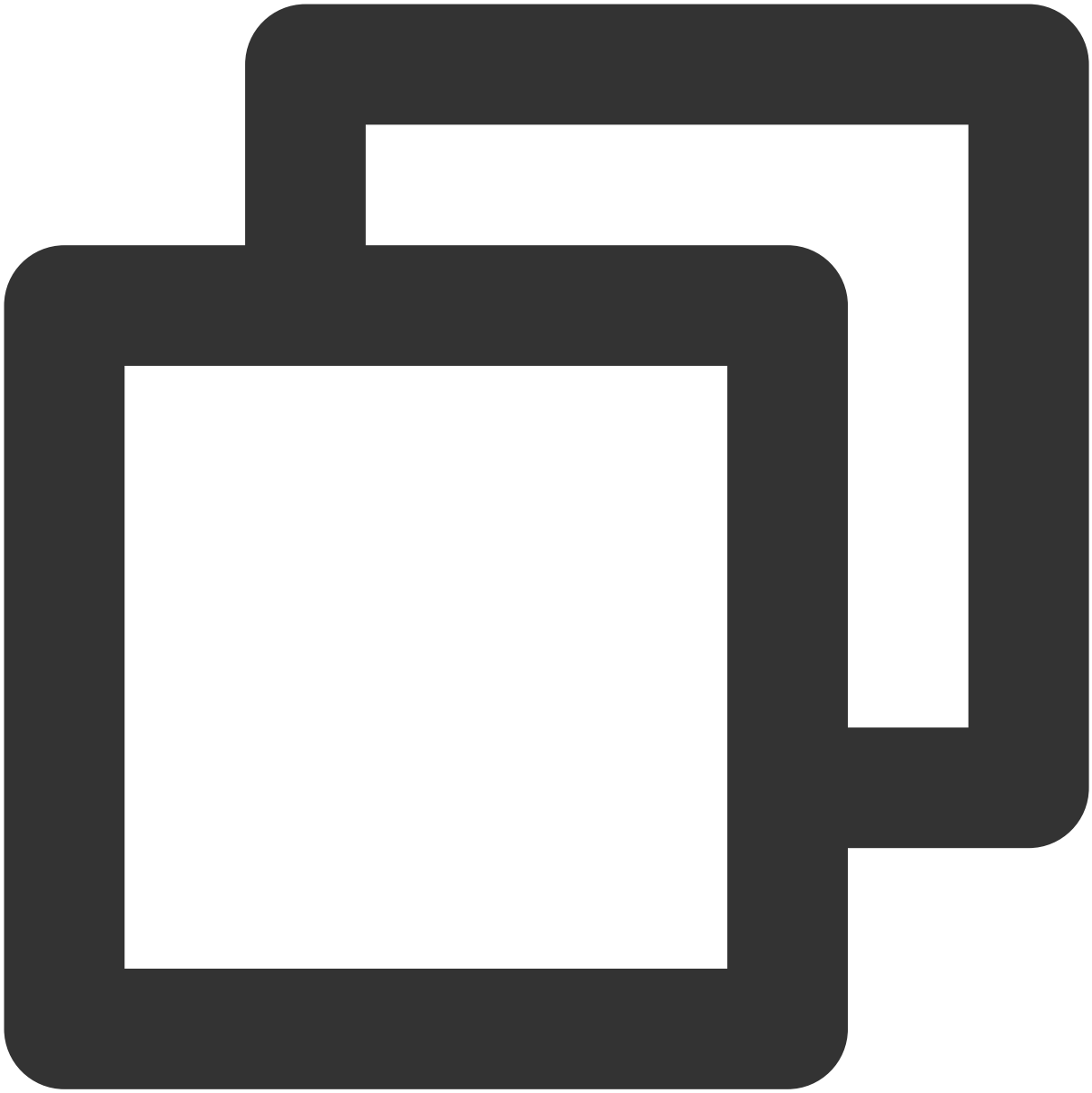
URLの長さを短縮するには、スタイル管理のドキュメントを参照し、コンソールで画像ウォーターマーク（変更なし）の部分をスタイル `watermark1` として追加します。

こうすることで、リンクは次のように短縮されます。



```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png/watermark1?wa
```

その後テキストの内容を変更したい場合は、リンク内の `VUlOOiAxMjM0NTY3OA` の部分を更新後のBase64エンコードに変更するだけで実現できます。例えば `UIN: 88888888` のコードは `VUlOOiA4ODg4ODg4OA` であり、この場合リンクを次の内容に変更するだけで、テキスト内容の入れ替えが実現できます。



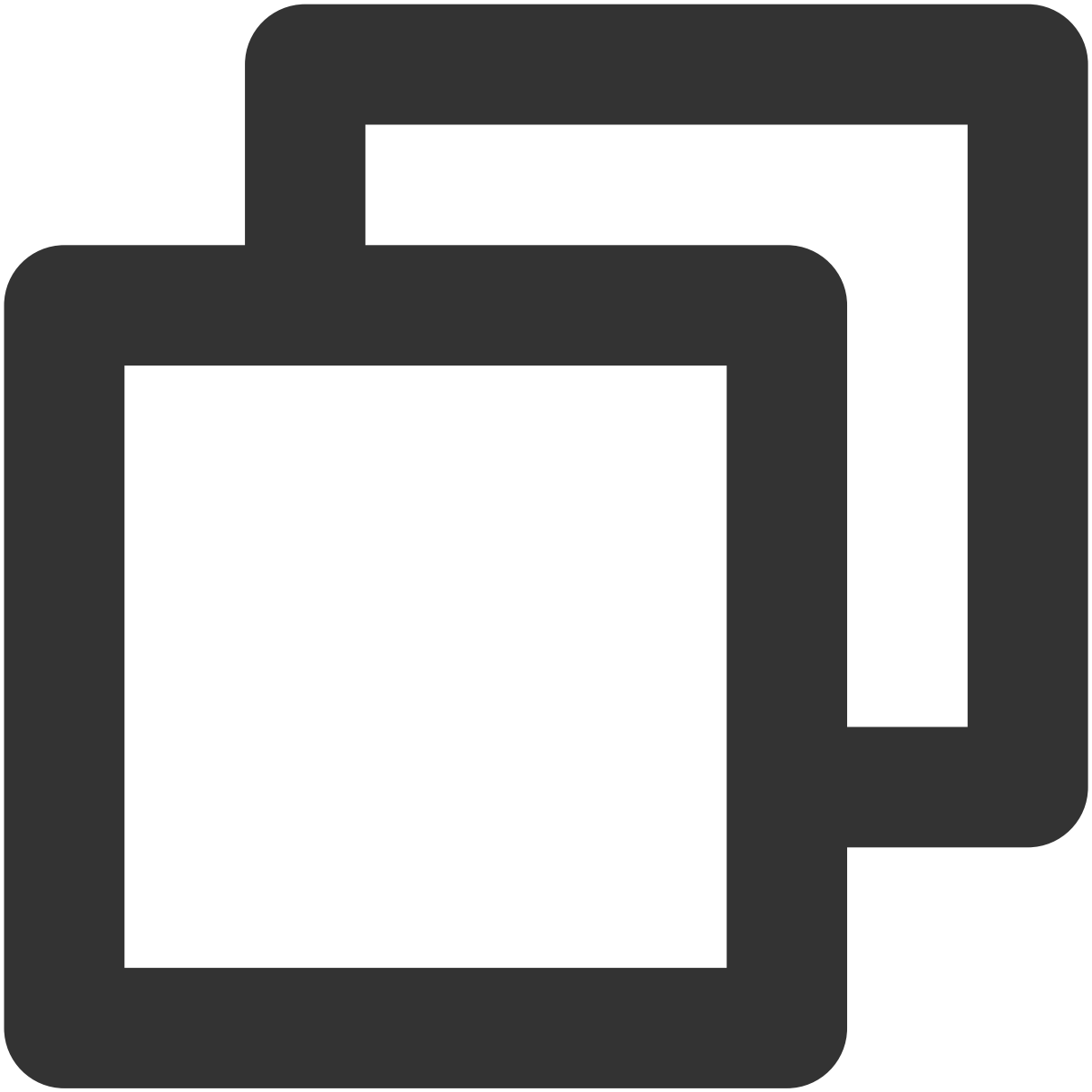
```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png/watermark1?wa
```

方法2：テキストおよび画像ウォーターマークを透明な画像上に印刷し、それを画像ウォーターマークにする

1. 400px * 400pxサイズの透明PNG画像1枚を用意し、 バケットにアップロードします。詳細についてはファイルのアップロードのドキュメントをご参照ください。

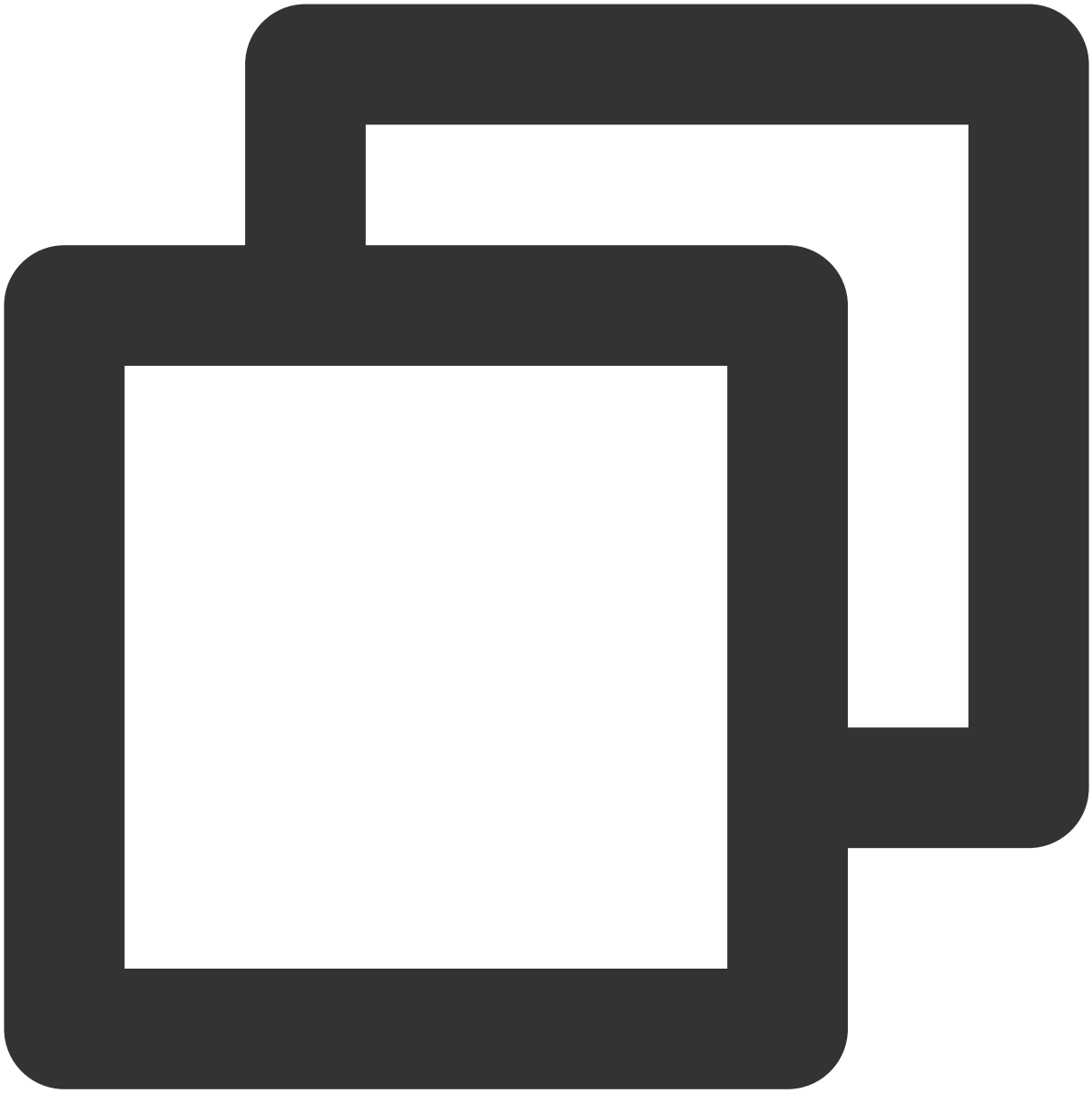
例： `https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/transparent.png`

2. **方法1のステップ1 - ステップ3**を参照し、画像ウォーターマークとテキストウォーターマークの2つの部分のパラメータを生成し、スプライシングします。
3. スプライシングしたパラメータを透明PNG画像のダウンロードリンクの後ろにつなげます。



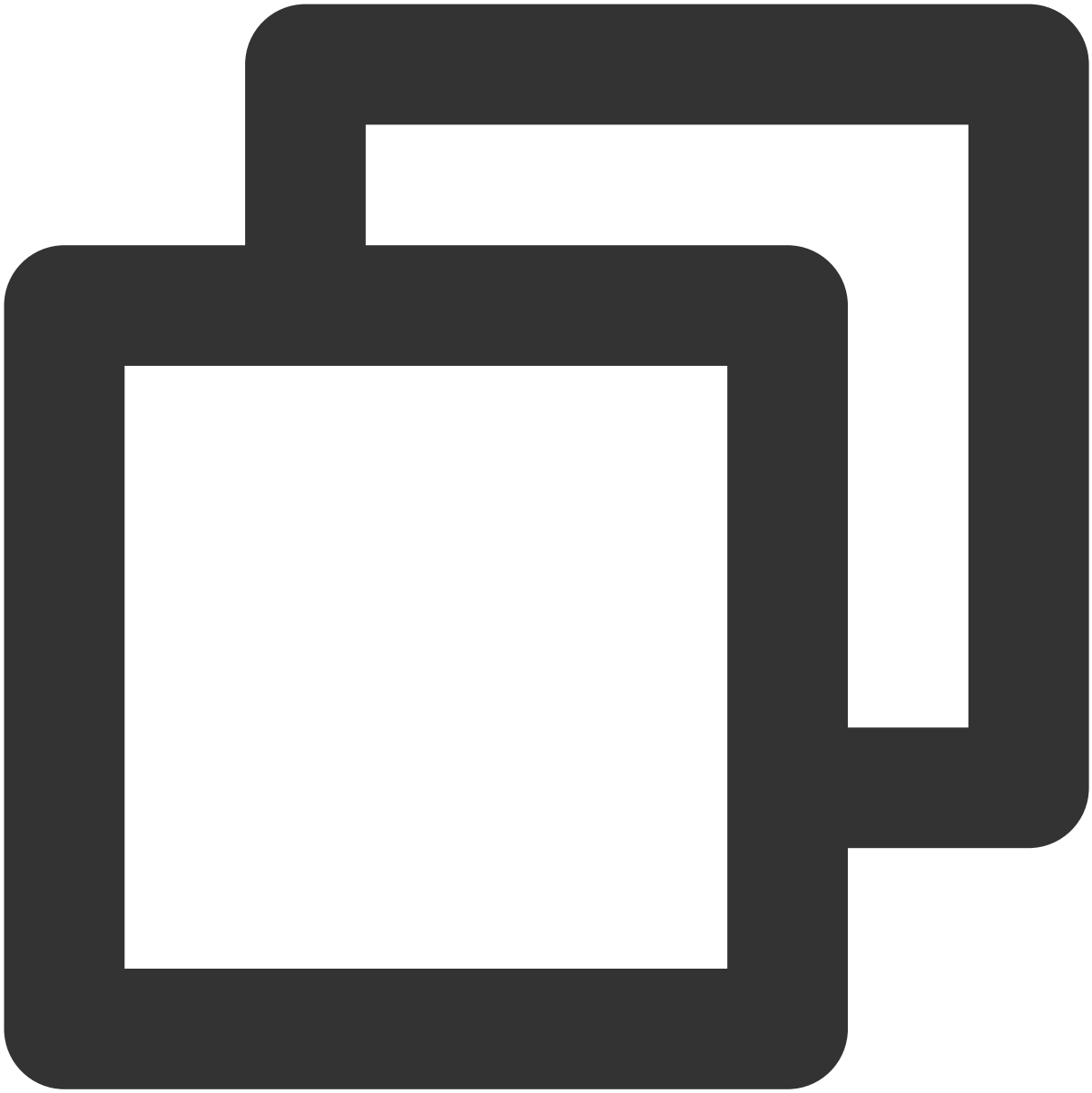
```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/transparent.png?watermark
```

4. この透明画像をウォーターマーク画像とし、オリジナル画像に対してウォーターマーク付与操作を行えば完了です。



```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png?watermark/1/i
```

また、 `scatype` パラメータによって、ウォーターマーク画像のサイズを画像の大きさに基づいて同じ比率でズームし、 `batch` パラメータによってタイリングを設定することもできます。



```
https://examples-1258125638.cos.ap-guangzhou.myqcloud.com/preview.png?watermark/1/i
```

COSオーディオビデオプレーヤーの実践
COSオーディオビデオプレーヤーの概要

最終更新日：：2024-06-26 10:42:27

ここでは主にCOSオーディオビデオクラウド処理と端末側の再生を実際に適用する方法についてご説明します。文中での実践事例には、オーディオビデオ処理がサポートするプロトコル、機能およびCOSオーディオビデオファイル再生方法の操作ガイドを含めています。またTencent Cloud Infinite（CI）の豊富なオーディオビデオ処理機能と組み合わせることで、製品機能使用のアイデアが広がり、より良い再生パフォーマンスを体験していただけるようになっています。

プロトコルのサポート

オーディオビデオプロトコル	URLアドレス形式	PCブラウザ	モバイル端末ブラウザ
MP3	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp3	サポートあり	サポートあり
MP4	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4	サポートあり	サポートあり
HLS (M3U8)	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8	サポートあり	サポートあり
FLV	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.flv	サポートあり	サポートあり
DASH	https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mpd	サポートあり	サポートあり

注意：
HLS、FLV、DASHビデオは一部のブラウザ環境で再生する場合、Media Source Extensions に依存する必要があります。

機能サポート

機能	TCPlayerプレーヤー	DPlayerプレーヤー	Videojsプレーヤー

MP4形式のビデオの再生	詳細を見る	詳細を見る	詳細を見る
HLS形式のビデオの再生	詳細を見る	詳細を見る	詳細を見る
FLV形式のビデオの再生	詳細を見る	詳細を見る	詳細を見る
DASH形式のビデオの再生	詳細を見る	詳細を見る	詳細を見る
PM3U8 (プライベートM3U8)ビデオの再生	詳細を見る	詳細を見る	詳細を見る
カバー画像の設定	詳細を見る	詳細を見る	詳細を見る
HLS標準暗号化の設定	詳細を見る	詳細を見る	詳細を見る
解像度の切り替え	詳細を見る	詳細を見る	-
動的ウォーターマークの設定	詳細を見る	-	-
左上隅のロゴ設定	-	詳細を見る	-
進捗プレビュー画像の設定	詳細を見る	-	-
字幕の設定	詳細を見る	-	-
多言語の設定	詳細を見る	-	-
ロール画像広告の設定	詳細を見る	-	-

説明：
プレーヤーは一般的なブラウザと互換性があり、プレーヤーがプラットフォームを自動的に識別し、最適な再生方式を使用します。例としては、Chromeなどの最新ブラウザではHTML5技術を優先的に使用してビデオ再生を実現します。またスマホブラウザでは、HTML5技術やブラウザカーネル機能を使用してビデオ再生を実現します。

使用ガイド

TCPlayerを使用したCOSビデオファイルの再生
DPlayerを使用したCOSビデオファイルの再生
VideojsPlayerを使用したCOSビデオファイルの再生

TcPlayerを使用したCOSビデオファイルの再生

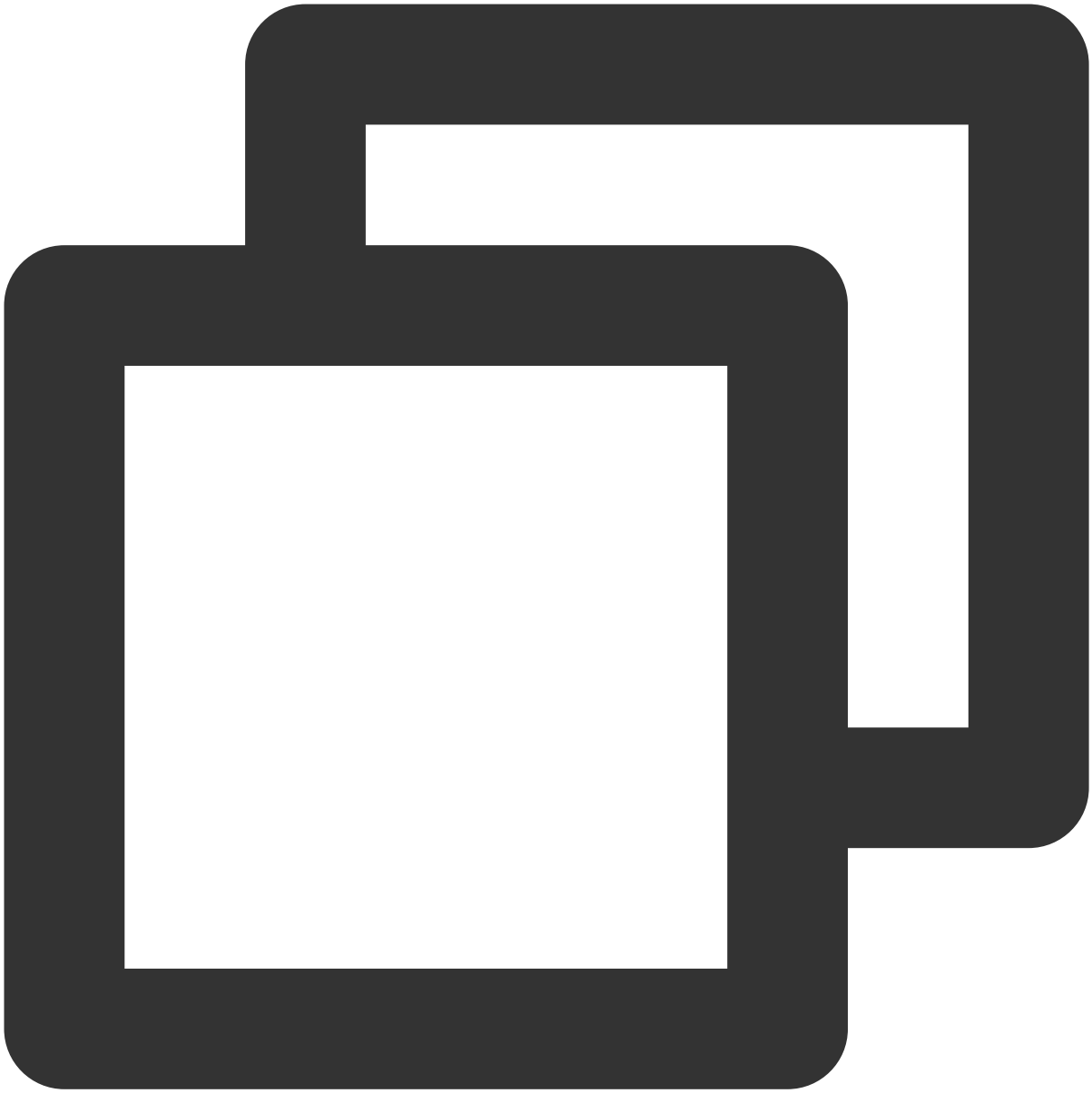
最終更新日：：2024-06-26 10:42:27

概要

ここでは、オーディオビデオ端末SDK（Tencent Cloud View Cube）を統合したTCPlayerを使用し、Tencent Cloud Infinite(CI)の提供する豊富なオーディオビデオ機能と組み合わせて、WebブラウザでCOSビデオファイルを再生する方法についてご説明します。

統合ガイド

ステップ1：ページにプレーヤースタイルファイルとスクリプトファイルを導入する



```
<!--プレーヤースタイルファイル-->
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/tcplayer.min.
<!--プレーヤースクリプトファイル-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.0/tcplayer.v4.
```

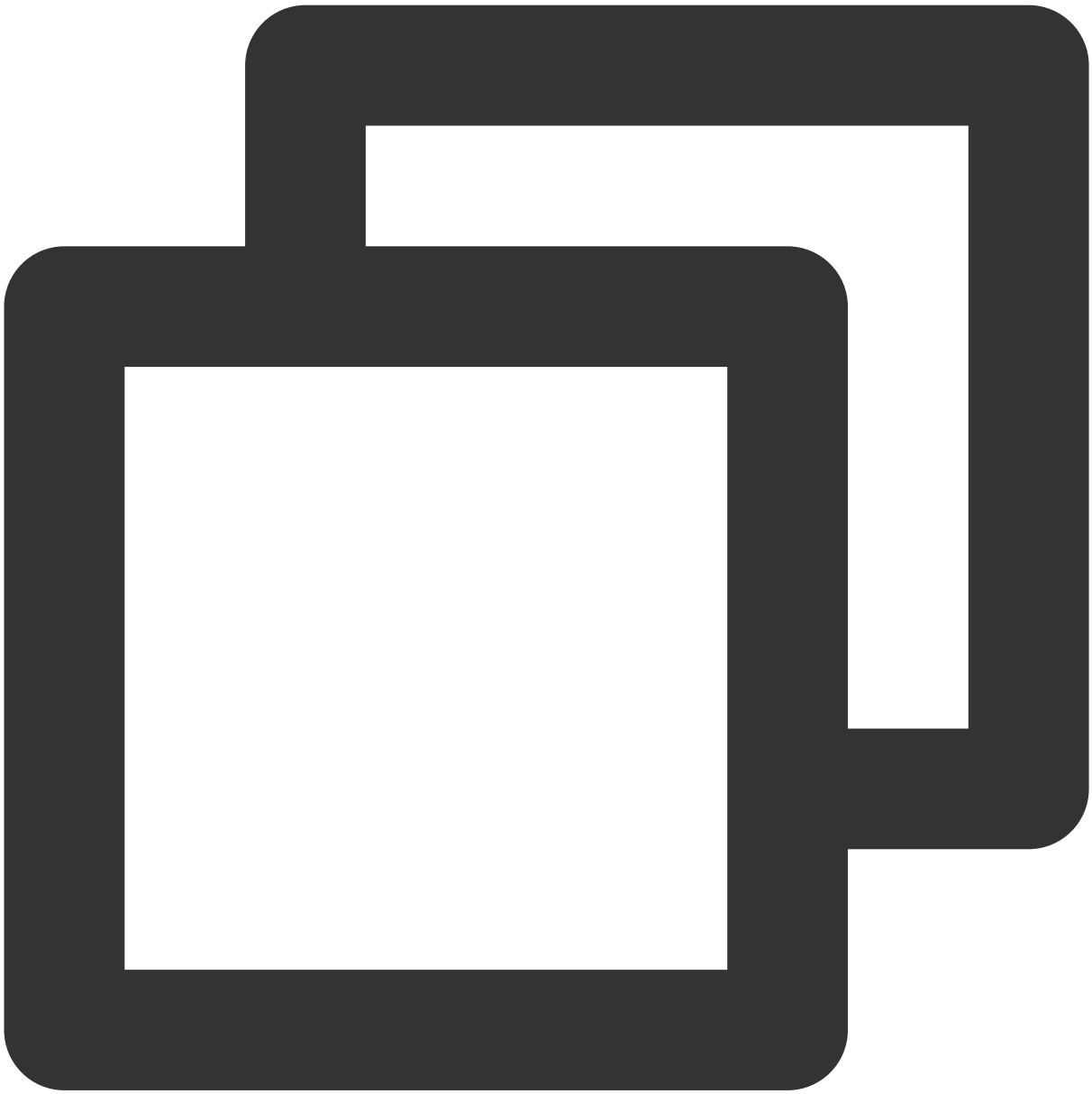
説明：

Player SDKを正式に使用する際は、上記の関連静的リソースをご自身でデプロイすることをお勧めします。クリックしてプレーヤーリソースをダウンロードします。

デプロイし解凍した後のフォルダについては、リソースの相互引用の異常を回避するため、フォルダ内のディレクトリを調整することができません。

ステップ2：プレーヤーコンテナノードを設定する

プレーヤーを表示したいページ位置にプレーヤーコンテナを追加します。例えば、index.htmlに次のコードを追加します（コンテナIDおよび幅と高さはいずれもカスタマイズできます）。



```
<video id="player-container-id" width="414" height="270" preload="auto" playsinline
</video>
```

説明：

プレーヤーコンテナは `<video>` タグである必要があります。

例中の `player-container-id` はプレーヤーコンテナのIDであり、自身で設定することができます。

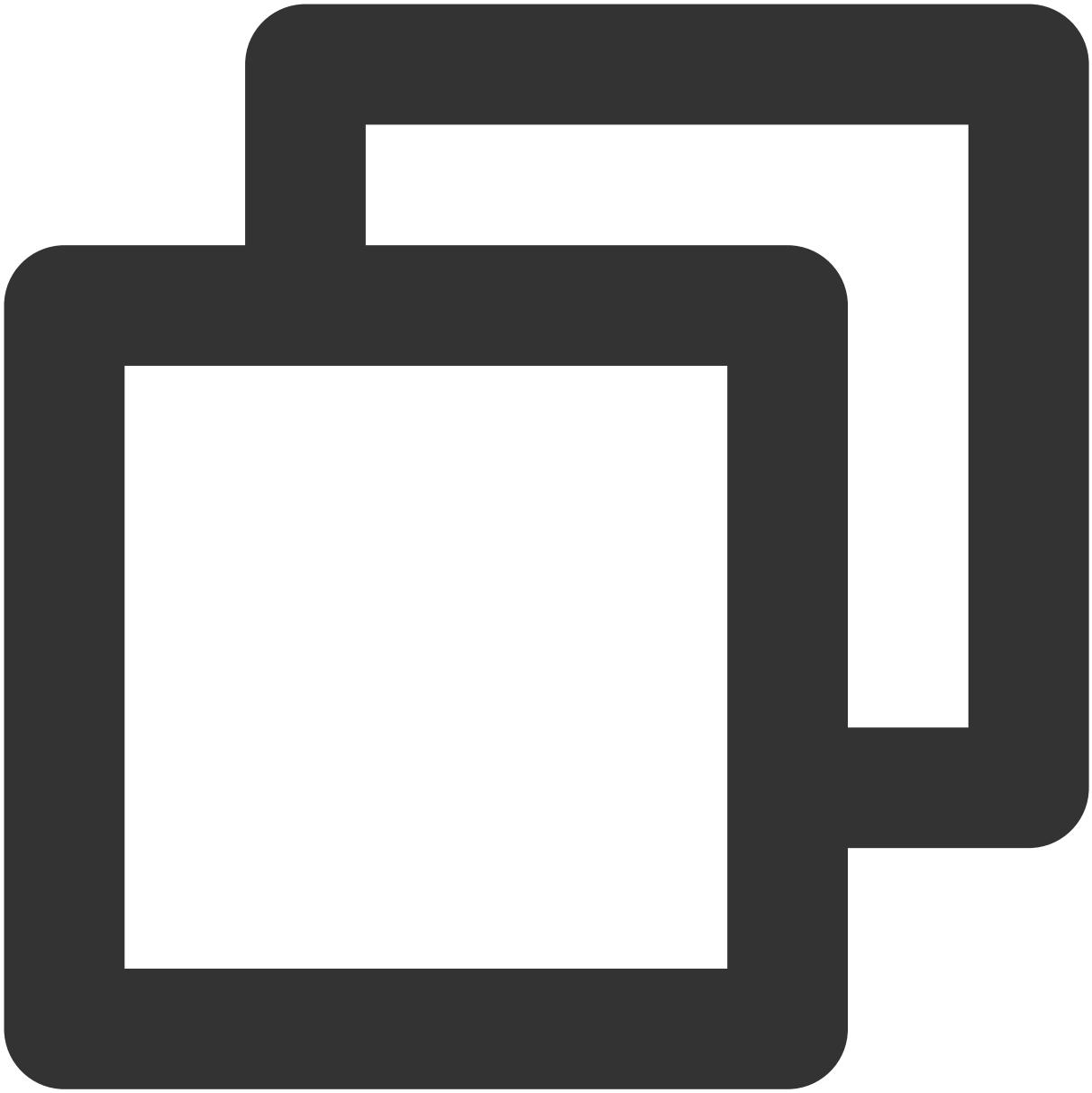
プレーヤーコンテナ領域のサイズについては、CSSを介して設定することをお勧めします。CSS設定はプロパティ設定よりもフレキシブルで、フルスクリーンやコンテナ適応などの効果を実現することができます。
例中の `preload` 属性はページのロード後にビデオをロードするかどうかを指定します。通常、ビデオ再生を高速化するため、 `auto` に設定しますが、その他のオプション値には、 `meta` （ページロード後にメタデータのみをロード）, `none` （ページロード後にビデオをロードしない）があります。モバイル端末ではシステムの制限のために自動的にビデオをロードすることができません。
`playsinline` や `webkit-playsinline` といったいくつかの属性は標準的なモバイル端末ブラウザがビデオ再生をハイジャックせずにインライン再生を実現するためのものです。ここでは例示するにとどめますが、必要に応じてご使用ください。
`x5-playsinline` 属性をTBSカーネルに設定すると、X5UIのプレーヤーを使用できます。

ステップ3：ビデオファイルオブジェクトアドレスを取得する

1. バケットの作成を行います。
2. ビデオファイルのアップロードを行います。
3. ビデオファイルのオブジェクトアドレスを取得します。形式は `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<ビデオ形式>` です。

説明：
クロスドメインの問題がある場合は、バケットのクロスドメインアクセスCORS設定を行う必要があります。詳細については、クロスドメインアクセスの設定をご参照ください。
バケットがプライベート読み取り/書き込みの場合、オブジェクトアドレスには署名が必要です。詳細については、リクエスト署名をご参照ください。

ステップ4：プレーヤーを初期化し、COSビデオファイルのオブジェクトアドレスURLを渡す



```
var player = TCPlayer("player-container-id", {}); // player-container-idはプレーヤー
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"); // COSと
```

機能ガイド

さまざまな形式のビデオファイルを再生する

1. COSバケット上のビデオファイルのオブジェクトアドレスを取得します。

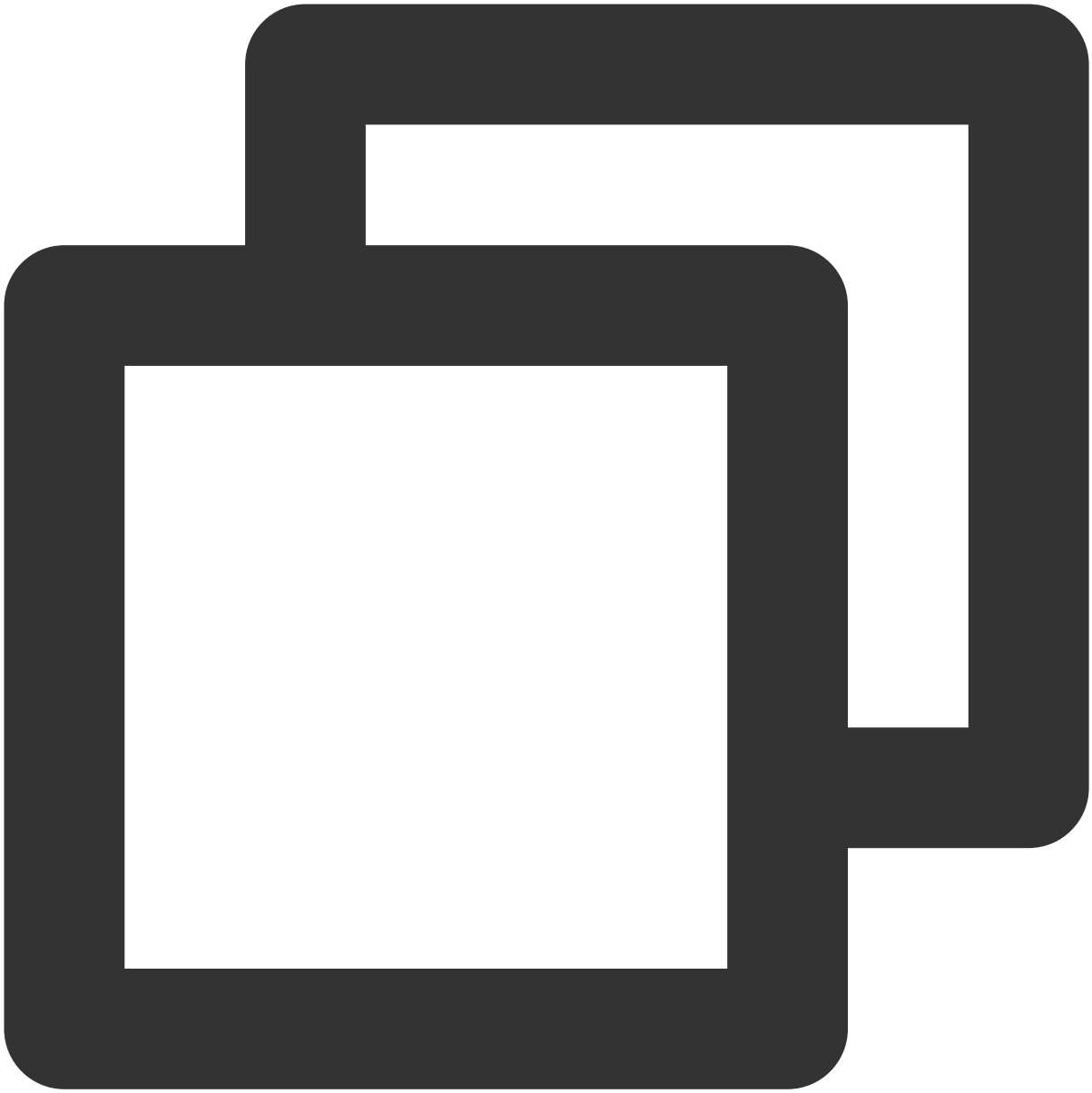
説明：

トランスコードされていないソースビデオは再生の際に互換性の問題が生じる可能性があるため、トランスコード後のビデオで再生を行うことをお勧めします。Cloud Infiniteのオーディオビデオトランスコーディング処理により、さまざまな形式のビデオファイルを取得することができます。

2. さまざまなビデオ形式に対し、マルチブラウザでの互換性を保証するためには、対応する依存関係の導入が必要です。

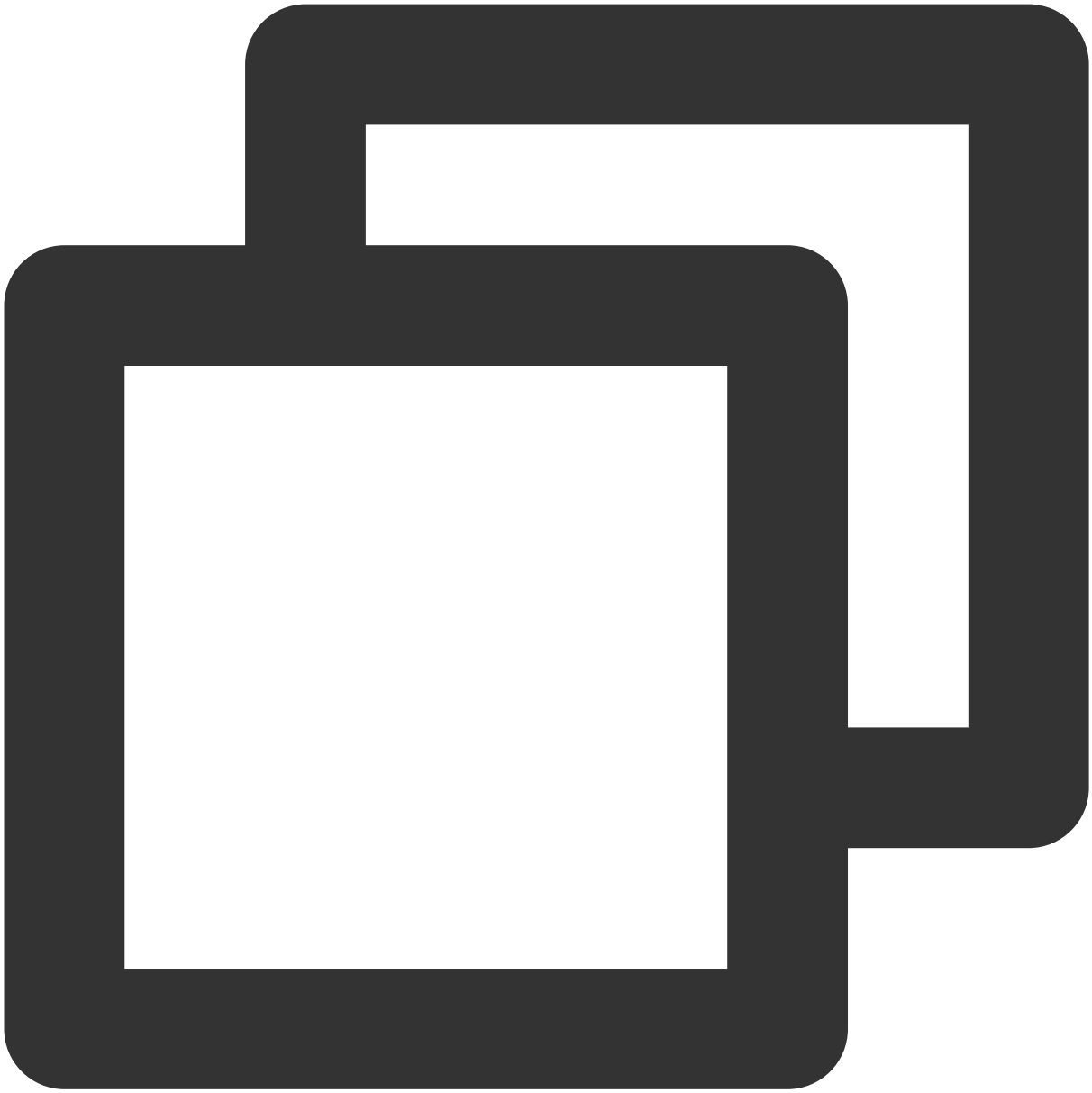
MP4：他の依存関係を導入する必要はありません。

HLS：ChromeやFirefoxなどの最新ブラウザでH5を介してHLS形式のビデオを再生したい場合は、 tcplayer.min.jsの前にhls.min.jsを導入する必要があります。



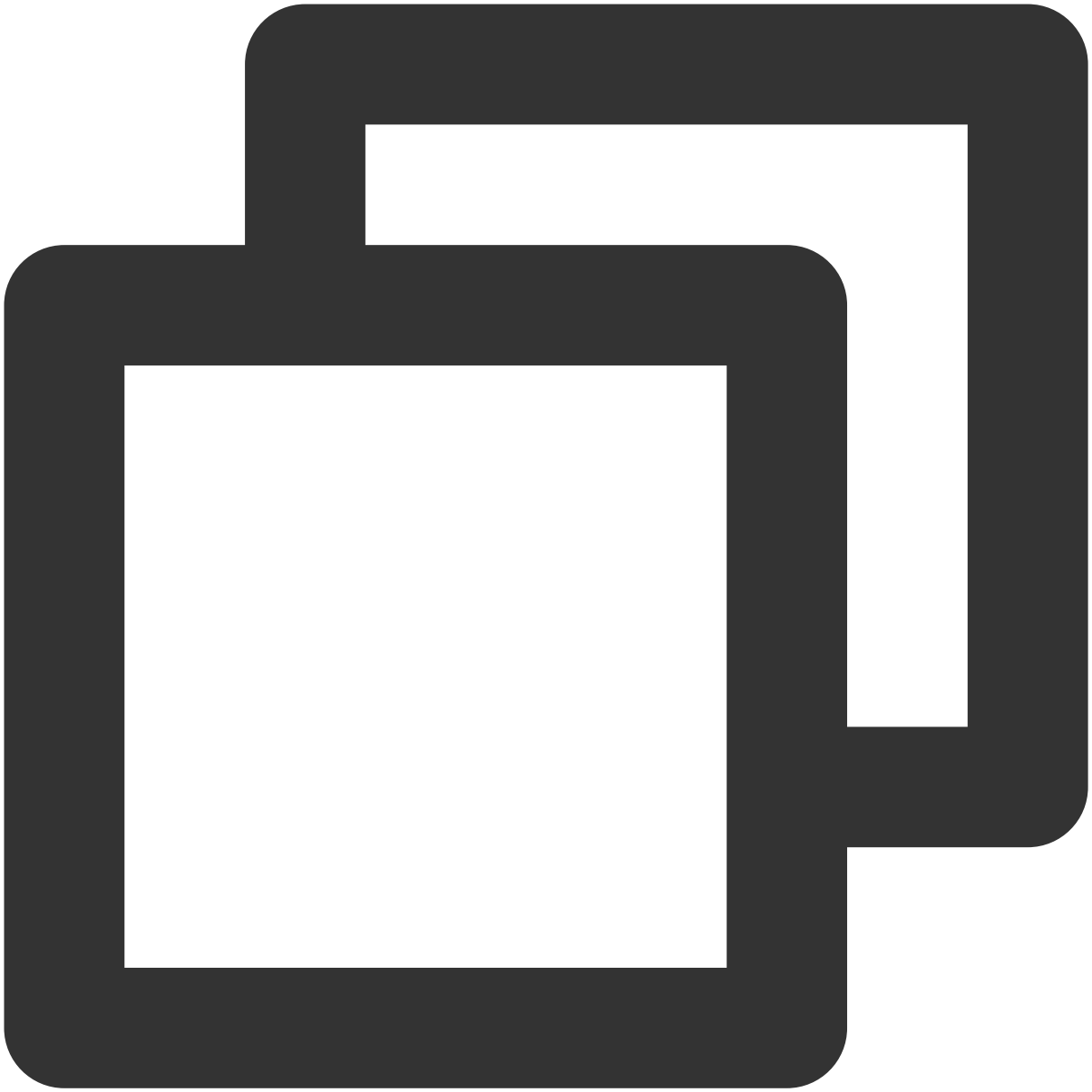
```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/libs/hls.m
```

FLV：ChromeやFirefoxなどの最新ブラウザでH5を介してFLV形式のビデオを再生したい場合は、 tcplayer.min.js の前にflv.min.jsを導入する必要があります。



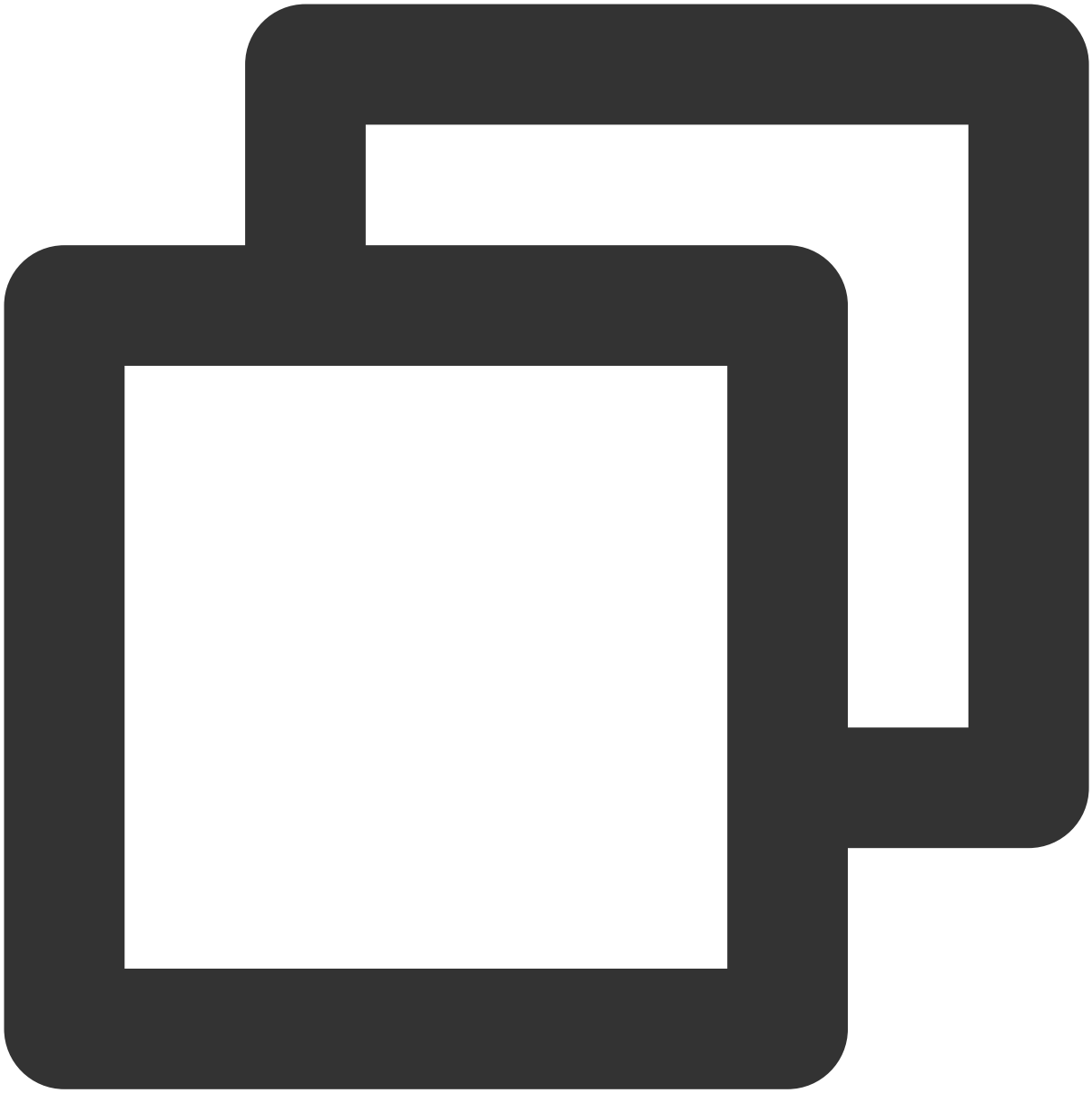
```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.m
```

DASH：DASHビデオにはdash.all.min.jsファイルのロードが必要です。



```
<script src="https://cos-video-1258344699.cos.ap-guangzhou.myqcloud.com/lib/dash.
```

3. プレーヤーを初期化し、オブジェクトアドレスを渡します。



```
var player = TCPlayer("player-container-id", {}); // player-container-idはプレーヤー
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"); // COSと
```

サンプルコードを取得します。

MP4再生サンプルコード

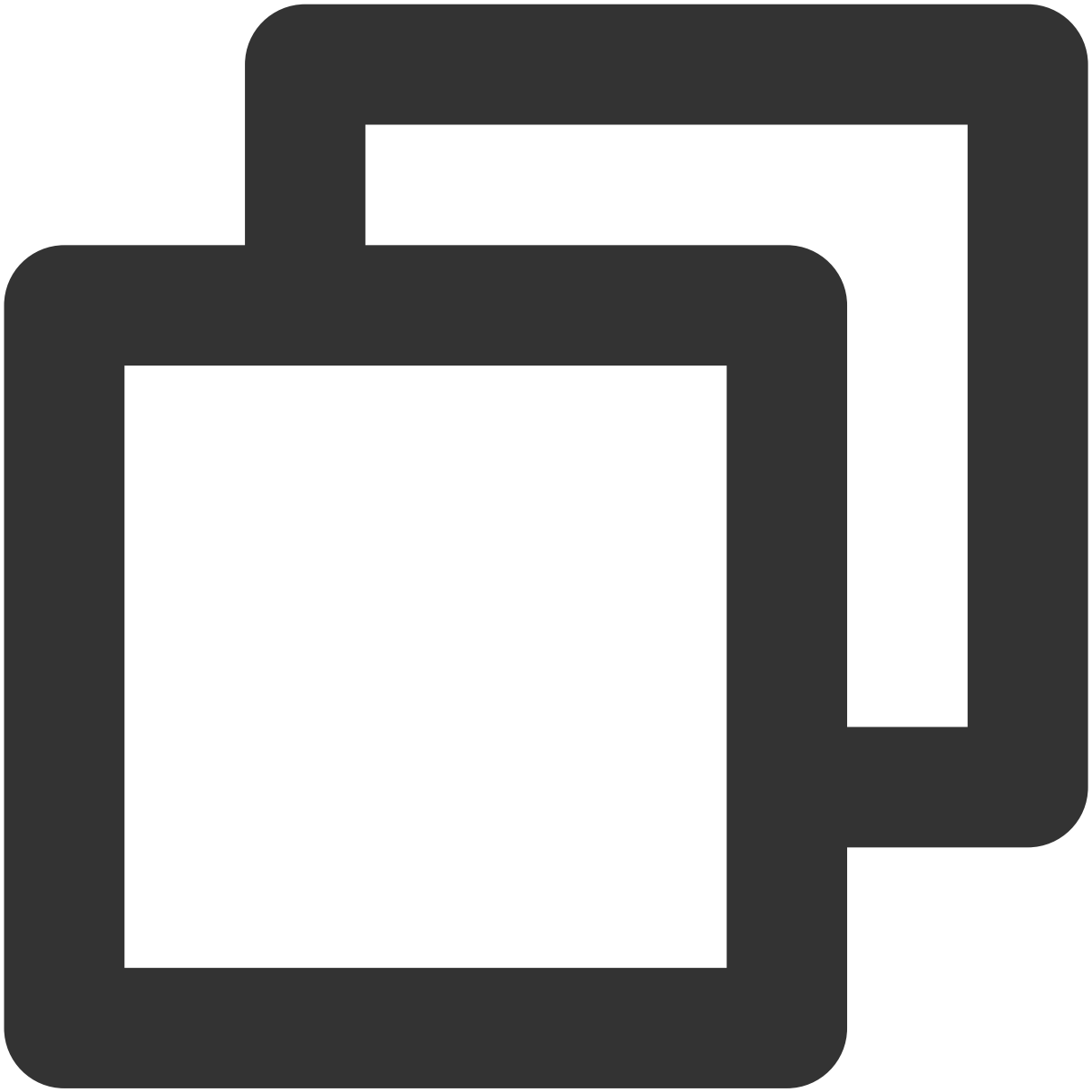
FLV再生サンプルコード

HLS再生サンプルコード

DASH再生サンプルコード

PM3U8ビデオを再生する

PM3U8はプライベートM3U8ビデオファイルのことです。COSはプライベートM3U8 TSリソースの取得に用いるダウンロード権限承認APIを提供しています。プライベートM3U8インターフェースをご参照ください。



```
var player = TCPlayer("player-container-id", {
    poster: "https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8?ci-proce
});
```

サンプルコードを取得します。

PM3U8再生サンプルコード

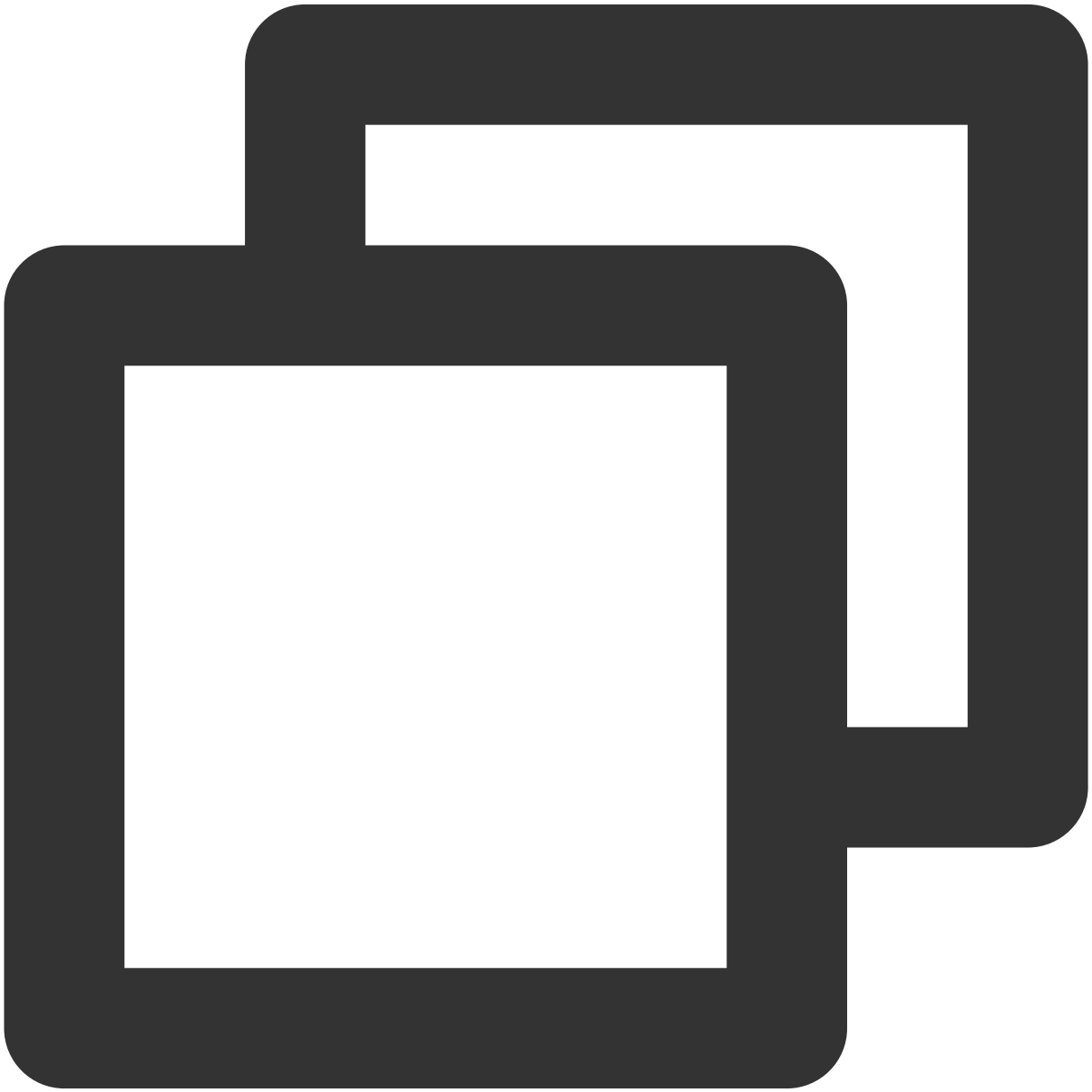
カバー画像を設定する

1. COSバケット上のカバー画像のオブジェクトアドレスを取得します。

注意：

Cloud Infiniteインテリジェントカバー機能により、最適なフレームを抽出してスクリーンキャプチャを生成し、それをカバーにすることでコンテンツの魅力をアップさせます。

2. カバー画像を設定します。



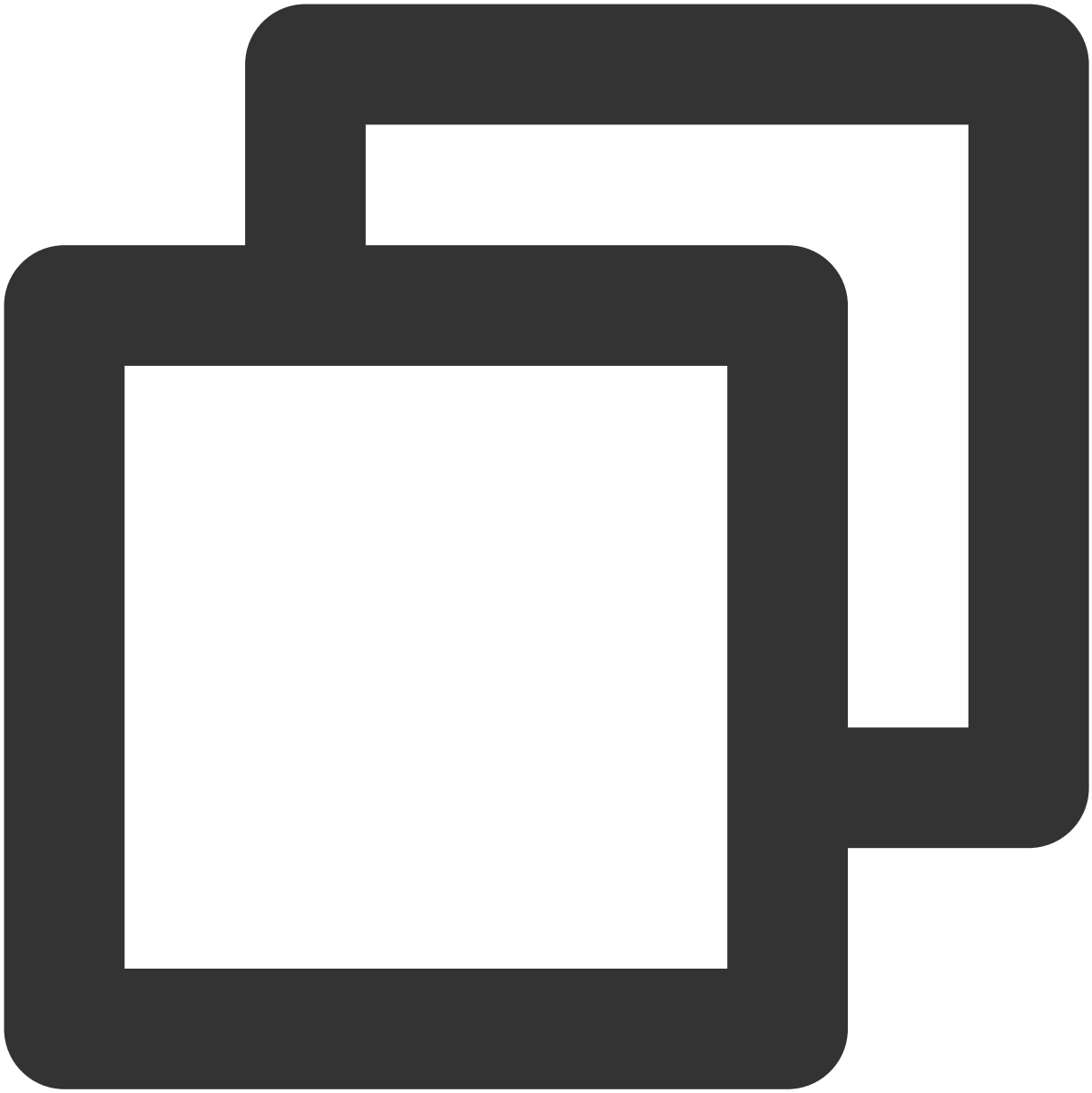
```
var player = TCPlayer("player-container-id", {
```

```
  poster: "https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.png"
 });
```

サンプルコードを取得します。
カバー画像設定サンプルコード

HLS暗号化ビデオを再生する

ビデオコンテンツの安全性を保障し、ビデオの違法なダウンロードと拡散を防止するため、Cloud InfiniteはHLSビデオコンテンツに対する暗号化機能を提供します。この機能はプライベート読み取りファイルよりさらに高いセキュリティレベルを有します。暗号化されたビデオは、アクセス権限のないユーザーの視聴用に配信できなくなります。ビデオがローカルにダウンロードされた場合でも、ビデオ自体が暗号化されているため、悪意ある二次配信が不可能であり、ビデオの著作権が違法に侵害されないよう保障することができます。
操作手順は次のとおりです。
1. HLS暗号化ビデオの再生のフローを参照し、暗号化したビデオを生成します。
2. プレーヤーを初期化し、ビデオオブジェクトアドレスを渡します。



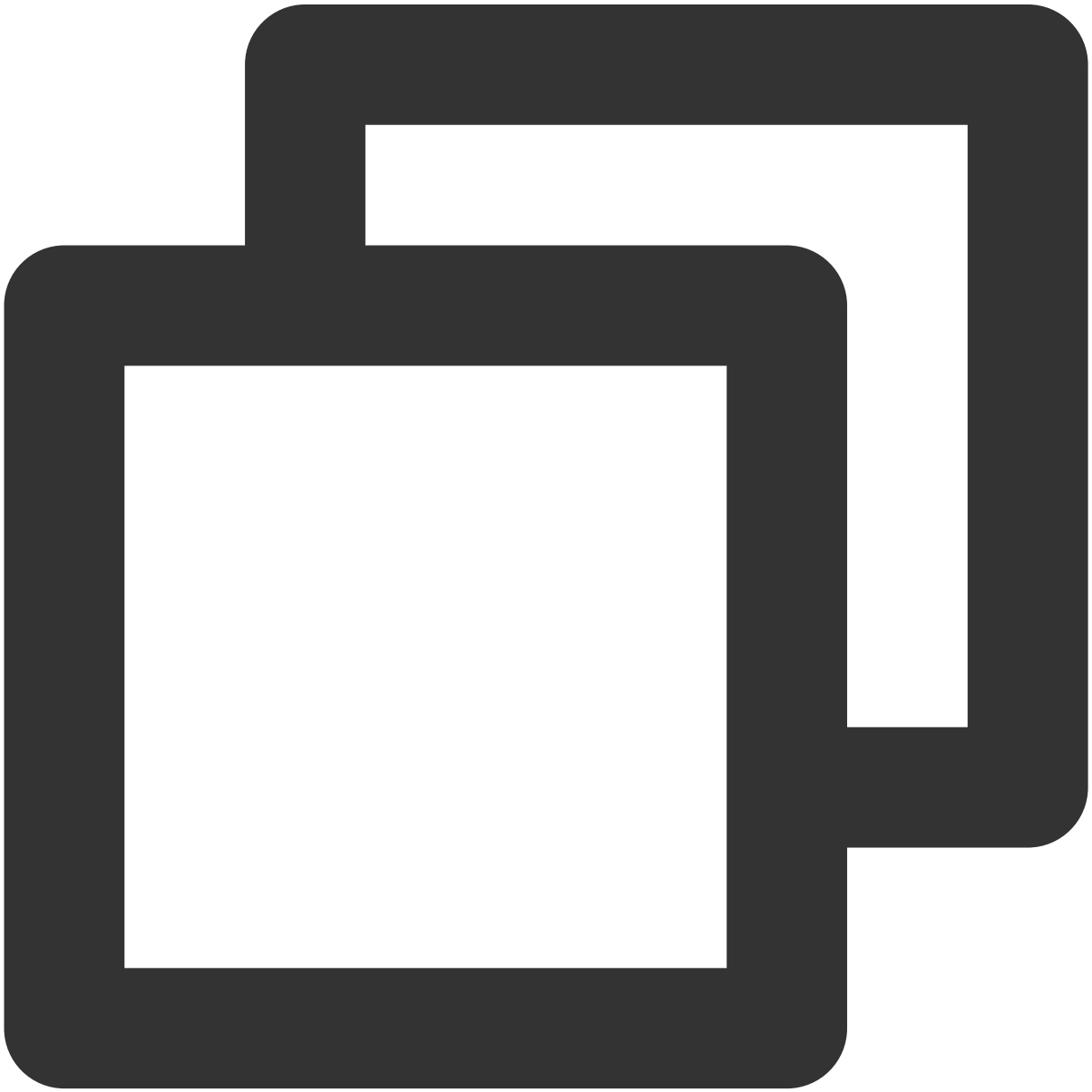
```
var player = TCPlayer("player-container-id", {}); // player-container-idはプレーヤー
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"); // hls
```

サンプルコードを取得します。

HLS暗号化ビデオ再生サンプルコード

解像度を切り替える

Cloud Infiniteのアダプティブビットレートストリーミング機能は、ビデオファイルをトランスコードおよびパッケージ化し、アダプティブビットレートストリーミング出力ファイルを生成することで、ユーザーがさまざまなネットワーク状態の下でビデオコンテンツをスピーディーに配信できるようサポートするものです。プレーヤーも現在の帯域幅に応じて、最適なビットレートを動的に選択して再生できるようになります。
操作手順は次のとおりです。
1. Cloud Infiniteのアダプティブビットレートストリーミング機能により、マルチビットレートアダプティブなHLSまたはDASHターゲットファイルを生成します。
2. プレーヤーを初期化し、ビデオオブジェクトアドレスを渡します。



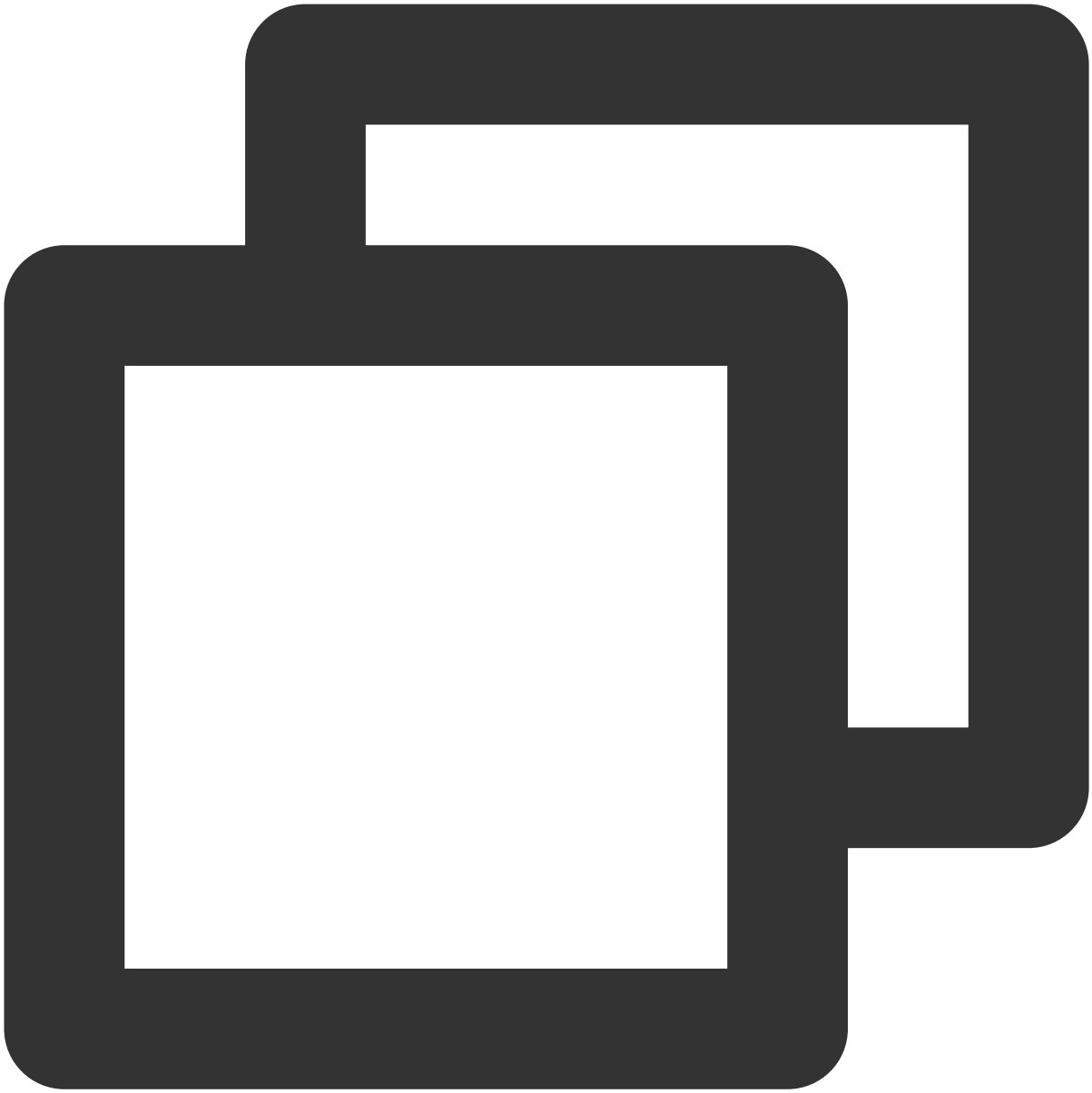
```
var player = TCPlayer("player-container-id", {}); // player-container-idはプレーヤー
```

```
player.src("https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"); // マル
```

サンプルコードを取得します。
解像度切り替えサンプルコード

動的ウォーターマークを設定する

プレーヤーはビデオに、位置と速度が変化するウォーターマークを追加することができます。動的ウォーターマーク機能を使用する際、プレーヤーによるオブジェクトの参照をグローバル環境に公開すると、動的ウォーターマークが簡単に除去できるため、公開してはなりません。Cloud Infiniteでは、クラウド側でのビデオに対する動的ウォーターマーク追加などの操作もサポートしています。詳細についてはウォーターマークテンプレートインターフェースをご参照ください。



```
var player = TCPlayer("player-container-id", {
    plugins:{
      DynamicWatermark: {
        speed: 0.2, // 速度
        content: "Tencent Cloud Infinite CI", // テキスト
        opacity: 0.7 // 透明度
      }
    }
  });
```

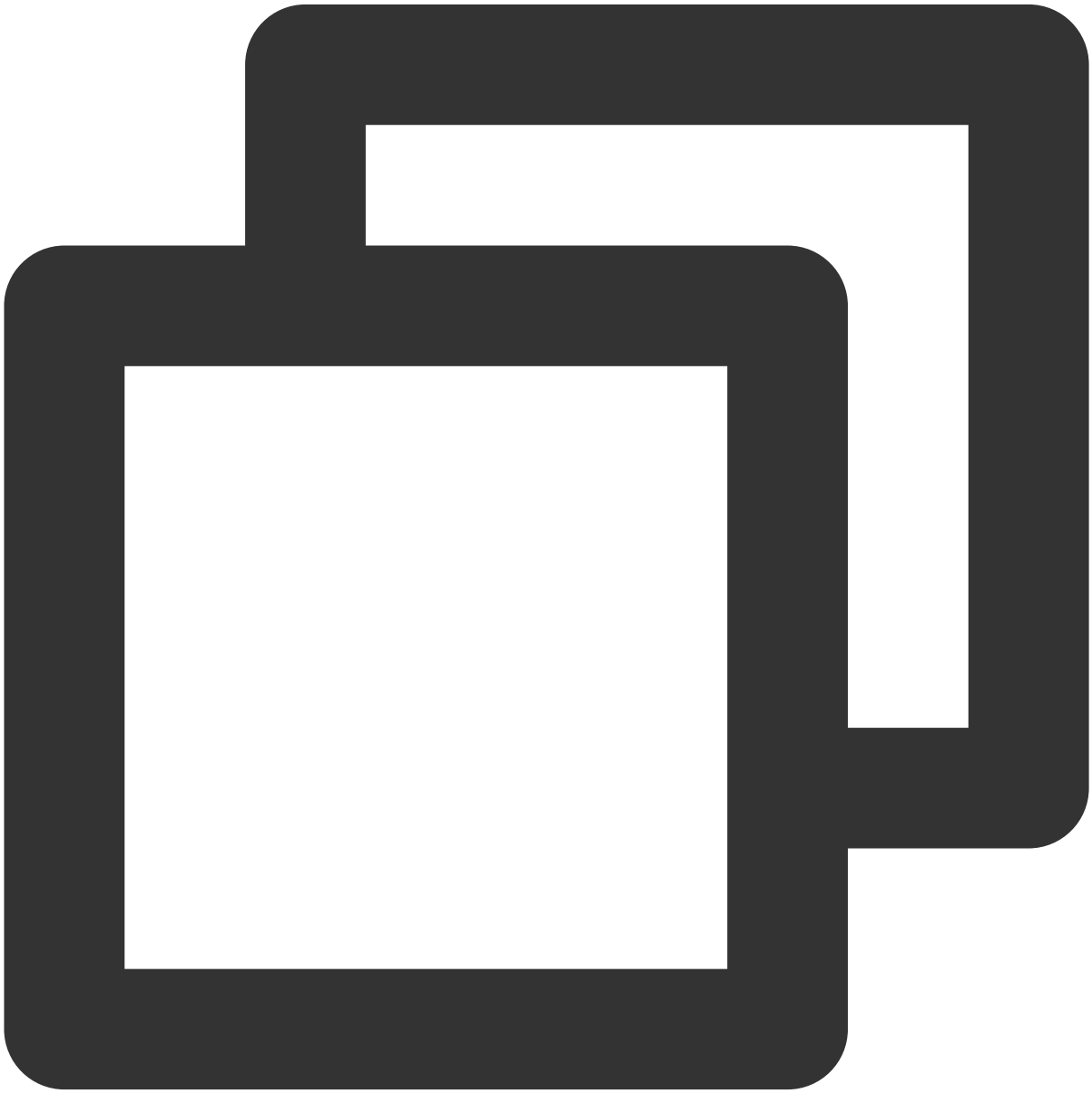
サンプルコードを取得します。

動的ウォーターマークの設定

ロール画像広告を設定する

操作手順は次のとおりです。

1. ビデオ広告のカバー画像および広告リンクを準備します。
2. プレーヤーを初期化し、広告のカバー画像およびリンクを設定し、広告ノードを設定します。



```
var PosterImage = TCPlayer.getComponent('PosterImage');
PosterImage.prototype.handleClick = function () {
```

```
 window.open('https://www.tencentcloud.com/products/ci'); // 広告リンクの設定
};

var player = TCPlayer('player-container-id', {
 poster: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx..png', // 広告カバ
});
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');

var adTextNode = document.createElement('span');
adTextNode.className = 'ad-text-node';
adTextNode.innerHTML = '広告';

var adCloseIconNode = document.createElement('i');
adCloseIconNode.className = 'ad-close-icon-node';
adCloseIconNode.onclick = function (e) {
  e.stopPropagation();
  player.posterImage.hide();
};

player.posterImage.el_.appendChild(adTextNode);
player.posterImage.el_.appendChild(adCloseIconNode);
```

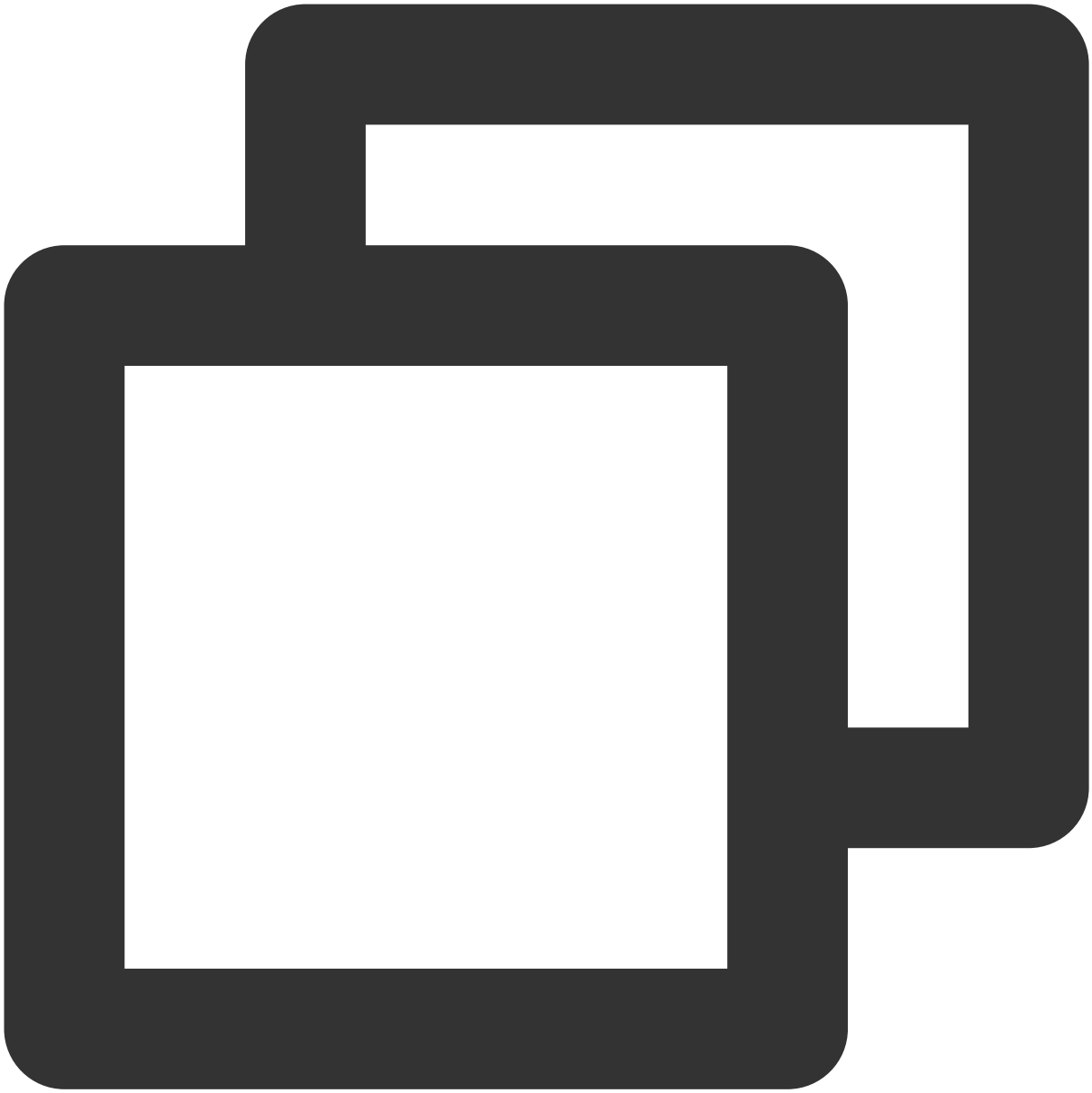
サンプルコードを取得します。

ロール画像広告設定サンプルコード

ビデオ進捗画像を設定する

操作手順は次のとおりです。

1. Cloud Infiniteでビデオフレームキャプチャを行い、スプライトイメージを生成します。
2. ステップ1で生成したスプライトイメージとVTT（スプライトイメージの位置記述ファイル）のオブジェクトアドレスを取得します。
3. プレーヤーを初期化し、ビデオアドレスとVTTファイルを設定します。



```
var player = TCPlayer('player-container-id', {
  plugins: {
 VttThumbnail: {
   vttUrl: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.vtt' // 進捗画像
 },
  },
});
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');
```

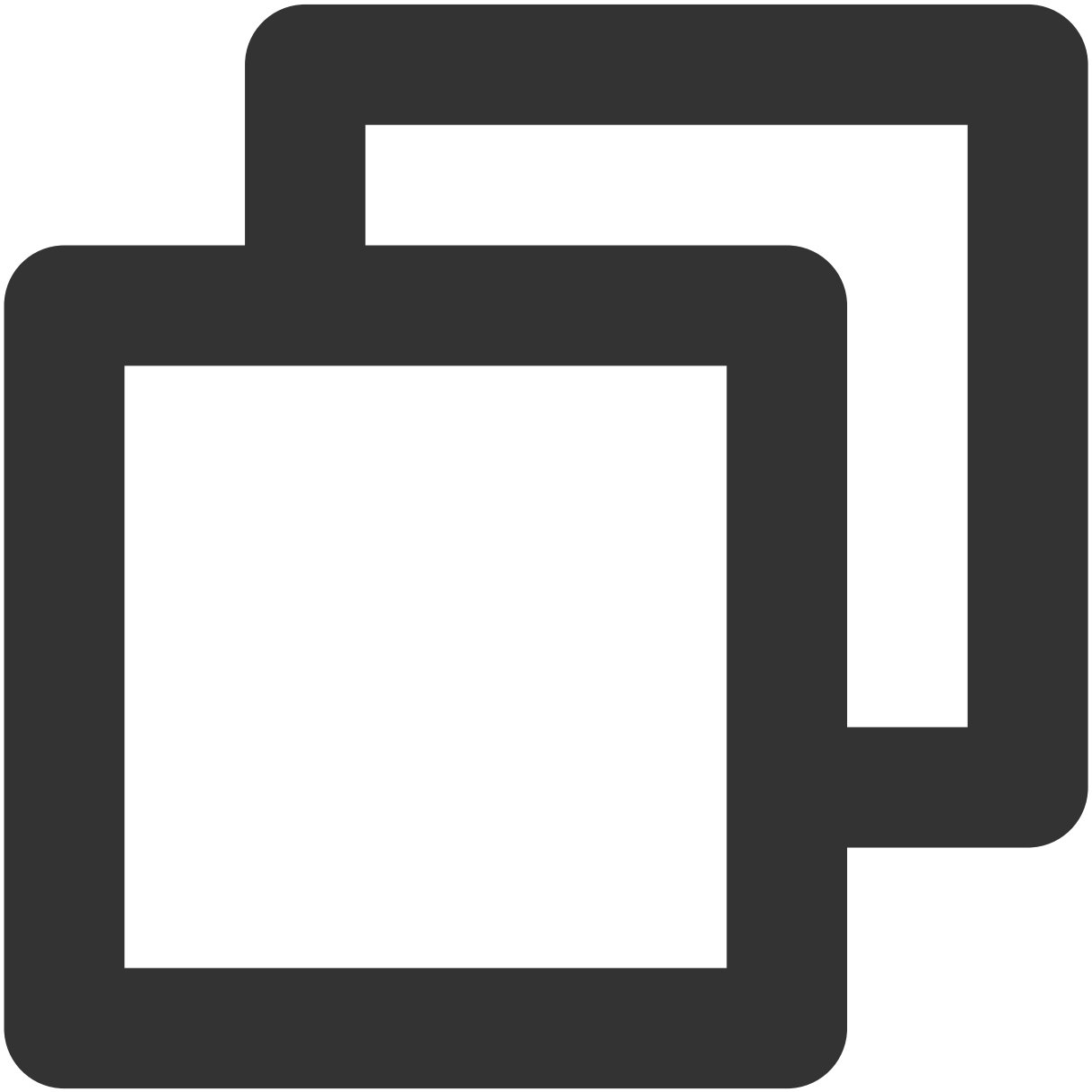
サンプルコードを取得します。

ビデオ進捗画像設定サンプルコード

ビデオ字幕を設定する

操作手順は次のとおりです。

1. Cloud Infiniteによって音声認識を行い、字幕ファイルを生成します。
2. ステップ1で生成した字幕SRTファイルのオブジェクトアドレスを取得します。
3. プレーヤーを初期化し、ビデオアドレスと字幕SRTファイルを設定します。



```
var player = TCPlayer('player-container-id', {});
```

```
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');
player.on('ready', function () {
  // 字幕ファイルの追加
  var subTrack = player.addRemoteTextTrack({
 src: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.srt', // 字幕ファイル
 kind: 'subtitles',
 srclang: 'zh-cn',
 label: '中国語',
 default: 'true',
  }, true);
});
```

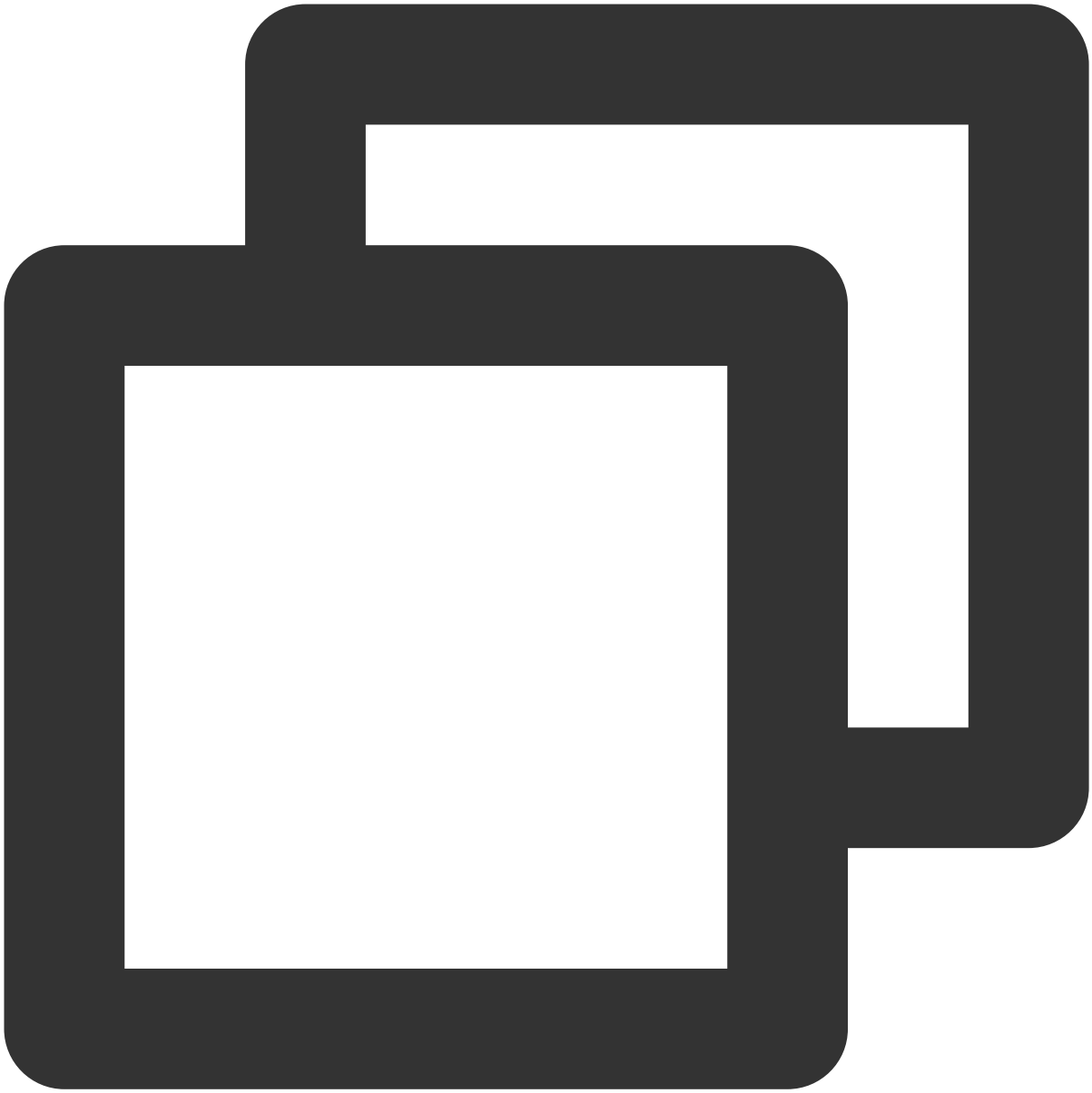
サンプルコードを取得します。

ビデオ字幕設定サンプルコード

ビデオ多言語字幕を設定する

操作手順は次のとおりです。

1. Cloud Infiniteによって音声認識と字幕ファイル生成を行い、同時に多言語に翻訳します。
2. ステップ1で生成した多言語字幕SRTファイルのオブジェクトアドレスを取得します。
3. プレーヤーを初期化し、ビデオアドレスと多言語字幕SRTファイルを設定します。



```
var player = TCPlayer('player-container-id', {});
player.src('https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4');
player.on('ready', function () {
  // 中国語字幕の設定
  var subTrack = player.addRemoteTextTrack({
 src: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/zh.srt', // 字幕ファイル
 kind: 'subtitles',
 srclang: 'zh-cn',
 label: '中国語',
 default: 'true',
  }, true);
```

```
    // 英語字幕の設定
    var subTrack = player.addRemoteTextTrack({
   src: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/en.srt', // 字幕ファイル
   kind: 'subtitles',
   srclang: 'en',
   label: '英語',
   default: 'false',
    }, true);
 });
```

サンプルコードを取得します。

ビデオ多言語字幕設定サンプルコード

DPlayerを使用したCOSビデオファイルの再生

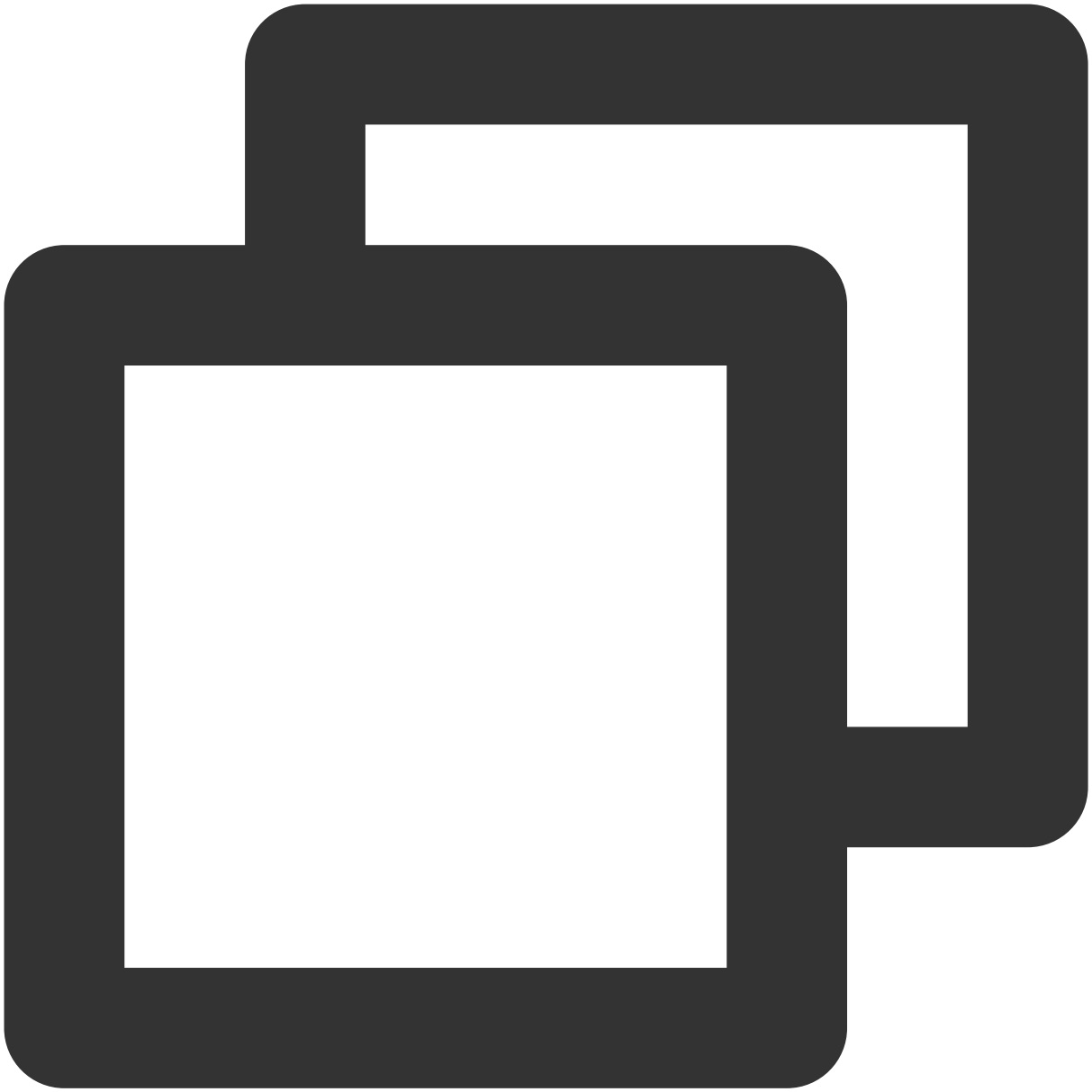
最終更新日：：2024-06-26 10:42:27

概要

ここでは、DPlayerを使用し、Tencent Cloud Infinite(CI)の提供する豊富なオーディオビデオ機能と組み合わせて、WebブラウザでCOSビデオファイルを再生する方法についてご説明します。

統合ガイド

ステップ1：ページにプレーヤースクリプトファイルおよび必要に応じて一部依存ファイルをインポートする



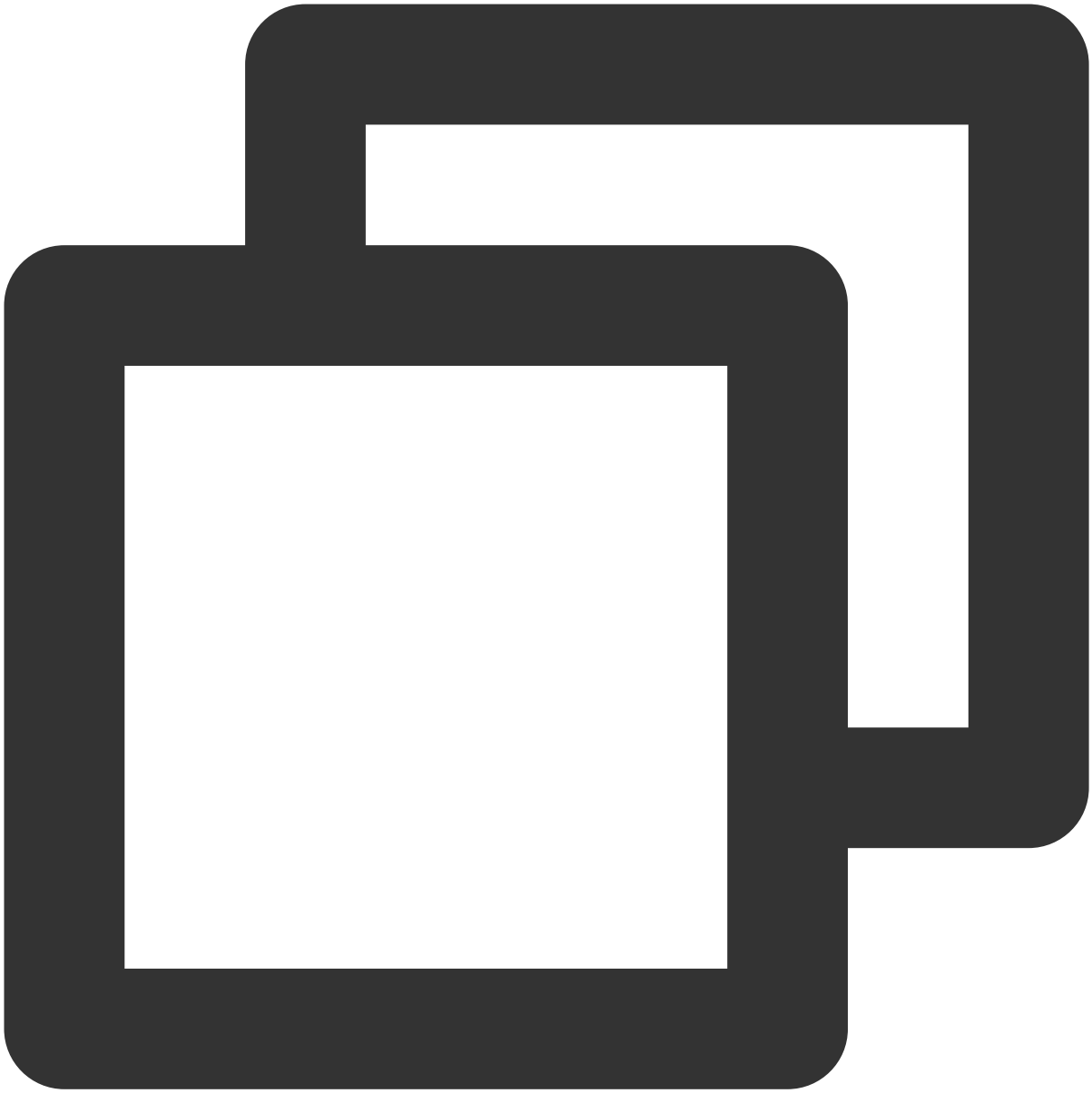
```
<!--プレーヤースクリプトファイル-->
<script src="https://cdn.jsdelivr.net/npm/dplayer@1.26.0/dist/DPlayer.min.js"></scr
```

説明：

プレーヤーを正式に使用する際は、ご自身で上記の静的リソースをデプロイすることをお勧めします。

ステップ2：プレーヤーコンテナノードを設定する

プレーヤーを表示したいページ位置にプレーヤーコンテナを追加します。例えば、index.htmlに次のコードを追加します（コンテナIDおよび幅と高さはいずれもカスタマイズできます）。



```
<div id="dplayer" style="width: 100%; height: 100%"></div>
```

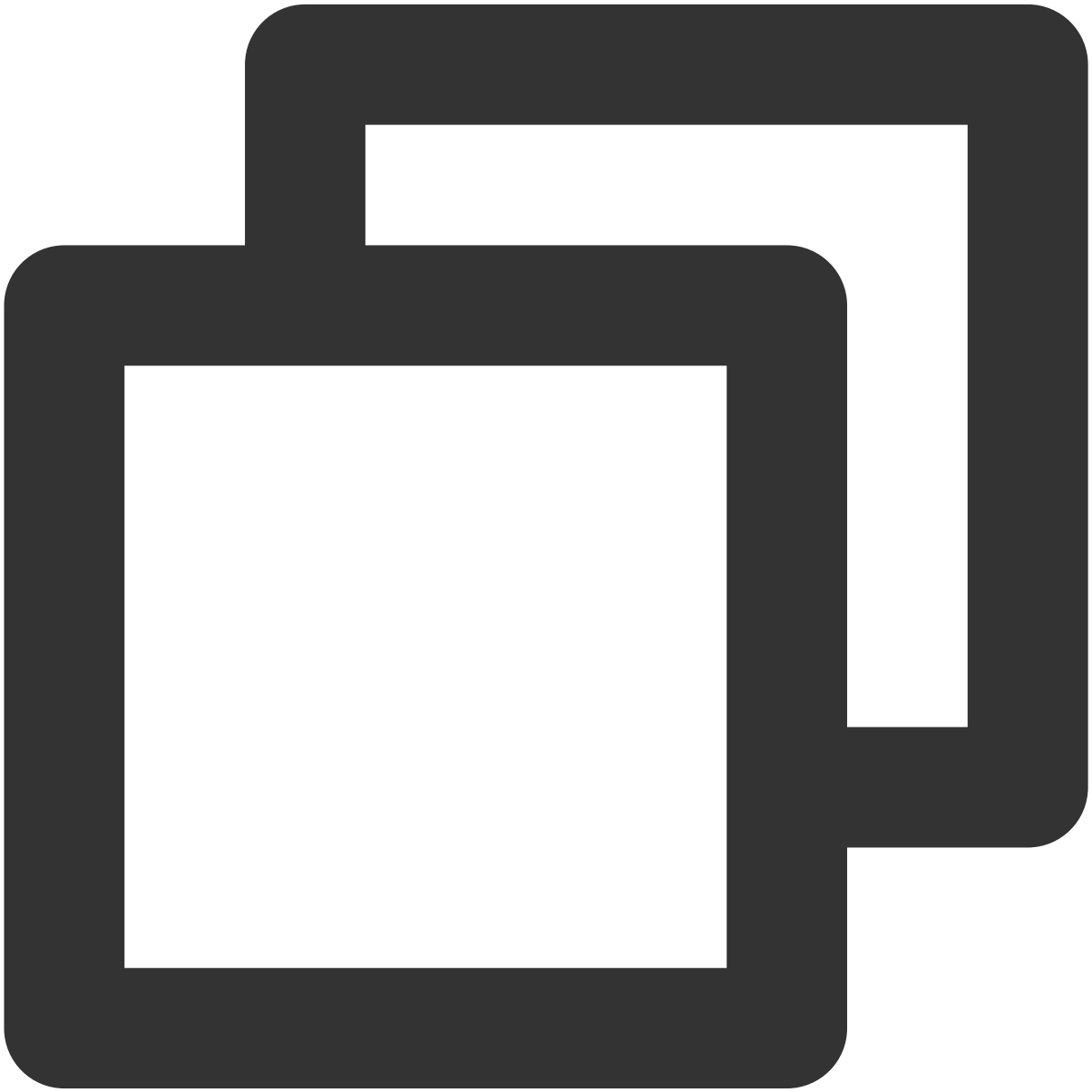
ステップ3：ビデオファイルオブジェクトアドレスを取得する

1. バケットの作成を行います。
2. ビデオファイルのアップロードを行います。
3. ビデオファイルのオブジェクトアドレスを取得します。形式は `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<ビデオ形式>` です。

説明：

クロスドメインの問題がある場合は、バケットのクロスドメインアクセスCORS設定を行う必要があります。詳細については、クロスドメインアクセスの設定をご参照ください。
バケットがプライベート読み取り/書き込みの場合、オブジェクトアドレスには署名が必要です。詳細については、リクエスト署名をご参照ください。

ステップ4：プレーヤーを初期化し、COSビデオファイルのオブジェクトアドレスURLを渡す



```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  video: {
```

```
        url: https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"); // COSビ
    },
});
```

機能ガイド

さまざまな形式のビデオファイルを再生する

1. COSバケット上のビデオファイルのオブジェクトアドレスを取得します。

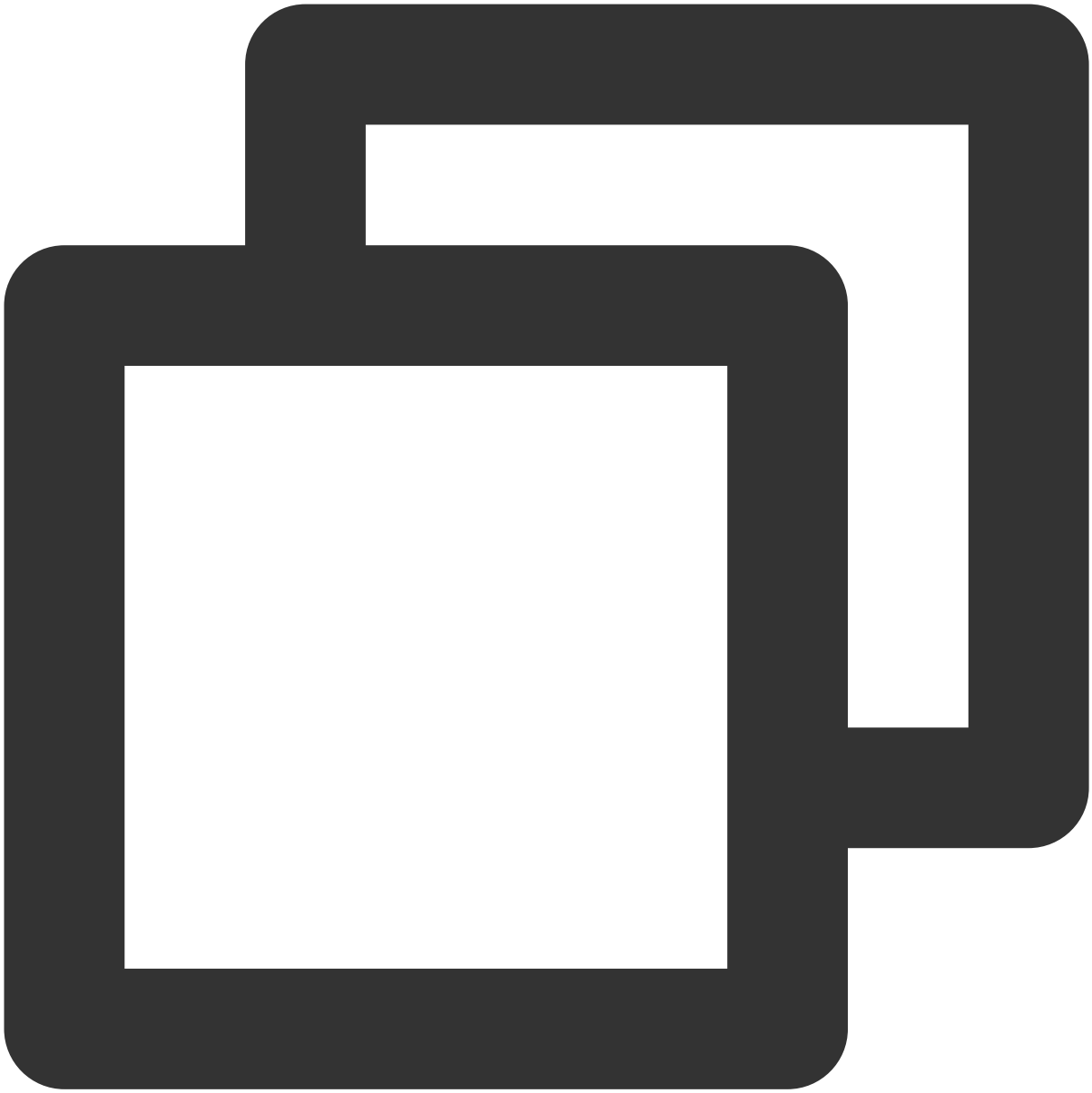
説明：

トランスコードされていないソースビデオは再生の際に互換性の問題が生じる可能性があるため、トランスコード後のビデオで再生を行うことをお勧めします。Cloud Infiniteのオーディオビデオトランスコーディング処理により、さまざまな形式のビデオファイルを取得することができます。

2. さまざまなビデオ形式に対し、マルチブラウザでの互換性を保証するためには、対応する依存関係の導入が必要です。

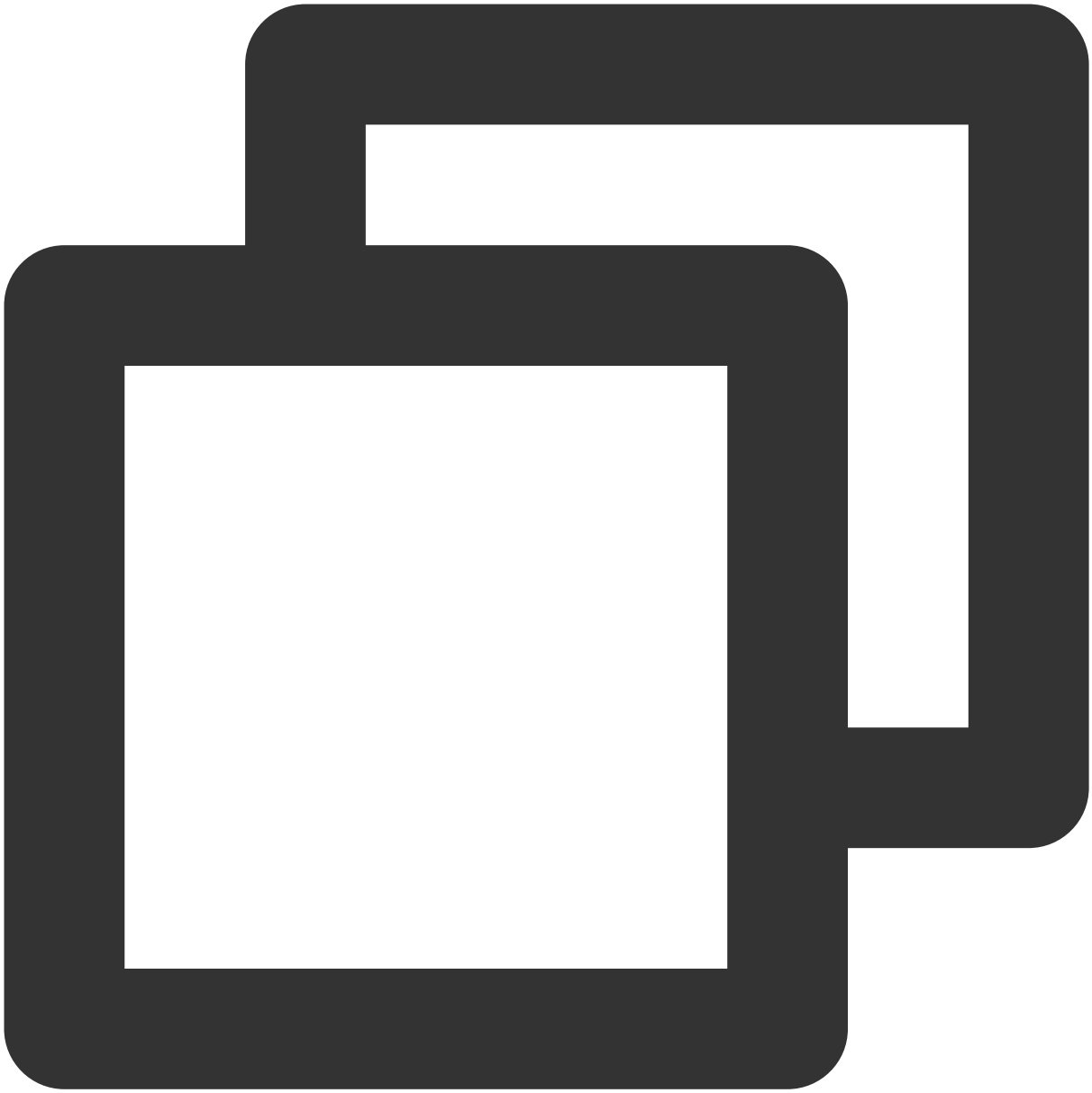
MP4：他の依存関係を導入する必要はありません。

HLS：ChromeやFirefoxなどの最新ブラウザでH5を介してHLS形式のビデオを再生したい場合は、 tcplayer.min.jsの前にhls.min.jsを導入する必要があります。



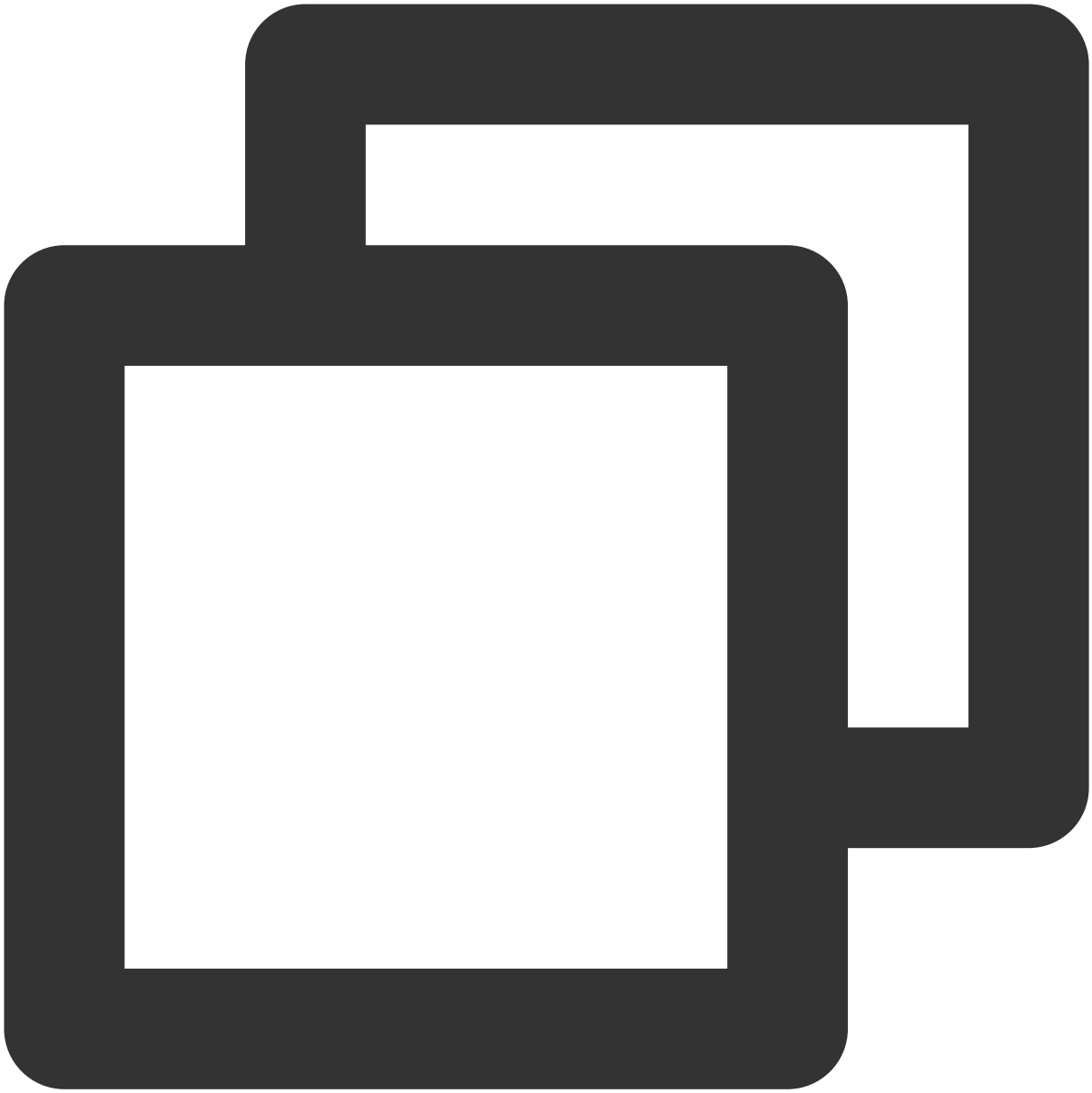
```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/libs/hls.m
```

FLV：ChromeやFirefoxなどの最新ブラウザでH5を介してFLV形式のビデオを再生したい場合は、 tcplayer.min.js の前にflv.min.jsを導入する必要があります。



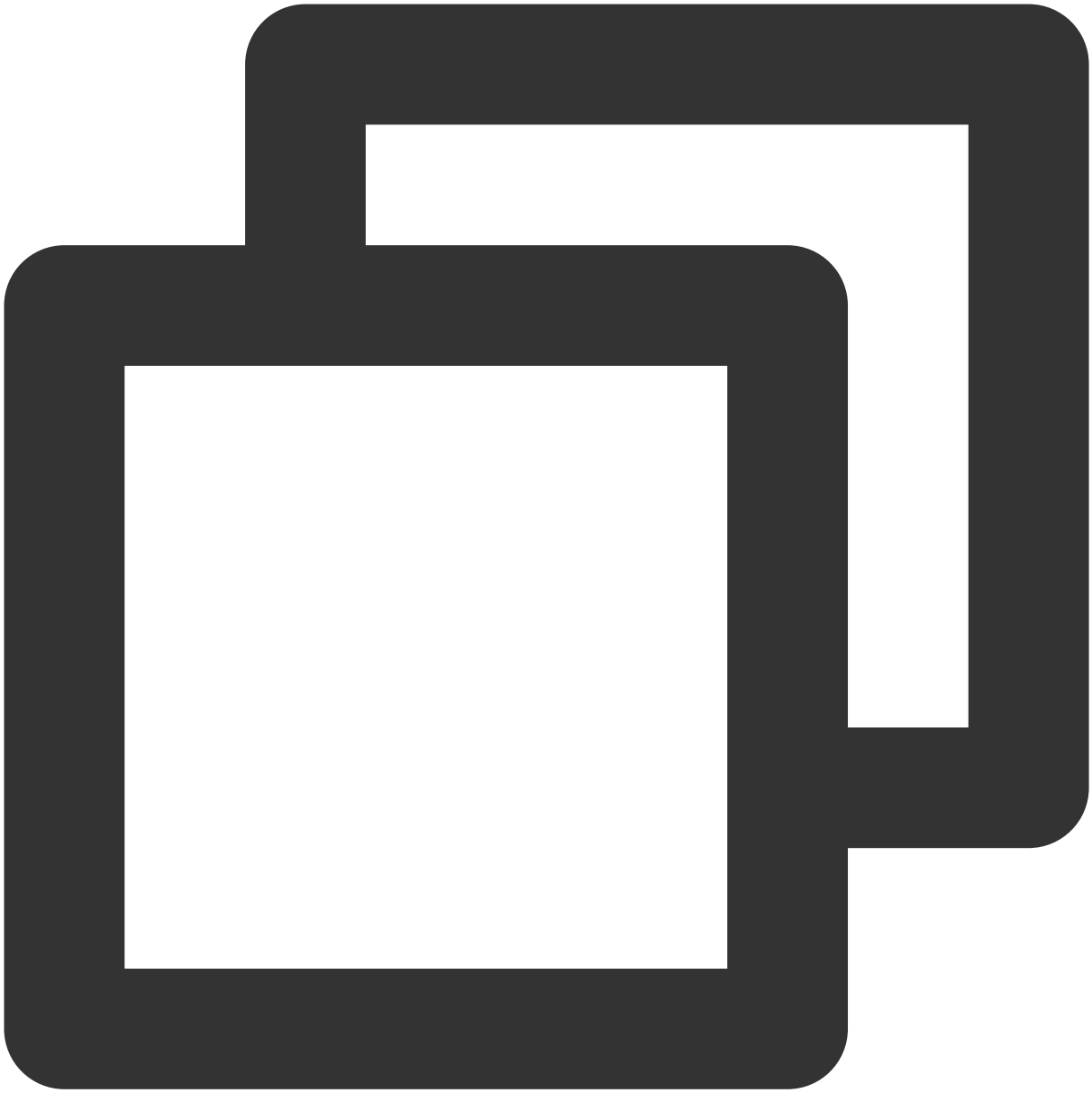
```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.m
```

DASH：DASHビデオにはdash.all.min.jsファイルのロードが必要です。



```
<script src="https://cos-video-1258344699.cos.ap-guangzhou.myqcloud.com/lib/dash.
```

3. プレーヤーを初期化し、オブジェクトアドレスを渡します。



```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  video: {
   url: https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"); // COSビデオ
  },
});
```

サンプルコードを取得します。

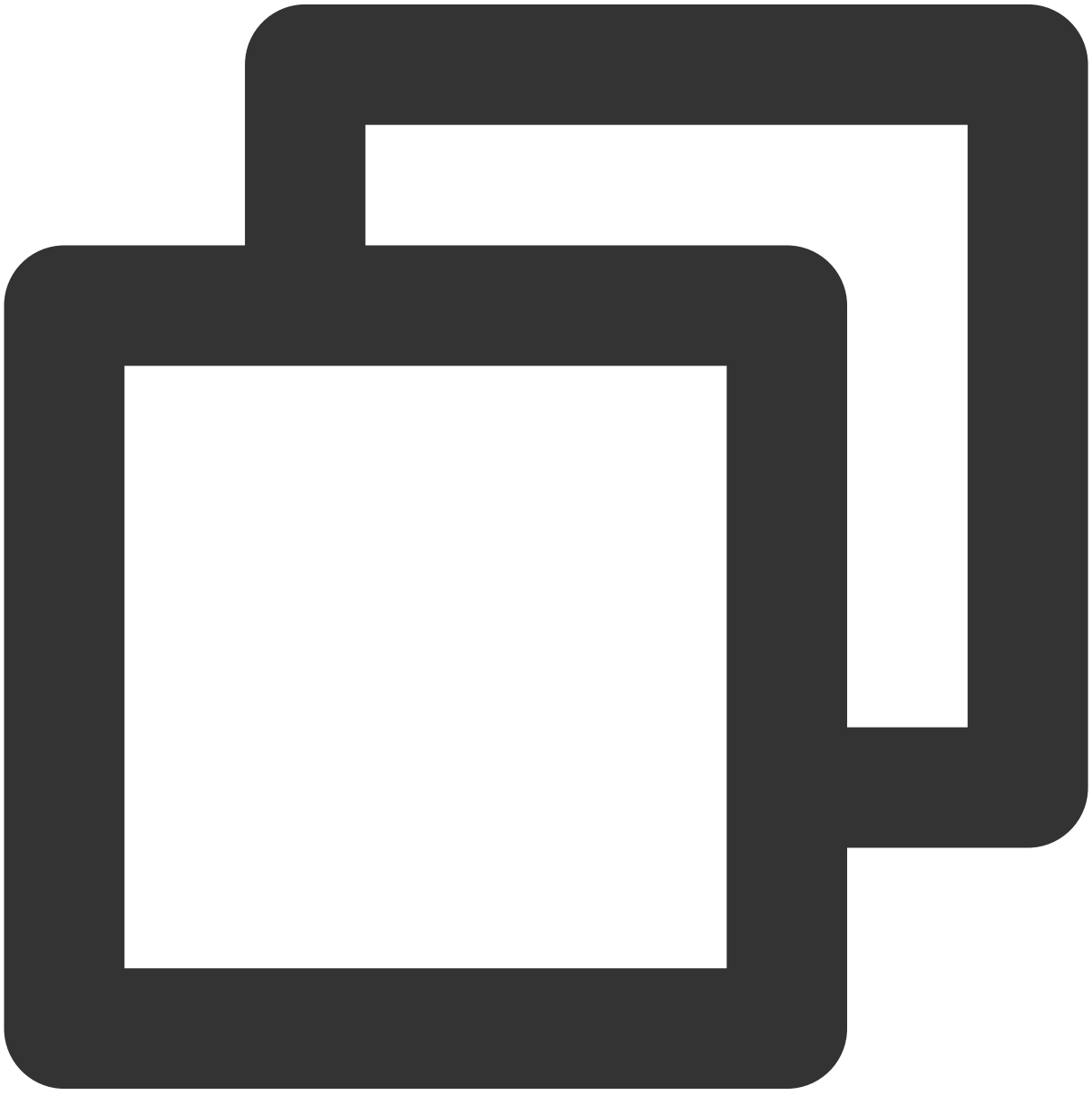
MP4再生サンプルコード

FLV再生サンプルコード

HLS再生サンプルコード
DASH再生サンプルコード

PM3U8ビデオを再生する

PM3U8はプライベートM3U8ビデオファイルのことです。COSはプライベートM3U8 TSリソースの取得に用いるダウンロード権限承認APIを提供しています。プライベートM3U8インターフェースをご参照ください。



```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
```

```
    // pm3u8の詳細については、関連ドキュメント（https://www.tencentcloud.com/document/prod
    video: {
      url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8?ci-process
    }
  });
```

サンプルコードを取得します。

PM3U8再生サンプルコード

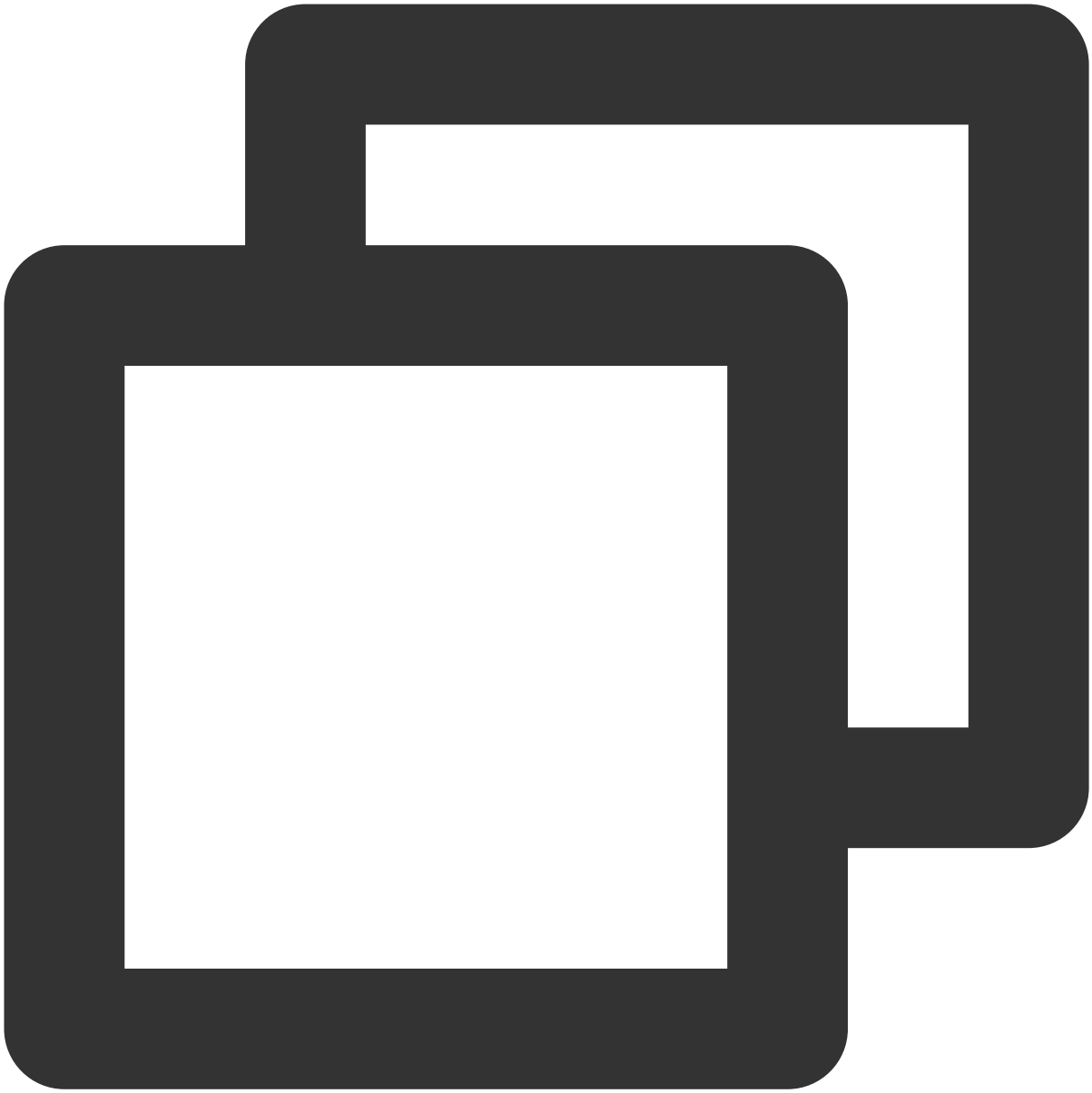
カバー画像を設定する

1. COSバケット上のカバー画像のオブジェクトアドレスを取得します。

注意：

Cloud Infiniteインテリジェントカバー機能により、最適なフレームを抽出してスクリーンキャプチャを生成し、それをカバーにすることでコンテンツの魅力をアップさせます。

2. プレーヤーを初期化し、カバー画像を設定します。



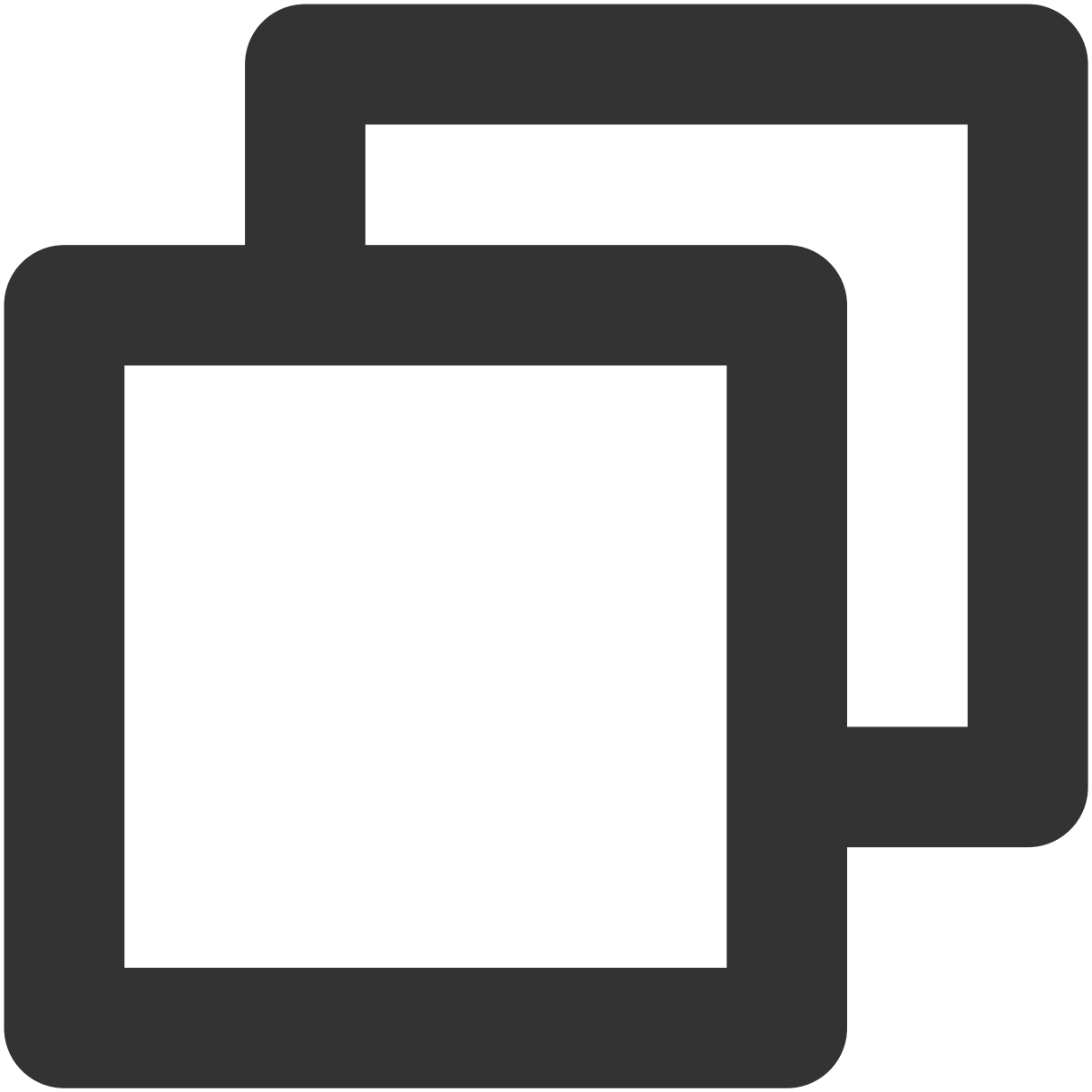
```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  video: {
 url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4',
 pic: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.png',
  },
});
```

サンプルコードを取得します。

カバー画像設定サンプルコード

HLS暗号化ビデオを再生する

ビデオコンテンツの安全性を保障し、ビデオの違法なダウンロードと拡散を防止するため、Cloud InfiniteはHLSビデオコンテンツに対する暗号化機能を提供します。この機能はプライベート読み取りファイルよりさらに高いセキュリティレベルを有します。暗号化されたビデオは、アクセス権限のないユーザーの視聴用に配信できなくなります。ビデオがローカルにダウンロードされた場合でも、ビデオ自体が暗号化されているため、悪意ある二次配信が不可能であり、ビデオの著作権が違法に侵害されないよう保障することができます。
操作手順は次のとおりです。
1. HLS暗号化ビデオの再生およびCOSオーディオビデオ実践 | ビデオをロックするのフローを参照し、暗号化ビデオを生成します。
2. プレーヤーを初期化し、ビデオオブジェクトアドレスを渡します。



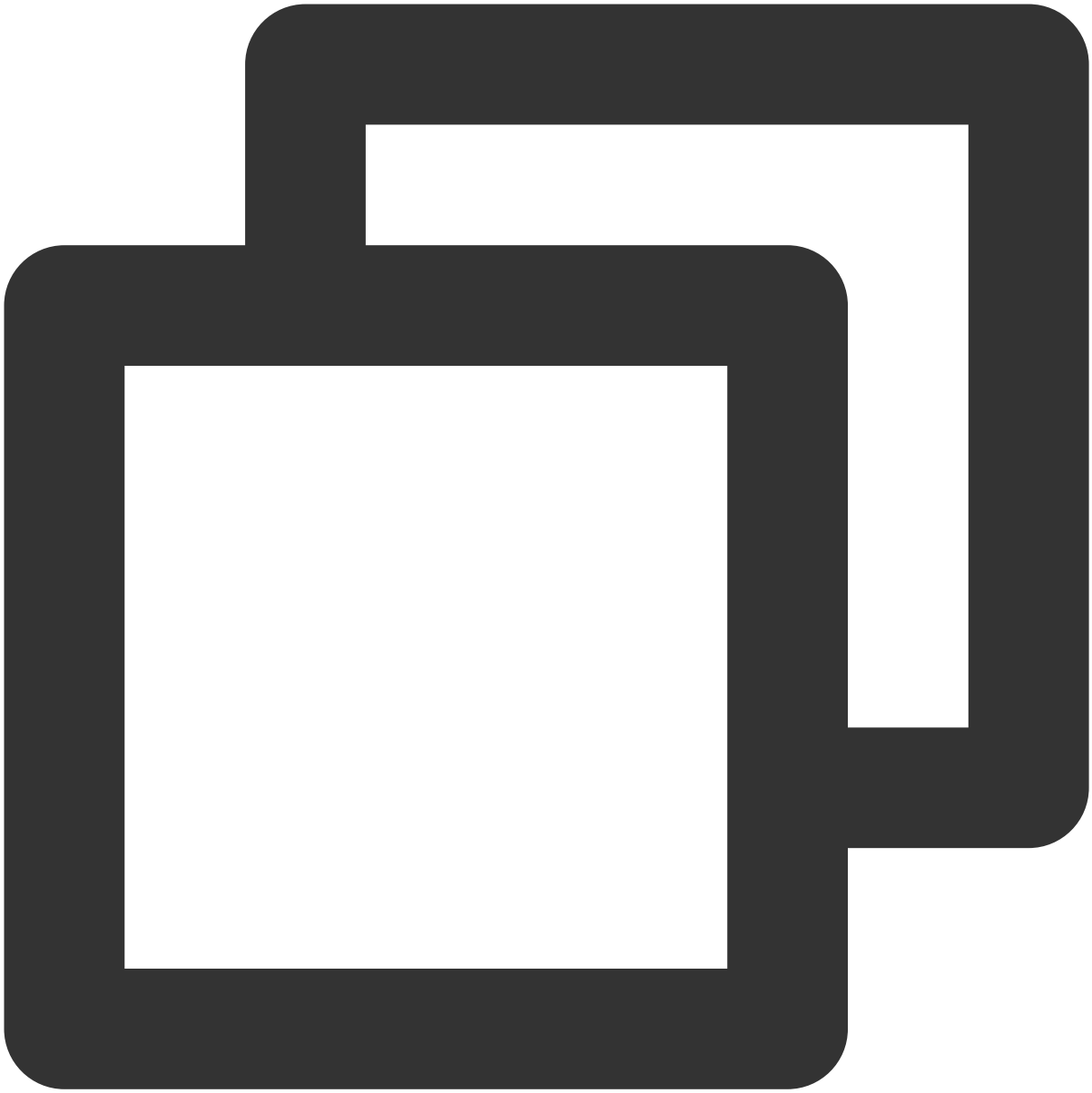
```
 const dp = new DPlayer({
container: document.getElementById('dplayer'),
video: {
  url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8' // 暗号化ビデ
}
 });
```

サンプルコードを取得します。

HLS暗号化ビデオ再生サンプルコード

マルチ解像度切り替え

Cloud Infiniteのアダプティブビットレートストリーミング機能は、ビデオファイルをトランスコードおよびパッケージ化し、アダプティブビットレートストリーミング出力ファイルを生成することで、ユーザーがさまざまなネットワーク状態の下でビデオコンテンツをスピーディーに配信できるようサポートするものです。プレーヤーも現在の帯域幅に応じて、最適なビットレートを動的に選択して再生できるようになります。詳細については、COSオーディオビデオ実践 | データワークフローによるマルチ解像度ビデオ再生のサポートをご参照ください。
操作手順は次のとおりです。
1. Cloud Infiniteのアダプティブビットレートストリーミング機能によって、マルチビットレートのアダプティブなHLSまたはDASHターゲットファイルを生成します。
2. プレーヤーを初期化し、ビデオオブジェクトアドレスを渡します。



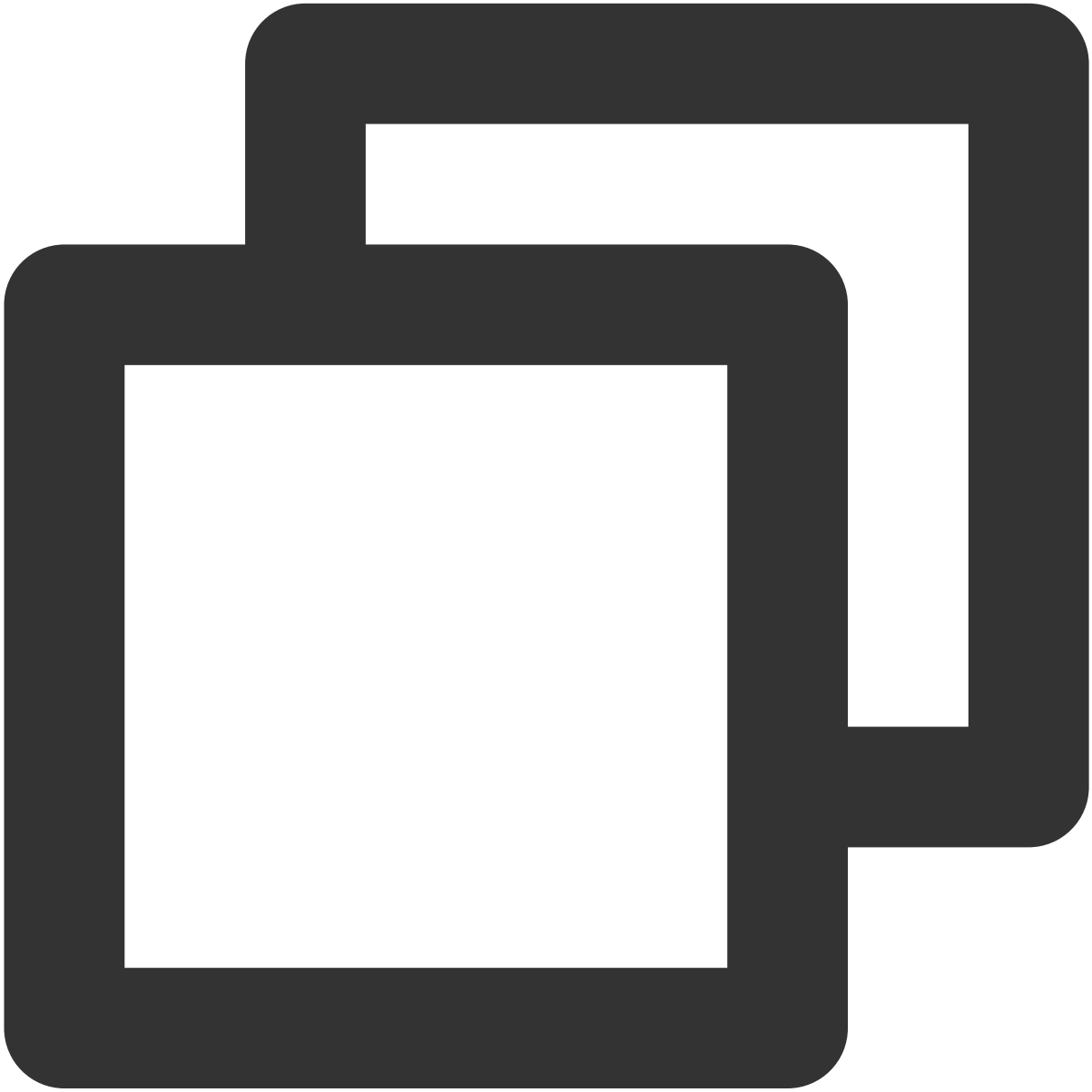
```
const dp = new DPlayer({
  container: document.getElementById('dplayer'),
  video: {
   url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8', //  マルチヒ
  },
});
```

サンプルコードを取得します。

解像度切り替えサンプルコード

左上隅のロゴ設定

プレーヤーは左上隅へのロゴ設定をサポートしています。
操作手順は次のとおりです。
1. COSバケット上のロゴアイコンのオブジェクトアドレスを取得します。
2. プレーヤーを初期化し、ロゴアイコンを設定します。



```
const dp = new DPlayer({
 container: document.getElementById('dplayer'),
  video: {
   url: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4',
```

```
    },
   logo: 'https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.svg'
  });
```

サンプルコードを取得します。

左上隅へのロゴ設定のサンプルコード

VideojsPlayerを使用したCOSビデオファイルの再生

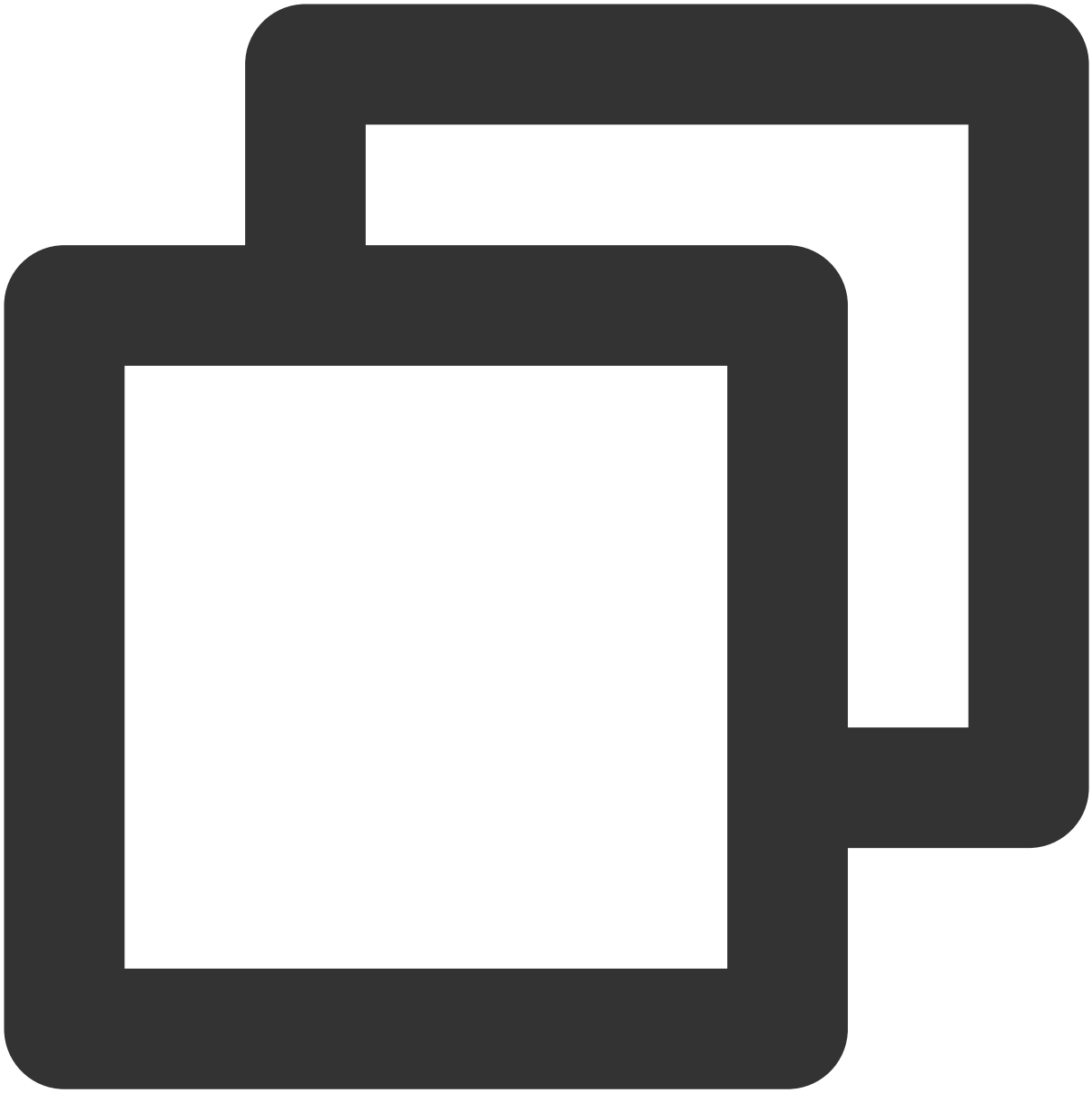
最終更新日：：2024-06-26 10:42:27

概要

ここでは、VideojsPlayerを使用し、Tencent Cloud Infinite(CI)の提供する豊富なオーディオビデオ機能と組み合わせて、WebブラウザでCOSビデオファイルを再生する方法についてご説明します。

統合ガイド

ステップ1：ページにプレーヤースタイルファイルとスクリプトファイルを導入する



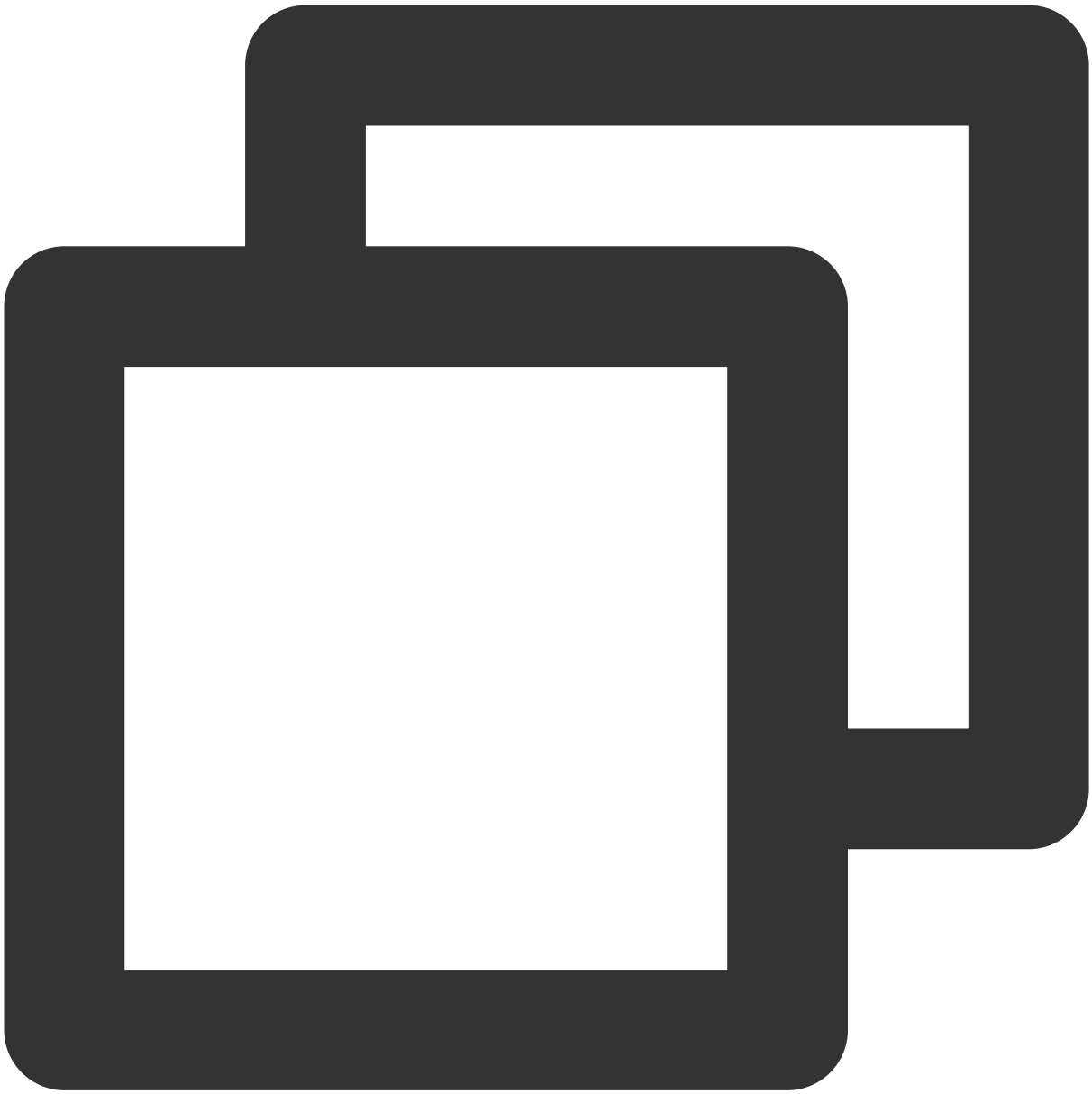
```
<!--プレーヤースタイルファイル-->
<link href="https://vjs.zencdn.net/7.19.2/video-js.css" rel="stylesheet" />
<!--プレーヤースクリプトファイル-->
<script src="https://vjs.zencdn.net/7.19.2/video.min.js"></script>
```

説明：

プレーヤーを正式に使用する際は、ご自身で上記の静的リソースをデプロイすることをお勧めします。

ステップ2：プレーヤーコンテナノードを設定する

プレーヤーを表示したいページ位置にプレーヤーコンテナを追加します。例えば、index.htmlに次のコードを追加します（コンテナIDおよび幅と高さはいずれもカスタマイズできます）。



```
<video
  id="my-video"
  class="video-js"
  controls
  preload="auto"
  width="100%"
  height="100%"
  data-setup="{}"
```

```
></video>
```

ステップ3：ビデオファイルオブジェクトアドレスを取得する

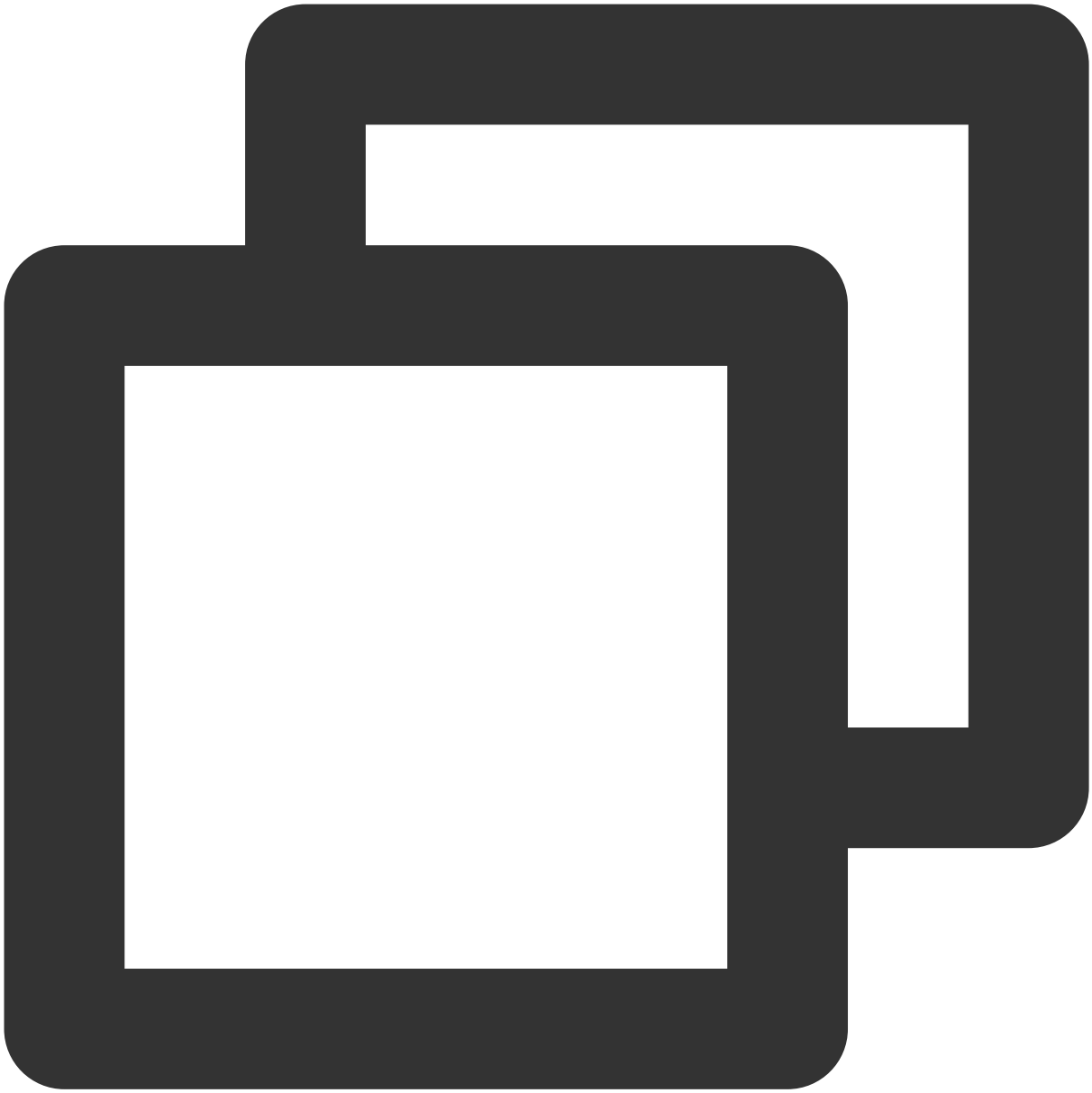
1. バケットの作成を行います。
2. ビデオファイルのアップロードを行います。
3. ビデオファイルのオブジェクトアドレスを取得します。形式は `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.<ビデオ形式>` です。

説明：

クロスドメインの問題がある場合は、バケットのクロスドメインアクセスCORS設定を行う必要があります。詳細については、クロスドメインアクセスの設定をご参照ください。

バケットがプライベート読み取り/書き込みの場合、オブジェクトアドレスには署名が必要です。詳細については、リクエスト署名をご参照ください。

ステップ4：プレーヤーコンテナ内にビデオアドレスを設定し、COSビデオファイルのオブジェクトアドレスURLを渡す



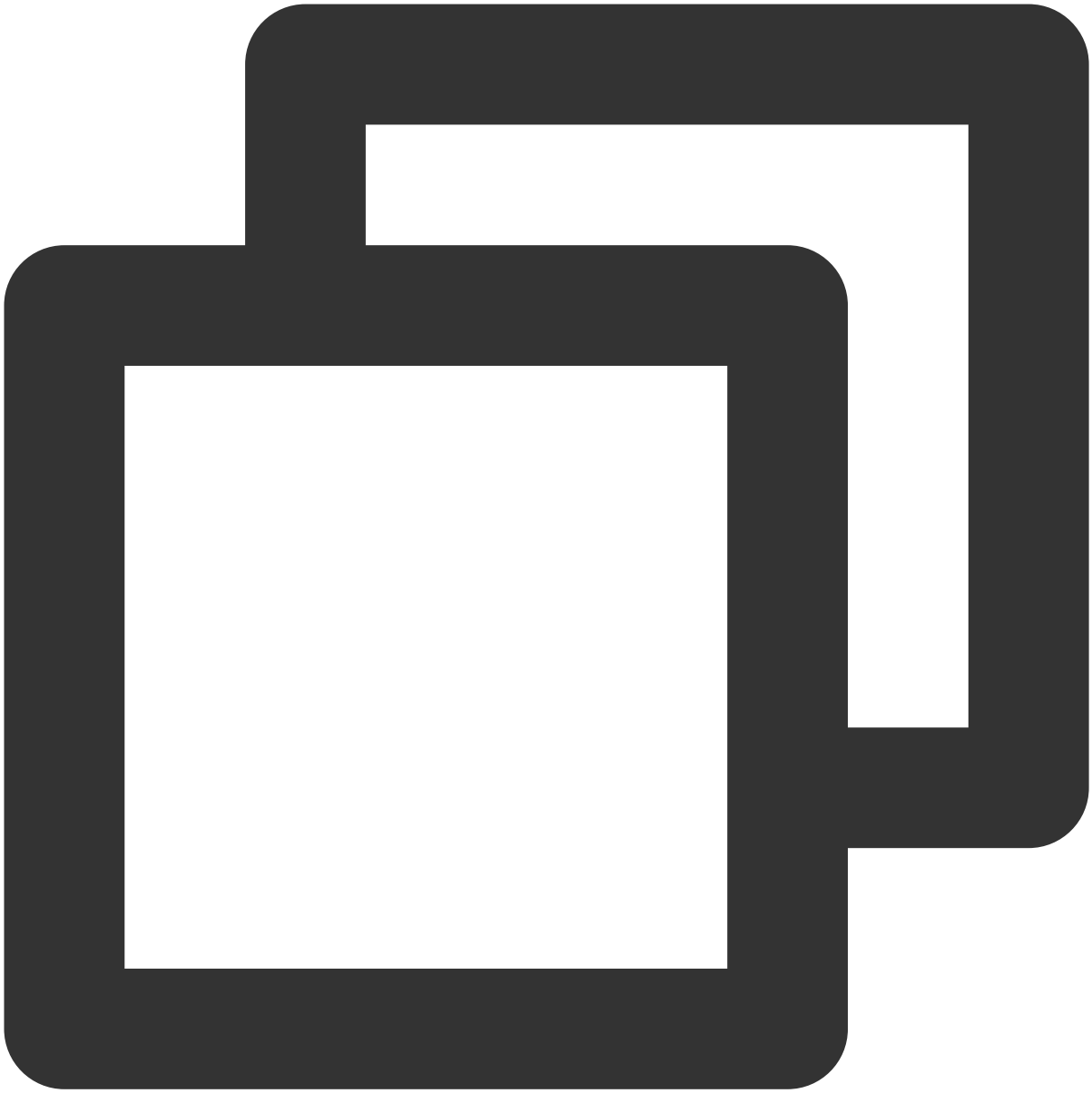
```
<video
  id="my-video"
  class="video-js"
  controls
  preload="auto"
  width="100%"
  height="100%"
  data-setup="{}"
>
  <source
    src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"
```

```
    type="video/mp4"
  />
</video>
```

機能ガイド

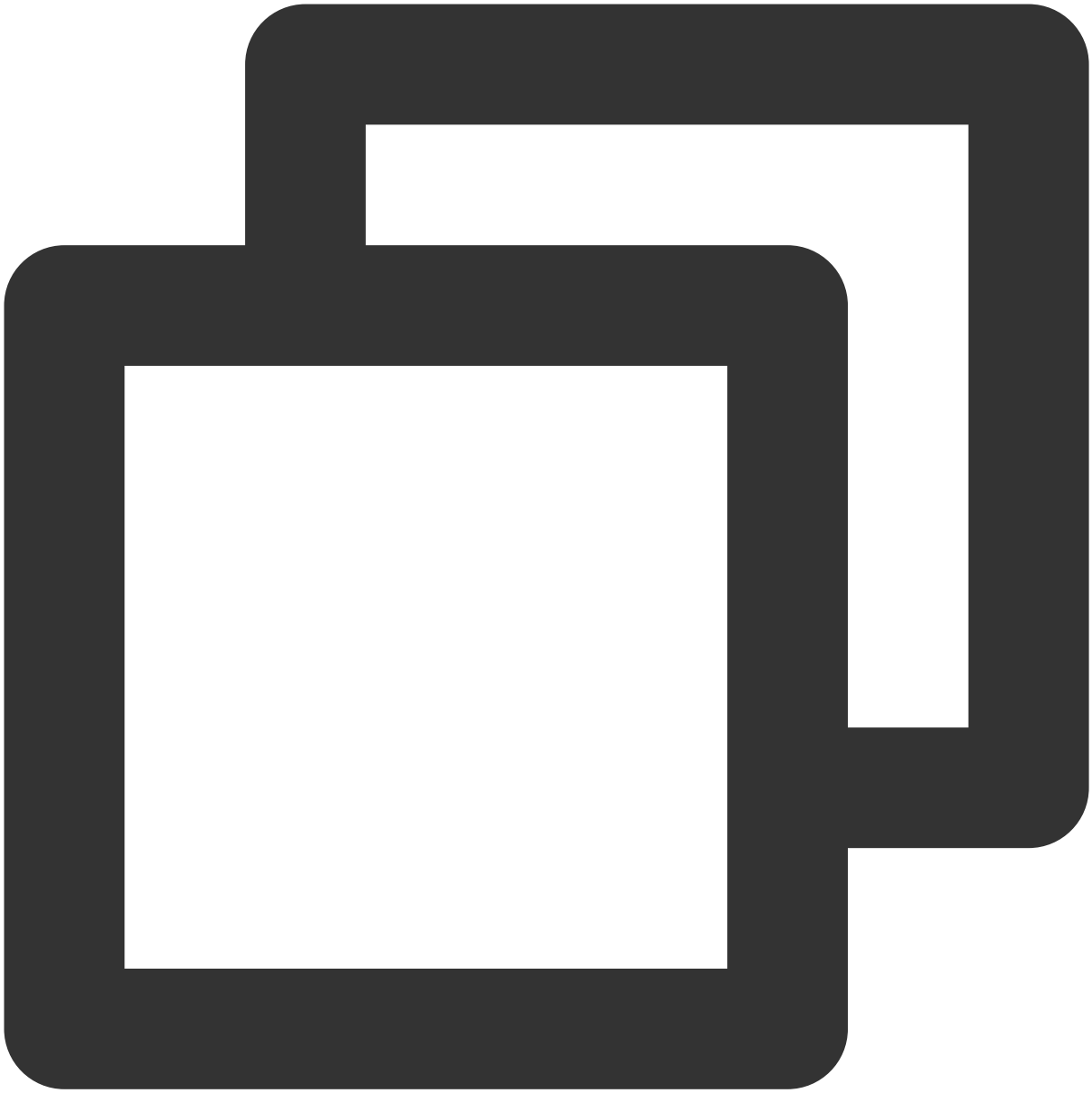
さまざまな形式のビデオファイルを再生する

1. COSバケット上のビデオファイルのオブジェクトアドレスを取得します。
説明：
トランスコードされていないソースビデオは再生の際に互換性の問題が生じる可能性があるため、トランスコード後のビデオで再生を行うことをお勧めします。Cloud Infiniteのオーディオビデオトランスコーディング処理により、さまざまな形式のビデオファイルを取得することができます。
2. さまざまなビデオ形式に対し、マルチブラウザでの互換性を保証するためには、対応する依存関係の導入が必要です。
MP4：他の依存関係を導入する必要はありません。
HLS：ChromeやFirefoxなどの最新ブラウザでH5を介してHLS形式のビデオを再生したい場合は、 tcplayer.min.jsの前にhls.min.jsを導入する必要があります。



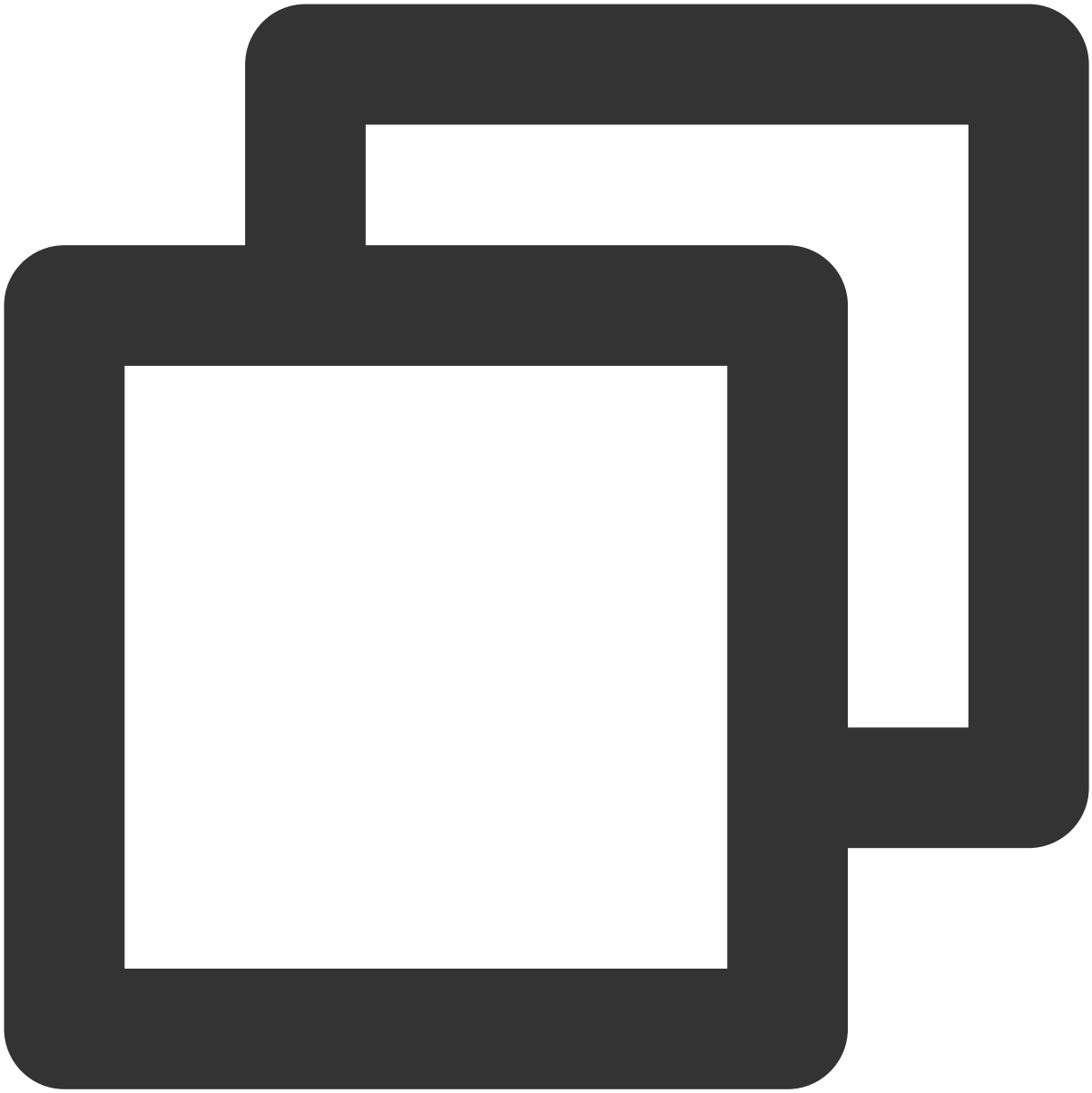
```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.2.1/libs/hls.m
```

FLV：ChromeやFirefoxなどの最新ブラウザでH5を介してFLV形式のビデオを再生したい場合は、 tcplayer.min.js の前にflv.min.jsを導入する必要があります。



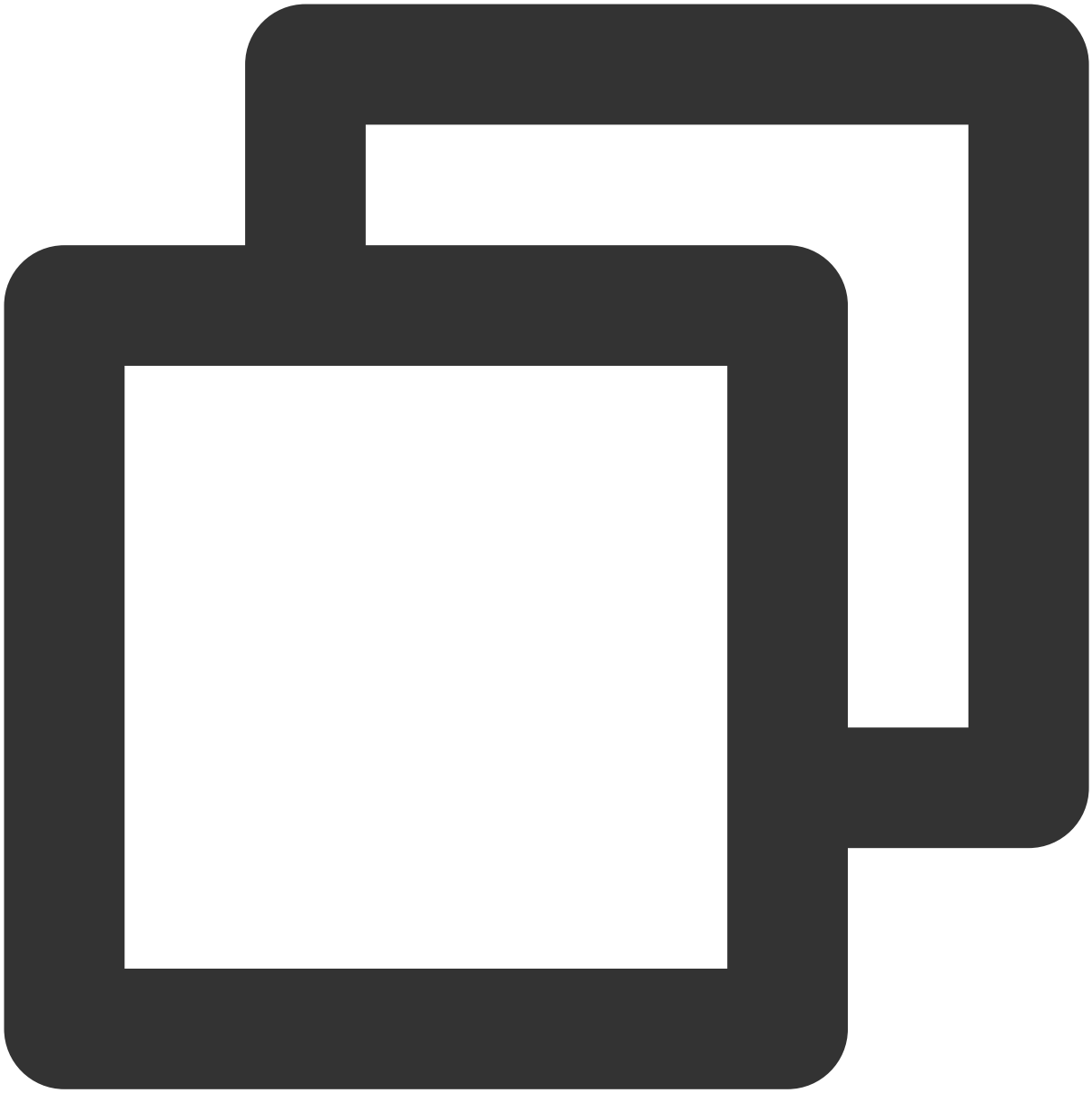
```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.m
```

DASH：DASHビデオにはdash.all.min.jsファイルのロードが必要です。



```
<script src="https://cos-video-1258344699.cos.ap-guangzhou.myqcloud.com/lib/dash.
```

3. プレーヤーを初期化し、オブジェクトアドレスを渡します。



```
<!-- MP4 -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"
  type="video/mp4"
/>

<!-- HLS -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"
  type="application/x-mpegURL"
/>
```

```
<!-- FLV -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.flv"
  type="video/x-flv"
/>

<!-- DASH -->
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mpd"
  type="application/dash+xml"
/>
```

サンプルコードを取得します。

MP4再生サンプルコード

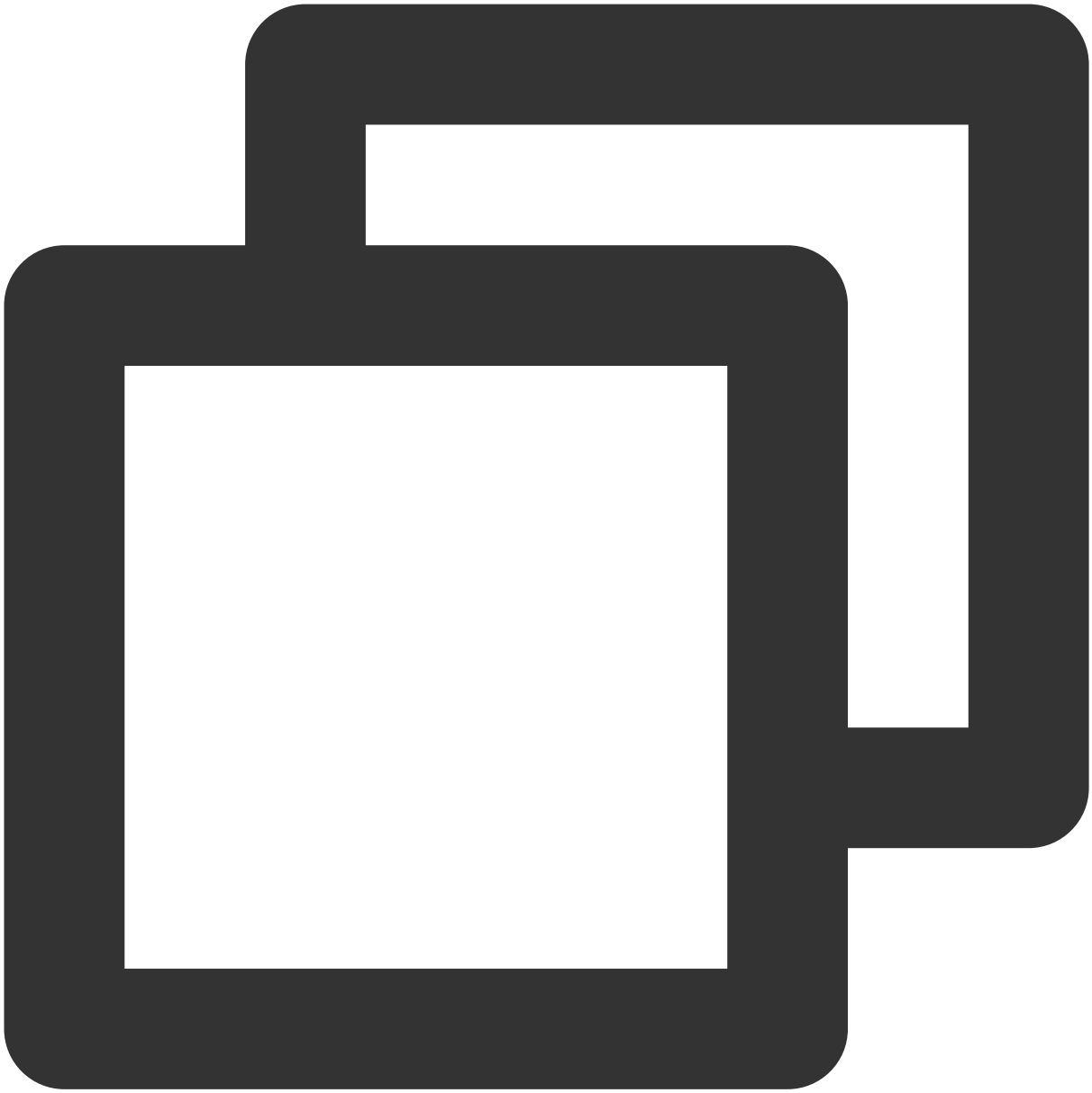
FLV再生サンプルコード

HLS再生サンプルコード

DASH再生サンプルコード

PM3U8ビデオを再生する

PM3U8はプライベートM3U8ビデオファイルのことです。COSはプライベートM3U8 TSリソースの取得に用いるダウンロード権限承認APIを提供しています。プライベートM3U8インターフェースをご参照ください。



```
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8?ci-process=pm3
  type="application/x-mpegURL"
/>
```

サンプルコードを取得します。

PM3U8再生サンプルコード

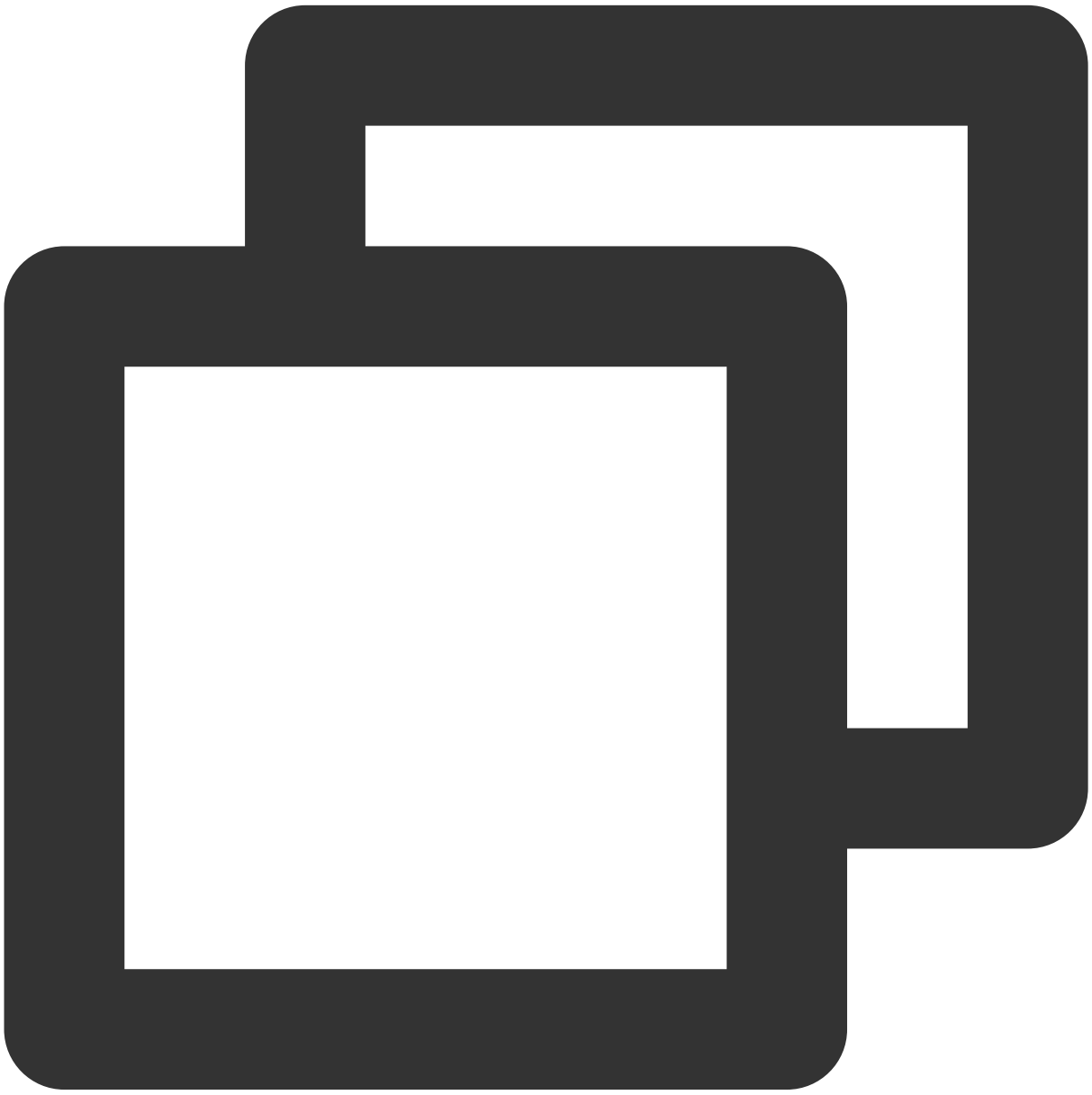
カバー画像を設定する

1. COSバケット上のカバー画像のオブジェクトアドレスを取得します。

注意：

Cloud Infiniteインテリジェントカバー機能により、最適なフレームを抽出してスクリーンキャプチャを生成し、それをカバーにすることでコンテンツの魅力をアップさせます。

2. プレーヤーを初期化し、カバー画像を設定します。



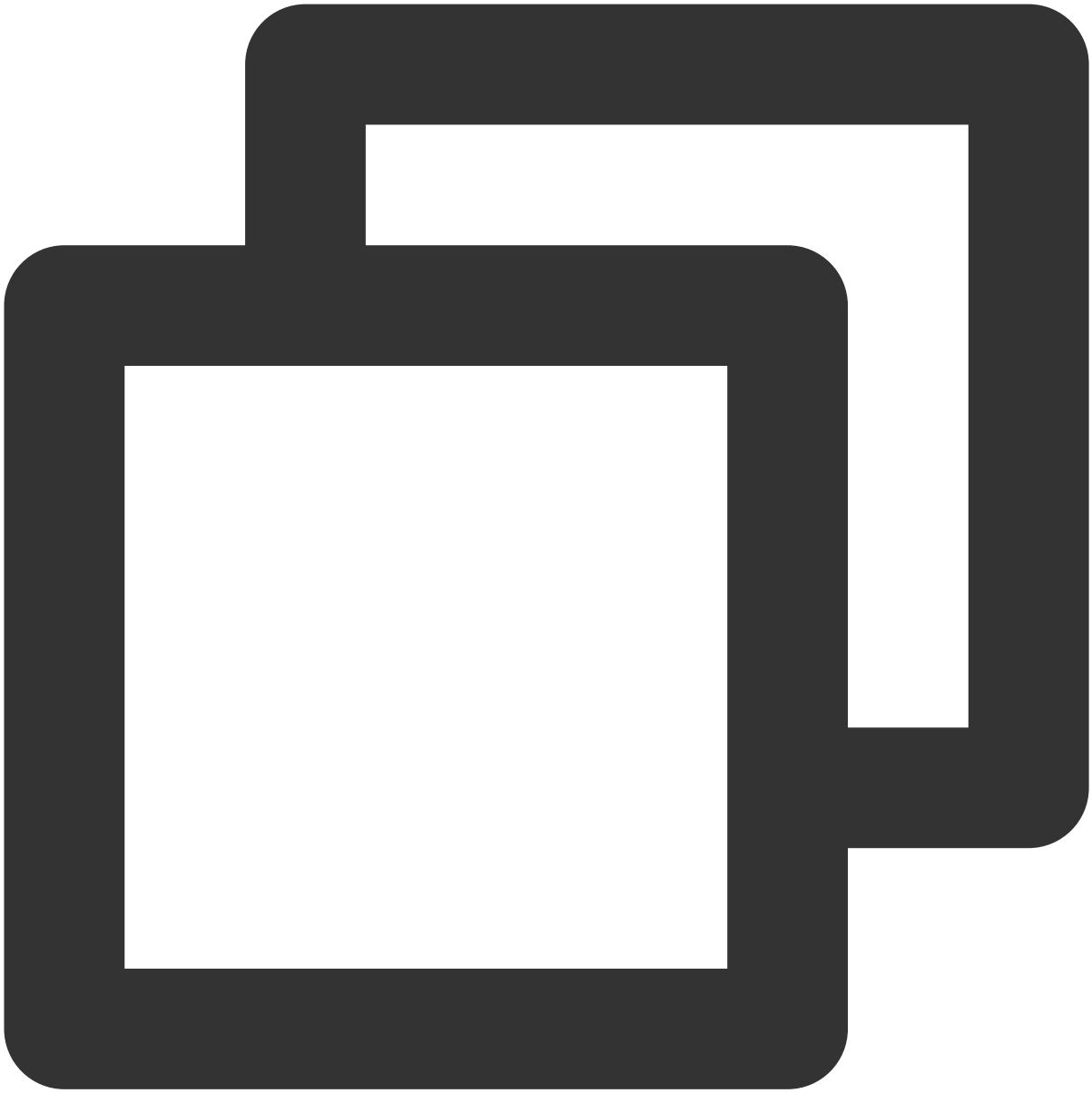
```
<video
  id="my-video"
  class="video-js"
  controls
```

```
    preload="auto"
    width="100%"
    height="100%"
    data-setup="{}"
    poster="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/poster.png"
  >
    <source
      src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.mp4"
      type="video/mp4"
    />
  </video>
```

サンプルコードを取得します。
カバー画像設定サンプルコード

HLS暗号化ビデオを再生する

ビデオコンテンツの安全性を保障し、ビデオの違法なダウンロードと拡散を防止するため、Cloud InfiniteはHLSビデオコンテンツに対する暗号化機能を提供します。この機能はプライベート読み取りファイルよりさらに高いセキュリティレベルを有します。暗号化されたビデオは、アクセス権限のないユーザーの視聴用に配信できなくなります。ビデオがローカルにダウンロードされた場合でも、ビデオ自体が暗号化されているため、悪意ある二次配信が不可能であり、ビデオの著作権が違法に侵害されないよう保障することができます。
操作手順は次のとおりです。
1. HLS暗号化ビデオの再生およびCOSオーディオビデオ実践 | ビデオをロックするのフローを参照し、暗号化ビデオを生成します。
2. プレーヤーを初期化し、ビデオオブジェクトアドレスを渡します。



```
<source
  src="https://<BucketName-APPID>.cos.<Region>.myqcloud.com/xxx.m3u8"
  type="application/x-mpegURL"
/>
```

サンプルコードを取得します。

HLS暗号化ビデオ再生サンプルコード

Data Security
盗用防止ガイド

最終更新日：：2024-07-17 16:23:42

前書き

近年、ますます多くのユーザーがウェブサイトや画像ホスティングを構築する際に、画像やビデオなどのリソースを Cloud Object Storage（COS）にアップロードしております。これによりアクセスの安定性が向上し、サーバーのストレージスペースの圧力を軽減していますが、それに伴うトラフィック盗用や画像リンク不正アクセスの問題も多くの開発者を悩ませており、一度ストレージスペースに悪意にアクセスされると、高いトラフィック料金が発生し、不必要な紛争が発生することになります。このような問題は、実際には様々な手段で防御することができます。この文章では、いくつかの一般的な防御手段を紹介し、開発者が合理的にバケットを設定し、セキュリティメカニズムを確立し、同様の問題による多額の資金損失のリスクを減らすことに役立ちます。

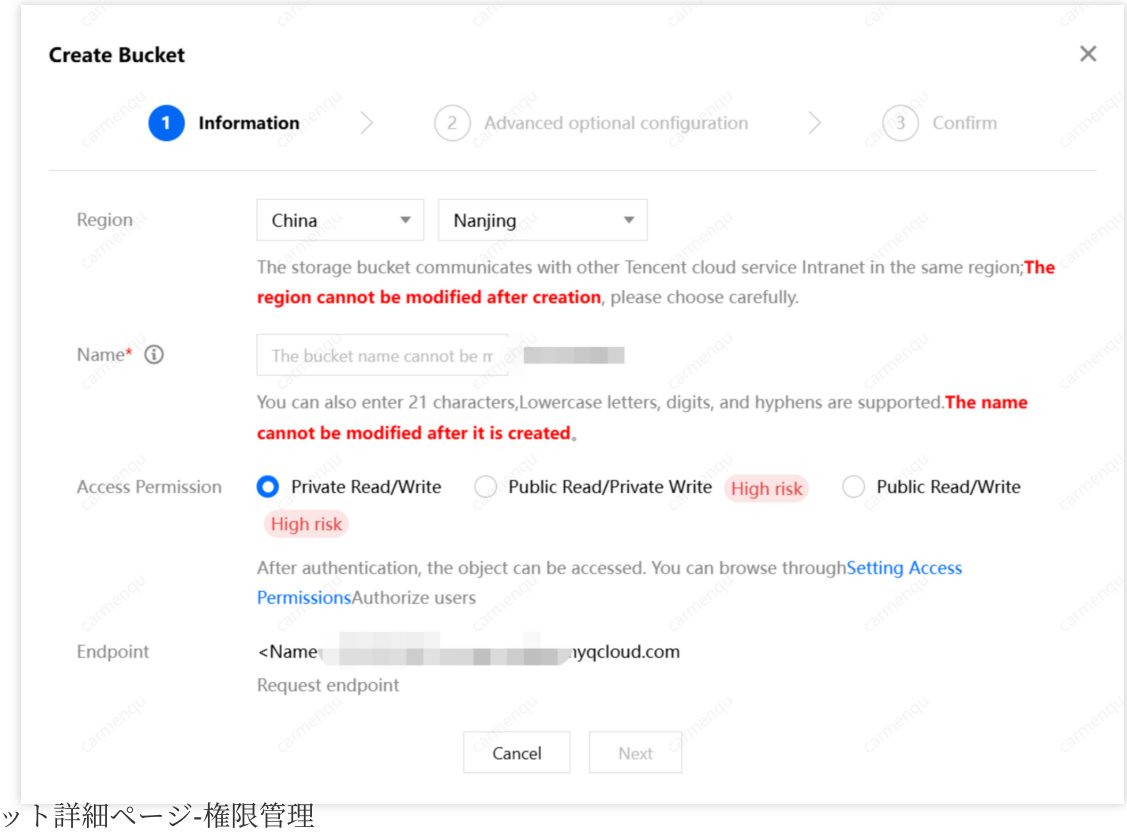
防御ソリューション

盗用に対する防御方法は数多くあります。ここでは、設定の難易度さやハードルによって区別され、開発者は実際の状況やニーズに応じて選択することができます。

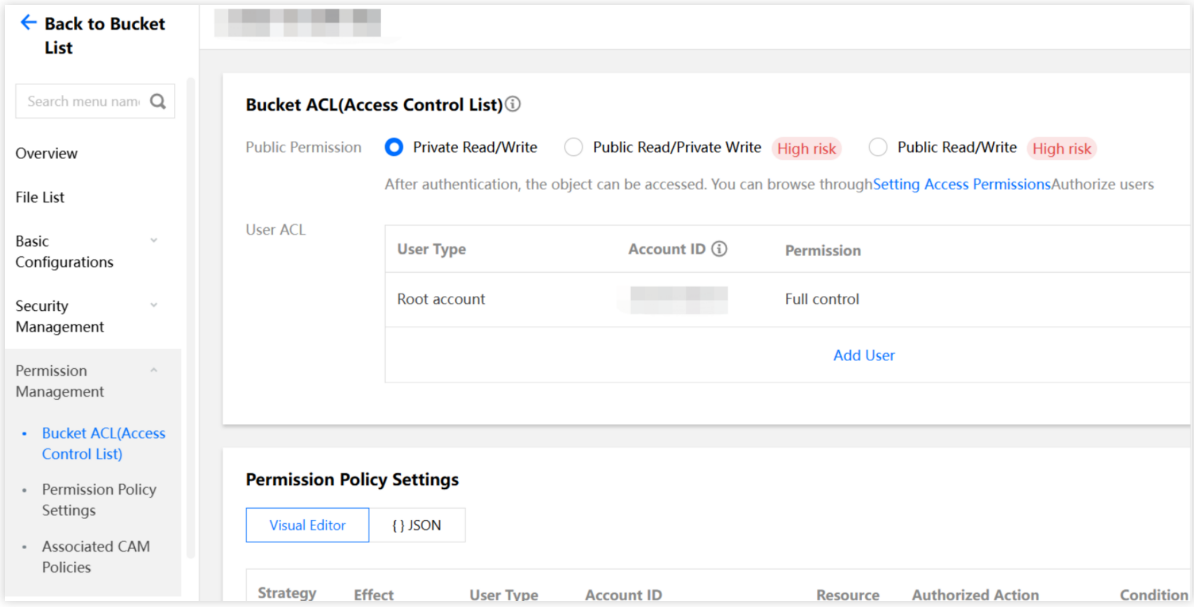
基本防御ソリューション

バケットアクセス権の変更

アクセス権は、バケットの最も核心的で機密性の高い設定の一つです。不完全な統計によると、盗用理由のほとんどは、ユーザーが**パブリックリード**権限を設定するためです。パブリックリード権限は署名なしで匿名でアクセスできるため、ブラックマーケットや攻撃者に悪用の機会を与えています。バケットをプライベートリード・ライトに変更することで、盗用リスクを大幅に減らすことができます。キーを取得できなければ、署名をコンピューティングできなく、アクセスは拒否されます。
特別なビジネスニーズがない場合、可能な限り**プライベートリード・ライト**権限を設定することを推奨します。
COS コンソールには、バケットパー権限を設定できる場所が 2 ヶ所あります。
バケット作成ポップアップ ウィンドウ



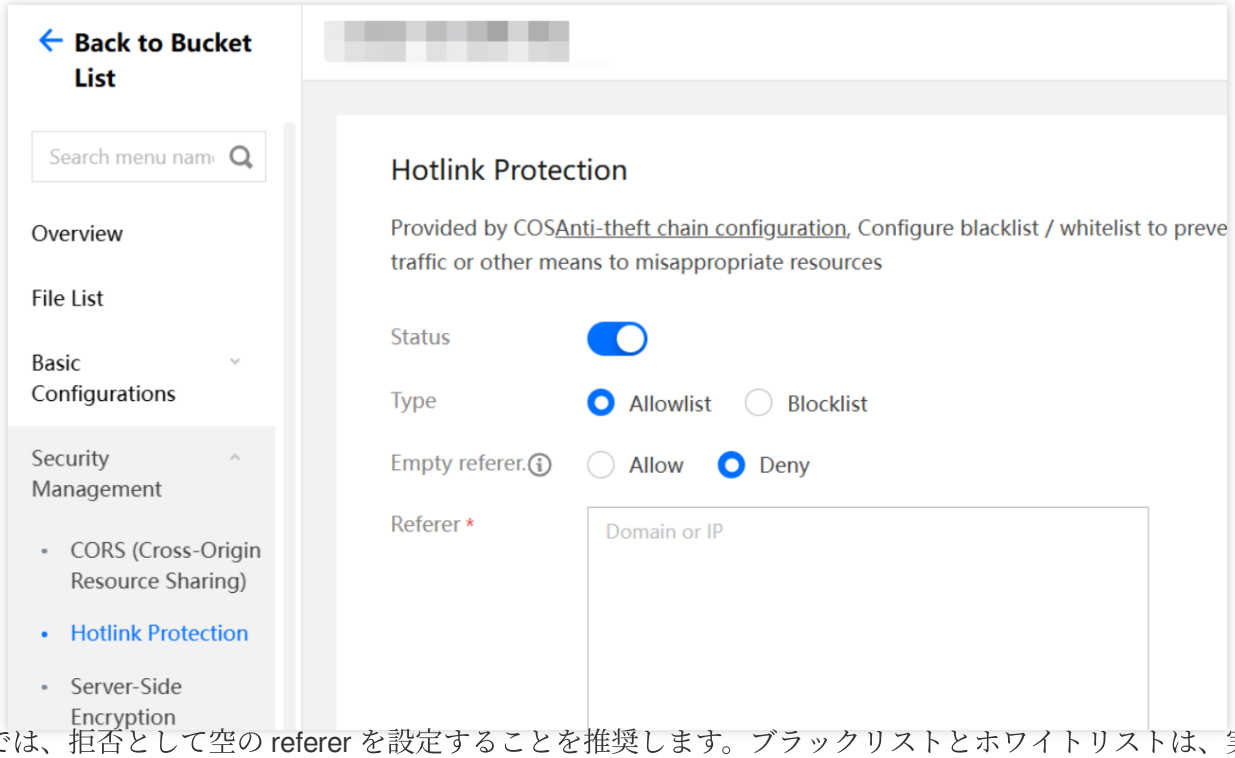
バケット詳細ページ-権限管理



バケット防犯リンクの有効化

防犯リンクは最も一般的な防御手段の一つで、原理は HTTP Referer ヘッダを介して判断・検証を行い、ユーザーはホワイトリストまたはブラックリストを設定すること、特定のアクセス元を許可または拒否することができま

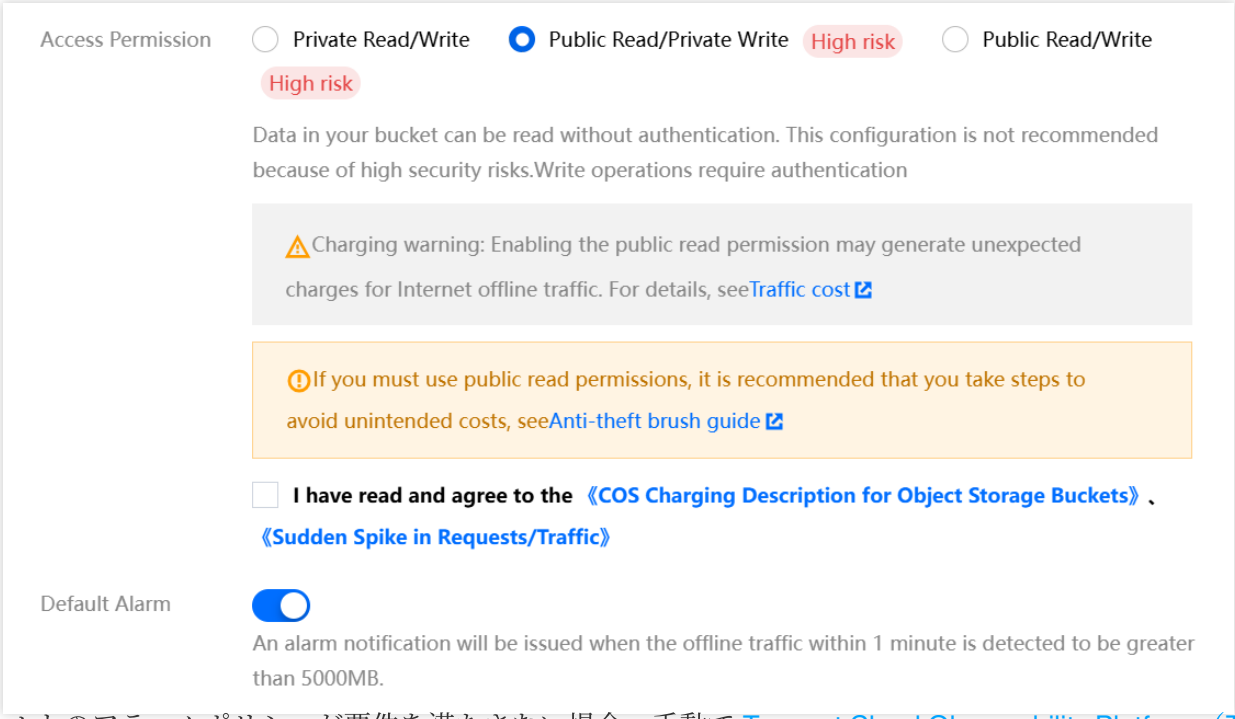
す。
COS コンソール-バケット詳細ページ-セキュリティ管理で防犯リンクの設定 を見つけることができます。



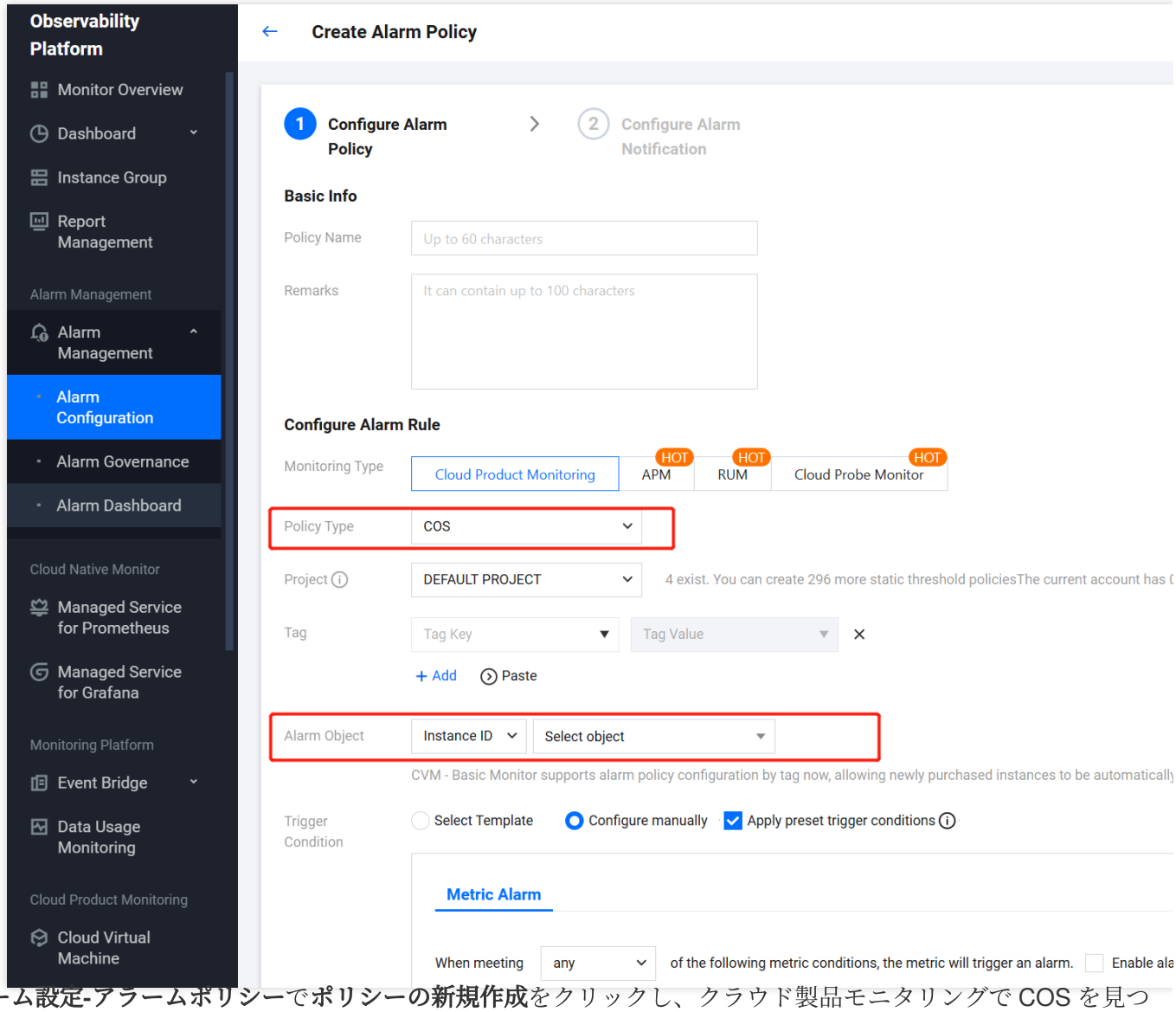
ここでは、拒否として空の referer を設定することを推奨します。ブラックリストとホワイトリストは、実際のビジネス状況に応じて選択することができます。特定のドメイン名の下に固定してアクセスする場合は、ホワイトリストを設定することができます。悪意のあるアクセスがあることを判明し、アクセスするドメイン名または IP を明確に知っている場合、手動でブラックリストを設定してブロックすることができます。防犯リンクを使用する際のテクニックについては、 防犯リンクの実践を参照してください。

Cloud Monitor アラームの設定

ログ管理を設定することで、盗用の来源を特定し、分析することに役立ち、クエリするための豊富なフィールドを提供することができますが、デメリットは、開発者が主動的にログをフォロー必要があることです。盗用問題をタイムリーに検出するために、Cloud Monitor アラームを設定することができます。現在、COS コンソールはバケットを作成する時にアラームを設定することをサポートしています。



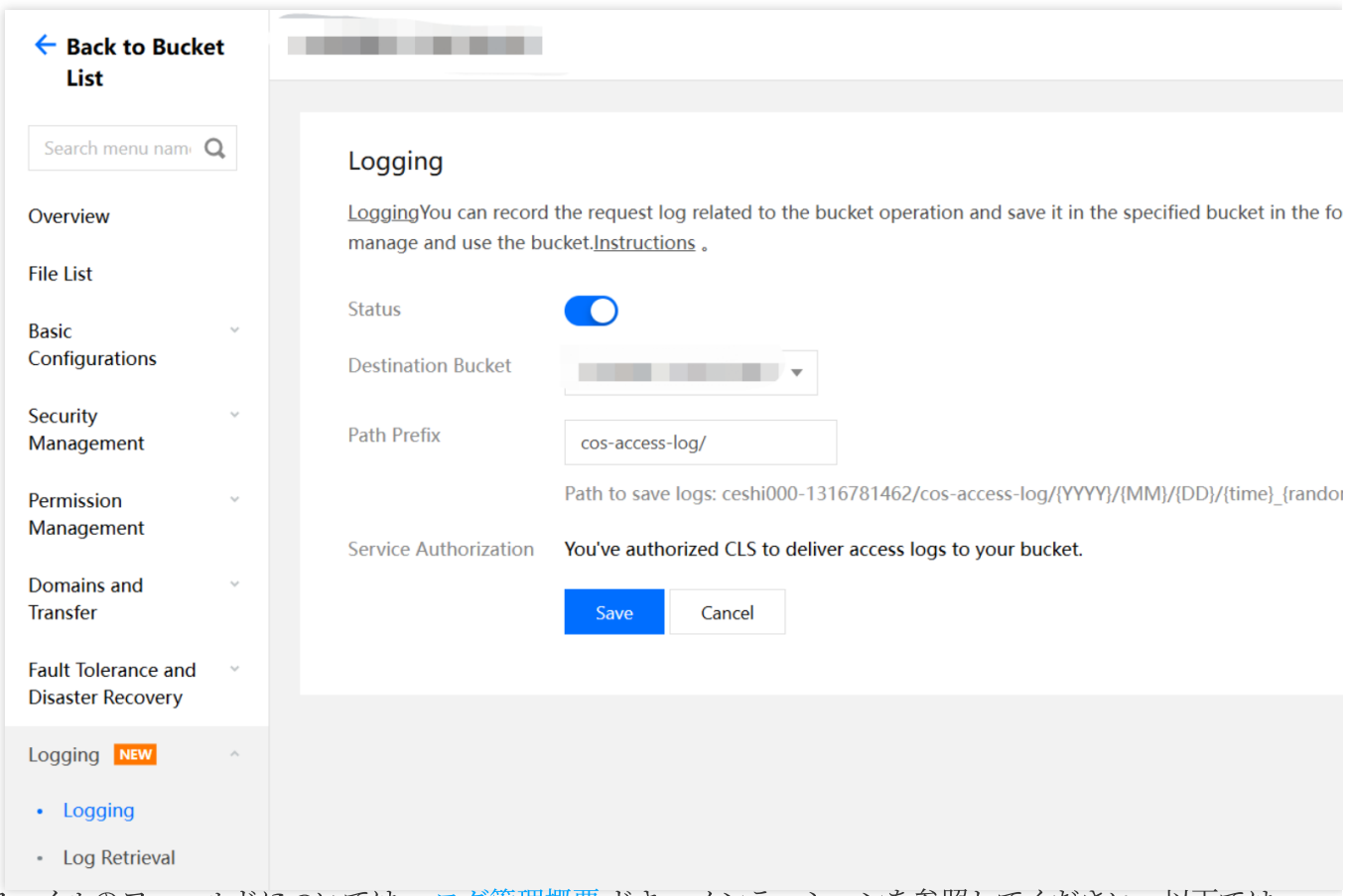
デフォルトのアラームポリシーが要件を満たさない場合、手動で Tencent Cloud Observability Platform（TCOP）にアクセスしてカスタムアラームポリシーを作成することもできます。



アラーム設定-アラームポリシーで**ポリシーの新規作成**をクリックし、クラウド製品モニタリングで COS を見つけ、アラームオブジェクトのインスタンス ID で設定が必要なバケットを見つけると、トリガー条件を自分で指定できます。COS バケットのデータモニタリング画面に表示されるすべての指標は、ここで設定することができます。

ログ管理の有効化

上記の防犯リンクのブラックリストでは、「ドメイン名や IP」に悪意のあるアクセスを発見し場合」について言及しました。ここでは、バケットのログ管理を有効にする必要があります。アクセスログは、各リクエストの様々なフィールドを記録し、アクセス元を素早く見つけることに役立ちます。
COS コンソール-バケット詳細ページ-ログ管理でログストレージを見つけることができます。 有効にすると、バケットの指定されたパスプレフィックスの下にアクセスログを見つけることができます。

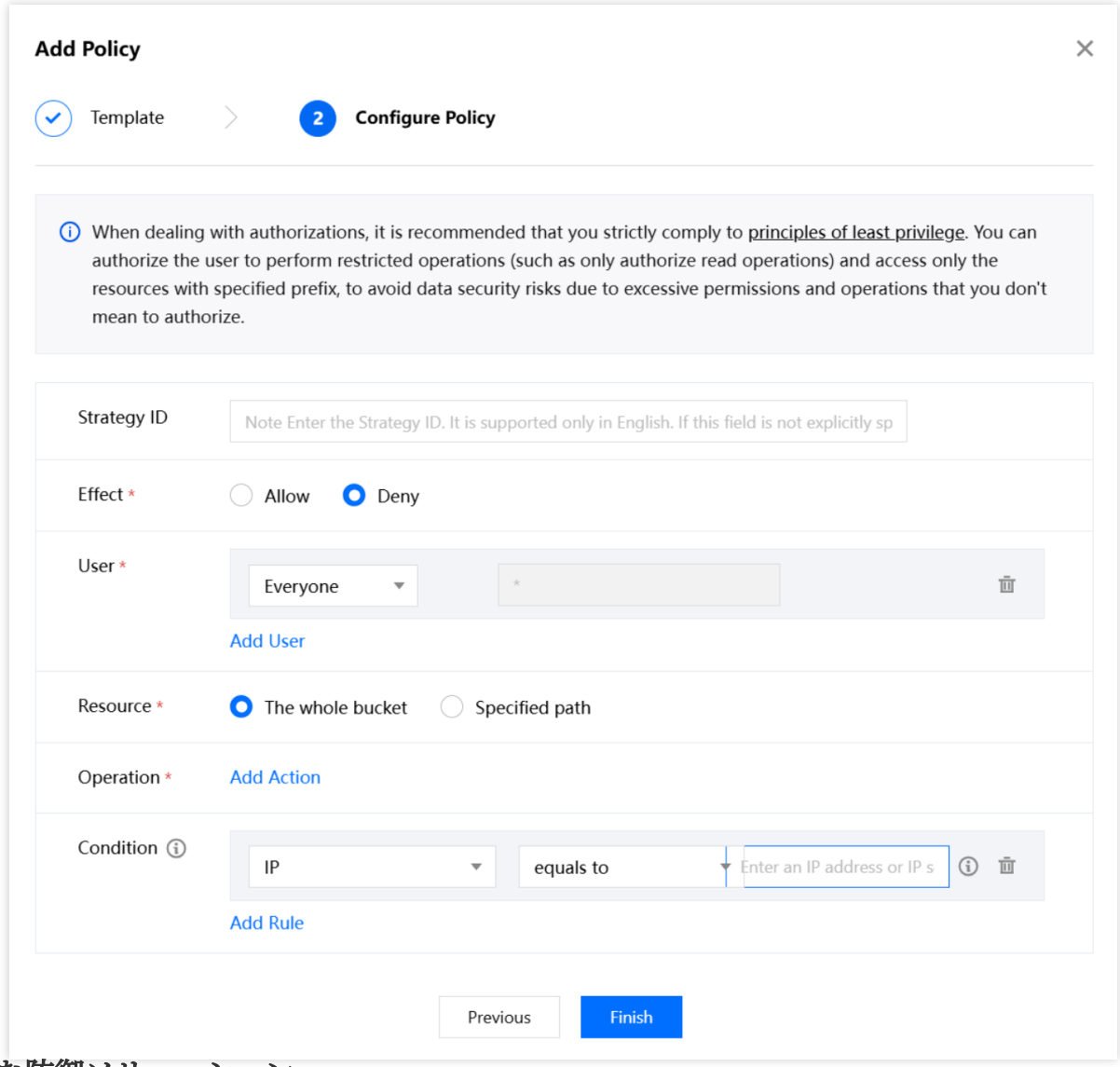


ログファイルのフィールドについては、 ログ管理概要 ドキュメンテーションを参照してください。以下では、一般的に使用されるフィールドについて説明します。

userSecretKeyId：リクエストがどの鍵 KeyId を介してアクセスされたかを判断することができます。リクエストがビジネス自身によって開始されたものでないことが判明された場合、キーが漏洩している可能性が高いので、Tencent Cloud CAM コンソールで安全でないキーを無効にすることができます。

referer：つまり、防犯リンクの設定で判断するために使用される条件です。不明な referer を発見した場合は、他のサイトによってリンク不正アクセスをされた可能性があり、防犯リンク-ブラックリストを設定してその referer のアクセスを制限することができます。詳細については、1.2 バケット防犯リンクの有効化を参照してください。

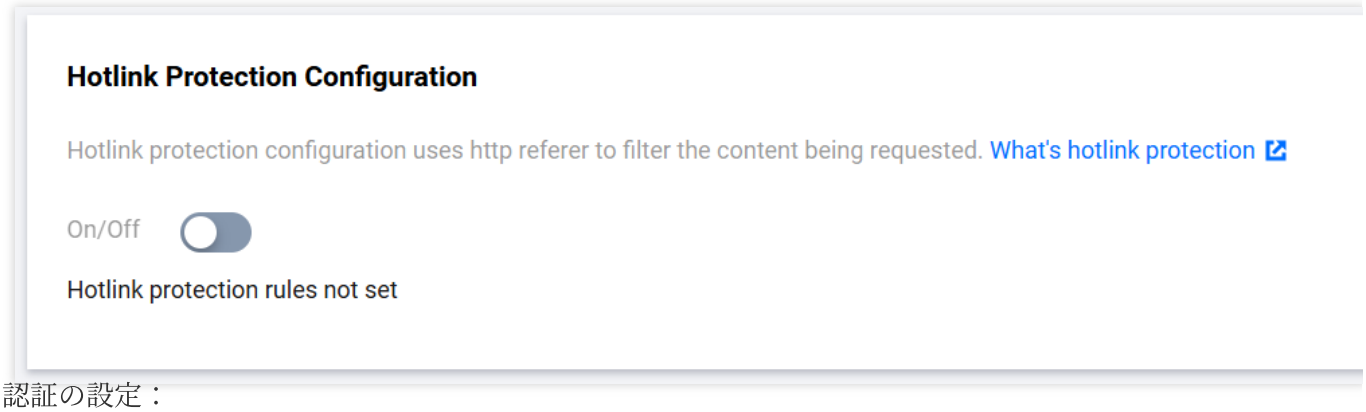
remoteIp：アクセス元 IP を特定することができます。信頼できない IP であることが判明された場合は、以下の例のように、**バケット詳細ページ-権限管理 - Policy 権限設定に進み、ポリシーの追加をクリックして、**その IP がバケットにアクセスすることを禁止する Policy を設定することができます。



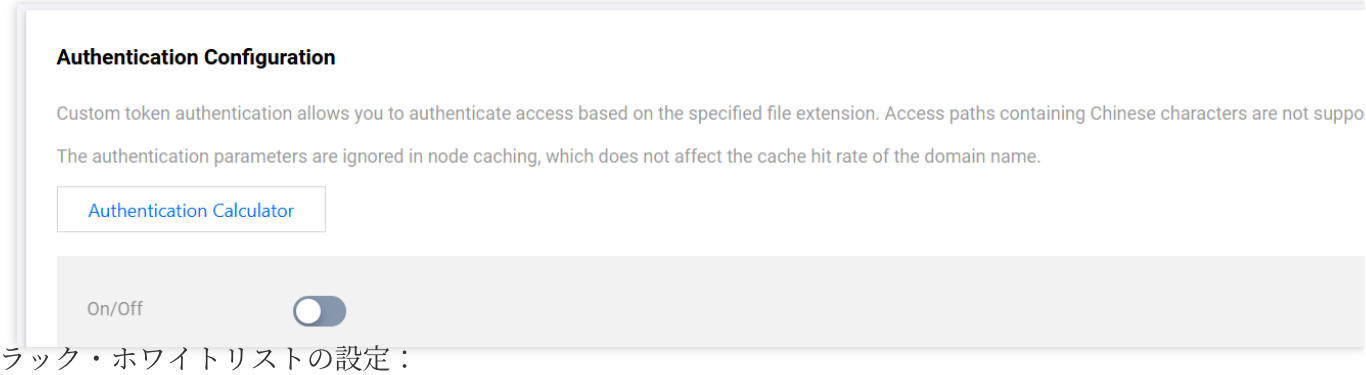
高度な防御ソリューション

カスタム CDN でドメイン名を加速

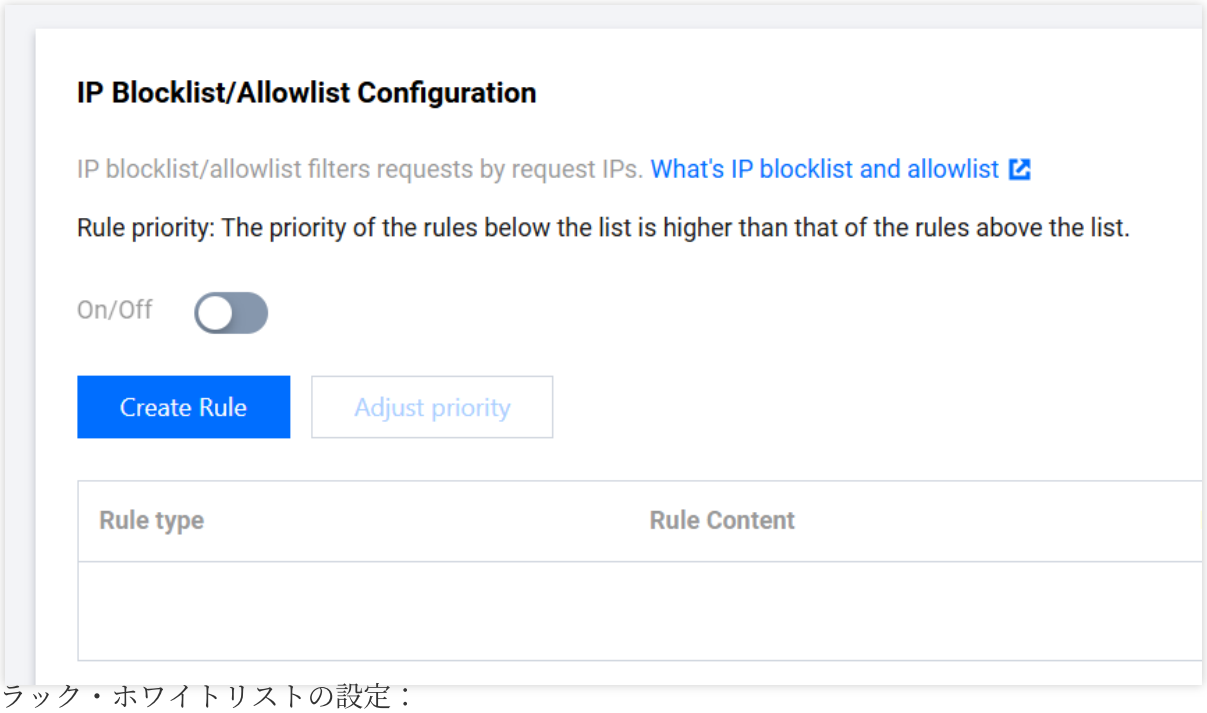
Tencent Cloud CDN はまた、盗用を防止するための多くの設定を提供しており、盗用を効果的に防御できます。ビジネスにカスタム CDN ドメイン名を使用している場合、 Tencent Cloud CDN コンソール**-ドメイン名詳細-アクセス管理ページ**で設定することができます。よく使われる設定は以下のとおりです。
防犯リンクの設定：



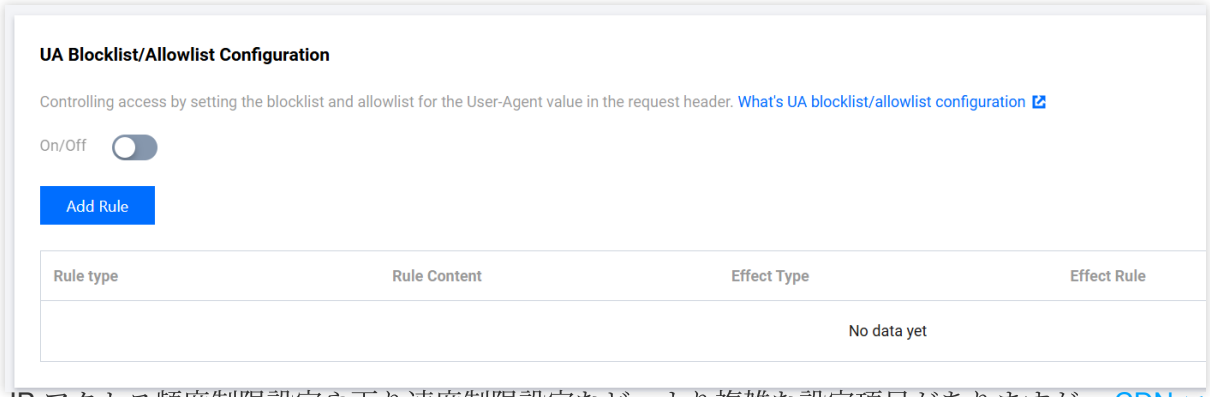
CDN 認証の設定：



IP ブラック・ホワイトリストの設定：



UA ブラック・ホワイトリストの設定：



また、IP アクセス頻度制限設定や下り速度制限設定など、より複雑な設定項目がありますが、 CDN コンソールに関するドキュメンテーション を参照して使用してください。

SCF+Cloud Monitor+COS API による自動ブロックの実現

基本的な防御では、Cloud Monitor アラームを設定することでトラフィックの異常をタイムリーに検出できると述べましたが、情報が漏洩した場合や、外出でタイムリーに対応できない場合があります。ここでは、自動化スクリプトコードで API を呼び出し、簡単な自動化ロジックを実現することができます。
Cloud Monitor から下りトラフィックメ指標(InternetTraffic)の異常を取得してから、 COS PutBucketAclを自動的に呼び出してバケットの権限をプライベートリード・ライトに変更します。主に以下の 2 つの API インタフェースに関連しており、ここではデバッグの参考としてオンライン呼び出しの例を提供します。
Cloud Monitor GetMonitorData - 呼び出しの例
COS の PutBucketAcl - 呼び出しの例

関連する提案

上記は盗用問題に対する一般的な防護手段ですが、それ以外には COS を日常的に使用する上で細かいところに注意を払う必要があります。ここでは、盗用リスクをある程度減らすための追加の使用アドバイスをいくつか提供します。
外部公開のオープンソースコードで平文の API アクセスキーを使用することを避け、キーの漏洩によるセキュリティリスクを避け、**最小権限の原則**を参考に使用することを推奨します。
クロスドメインルールを設定する時には、すべてのソース（つまり Origin: *）からのアクセスを許可することを避け、可能な限り明確なソースを設定するようにしてください。
大量のファイルをアップロードする場合、簡単すぎる順序プレフィックス（例えば、数字、タイムスタンプなど)の使用は避けてください。これにより、攻撃者がバケット下のファイルをトラバーサルしやすくなる可能性があります。

ホットリンク保護実践

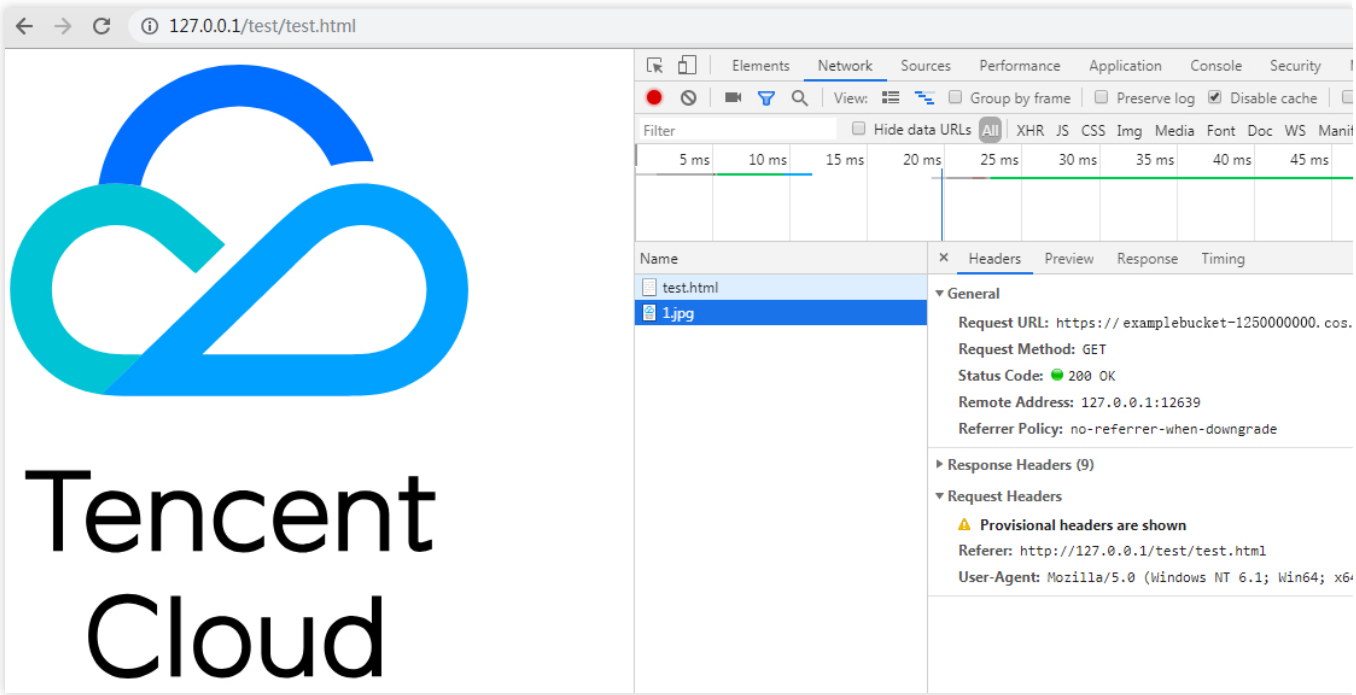
最終更新日：：2024-06-26 10:47:37

概述

COSはホットリンク保護を配置することができます。この機能はアクセス元のブラックリストとホワイトリストを設定することができて、資源の熱リンクを避けることができます。本文はどのように貯蔵桶に熱リンク保護を配置するかを紹介します。

ホットリンク保護の仕組み

ホットリンク保護は，リクエストヘッダ中の引用アドレスを検査することで動作する：
Refererはヘッダの一部であり，ブラウザがWebサーバに要求を送信するとき，通常はどのページからの要求をサーバに要求するかをサーバに伝えるRefererを携帯し，サーバは資源へのアクセスを拒否するか許可するかを決定する.
ファイルリンクを開くと `https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/1.jpg` 直接ブラウザで、要求ヘッダは引用しない。
例えば、次の図では、画像が `1.jpg` 組み込まれています `https://127.0.0.1/test/test.html` あなたが訪問する時、訪問の原点への引用を持っていきます。 `https://127.0.0.1/test/test.html` ：

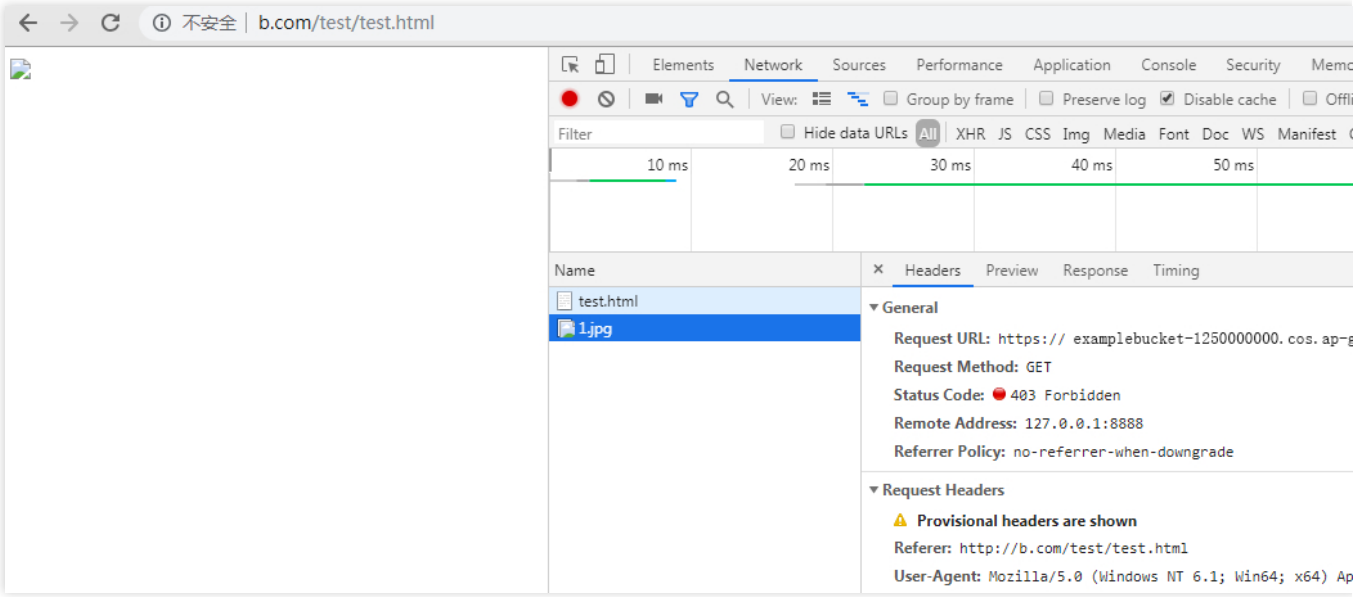


ホットリンク保護ケース研究

ユーザAは、画像リソースをアップロードする `1.jpg` COSに接続されていますが、画像のアクセス可能リンクは `https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/1.jpg` .
ユーザAは、そのウェブページに画像を埋め込む `https://example.com/index.html` 画像はアクセス可能。
ユーザBは，ユーザAのページ上で画像を見て，自分のページに埋め込むことにした. `https://b.com/test/test.html` また，ユーザBのWebページも画像を正確に表示することができる.
上記の場合、ユーザAの画像リソース `1.jpg` ユーザAは、COS内のリソースがユーザBのウェブページによって使用されていることを知らず、追加の通信料によって損失を受ける。

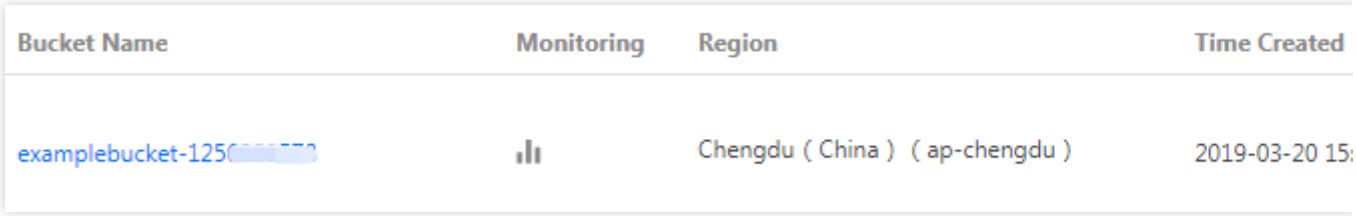
溶液

上だホットリンク保護ケース研究ユーザAは、以下のように熱リンク保護を設定することにより、ユーザBがその画像を熱リンクすることを阻止することができる：
1. ボックス「examplebucket-125000000」のためにホットリンク保護ルールを設定します。2つの方法でユーザBがホットリンクを行うことを防ぐことができます：
方法1：ブラックリストを配置し、入力する `*.b.com` ドメイン名にして保存する。
方法2：ホワイトリストモードを配置し、入力 `*.example.com` ドメイン名にして保存する。
2. ホットリンク保護を開始した後：
以下の場合には画像を正確に表示することができる： `https://example.com/index.html` アクセスできます。
画像を表示できない `https://b.com/test/test.html` 訪問されたのは以下の通りである.

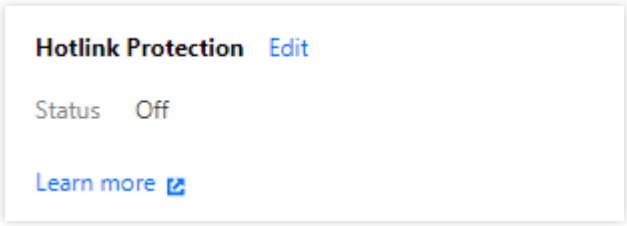


方向.

1. 登録するCOSコンソールクリックして**桶の時計**左側欄にバケツリストページを入力します。
2. ホットリンク保護のためのストレージを選択し，それを入力する.



3. クリックする.**基本配置**ホットリンクの保護設定を見つけてクリックして**へんしゅう者**編集状態に入るには、以下の操作を実行してください。



4. ホットリンクを有効にしてリストタイプとドメイン名を保護し、構成する。ここで、イネーブル方法2を選択すると、以下のようになる：

タイプ：ブラックリストとホワイトリストの2種類があります。

ブラックリスト：リスト内のドメイン名がストレージバレルのデフォルトアクセスアドレスにアクセスすることを禁止します。**リストに**リポジトリのデフォルトアクセスアドレスにアクセスした場合、403エラーが返される。

ホワイトリスト：リスト内のドメイン名ではないストレージ·バケツのデフォルト·アクセス·アドレスにアクセスすることを禁止します。**リストにはない**リポジトリのデフォルトアクセスアドレスにアクセスした場合、403エラーが返される。

推薦人：最大10ドメインを設定し、接頭辞でマッチングすることができます。ドメイン、IP、ワイルドカード `*` サポートするフォーマット(行ごとに1つのアドレス)。以下に構成規則の説明と例を示す：

IPおよびポートを有するドメインをサポートする、例えば `example.com:8080` 和 `10.10.10.10:8080` .

もし `example.com` 構成され、アドレスプレフィックスは `example.com` 例えば、当たってもいいです。 `example.com/123` 和 `example.com.cn` .

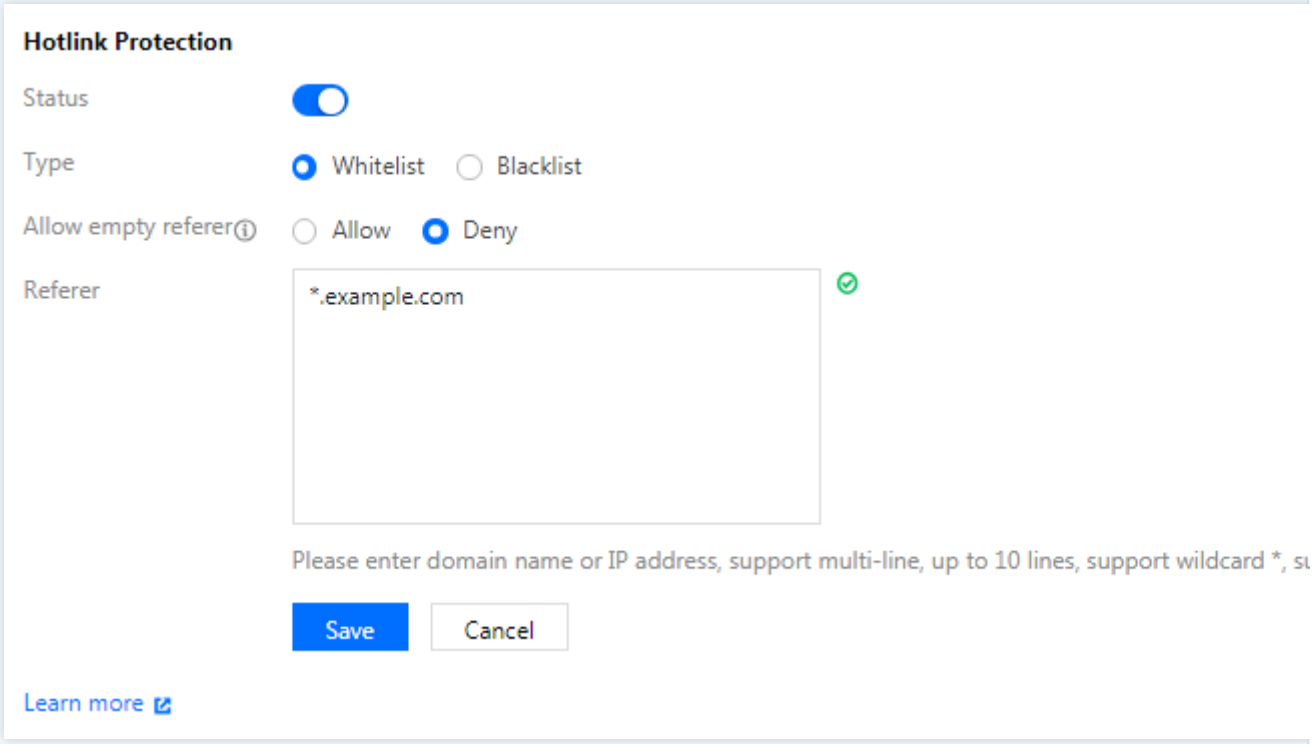
もし `example.com` 構成され、アドレスプレフィックスは `https://example.com` 和 `http://example.com` 当たってもいい。

もし `example.com` このようなポートドメイン名を特定することができます `example.com:8080` .

もし `example.com:8080` 構成されています `example.com` 当たってはいけない。

もし `*.example.com` この場合、第2レベルおよび第3レベルのドメイン名を制限することができ、例えば `example.com``b.example.com` そして `a.b.example.com` .

ホットリンク保護後すでに起用されている.そのためには，対応するドメイン名を入力しなければならない.
5.構成が完了したら、クリック保存する.



ありふれた問題

ホットリンク保護の問題については、ご参照くださいデータセキュリティCOSでよくある問題の解答中の部分。

データ検証
MD5検証

最終更新日：：2024-06-26 10:47:37

概要

データをクライアントとサーバー間で伝送する際にエラーが発生することがあります。COSはMD5検証の方式でアップロードしたデータの完全性を保証することができます。COSサーバーが受信したデータのMD5チェックサムとユーザーが設定したMD5チェックサムが一致した場合のみ、データのアップロードが成功します。
COS内の各オブジェクトには1つのETagが対応しています。ETagはオブジェクトが作成された時点でのオブジェクト内容の情報タグですが、ETagはオブジェクト内容のMD5チェックサムと必ずしも同じではありません。このため、ETagによってダウンロードしたオブジェクトと元のオブジェクトが同じかどうかを検証することはできませんが、 ユーザーはカスタムオブジェクトメタデータ（x-cos-meta-*）を使用することで、ダウンロードしたオブジェクトと元のオブジェクトの整合性検証を実現することができます。

データチェック検証方式

アップロードオブジェクトの検証
COSにアップロードしたオブジェクトとローカルのオブジェクトが一致しているかを検証したい場合、ユーザーはアップロードの際にHTTPリクエストのContent-MD5フィールドを、Base64エンコードを経たオブジェクト内容のMD5チェックサムに設定することができます。このときCOSサーバーはユーザーがアップロードしたオブジェクトを検証し、COSサーバーが受信したオブジェクトのMD5チェックサムとユーザーが設定したContent-MD5が一致した場合のみ、オブジェクトを正常にアップロードします。
ダウンロードオブジェクトの検証
ダウンロードしたオブジェクトと元のオブジェクトが一致しているかを検証したい場合、ユーザーはオブジェクトをアップロードする際に検証アルゴリズムを使用してオブジェクトのチェックサムを計算し、カスタムメタデータによってオブジェクトのチェックサムを設定し、オブジェクトのダウンロード後にオブジェクトのチェックサムを再計算し、カスタムメタデータとの比較によって検証することができます。この方式では、ユーザーはご自身で検証アルゴリズムを選択できますが、同一のオブジェクトについては、アップロードとダウンロードの際に使用する検証アルゴリズムは一致させる必要があります。

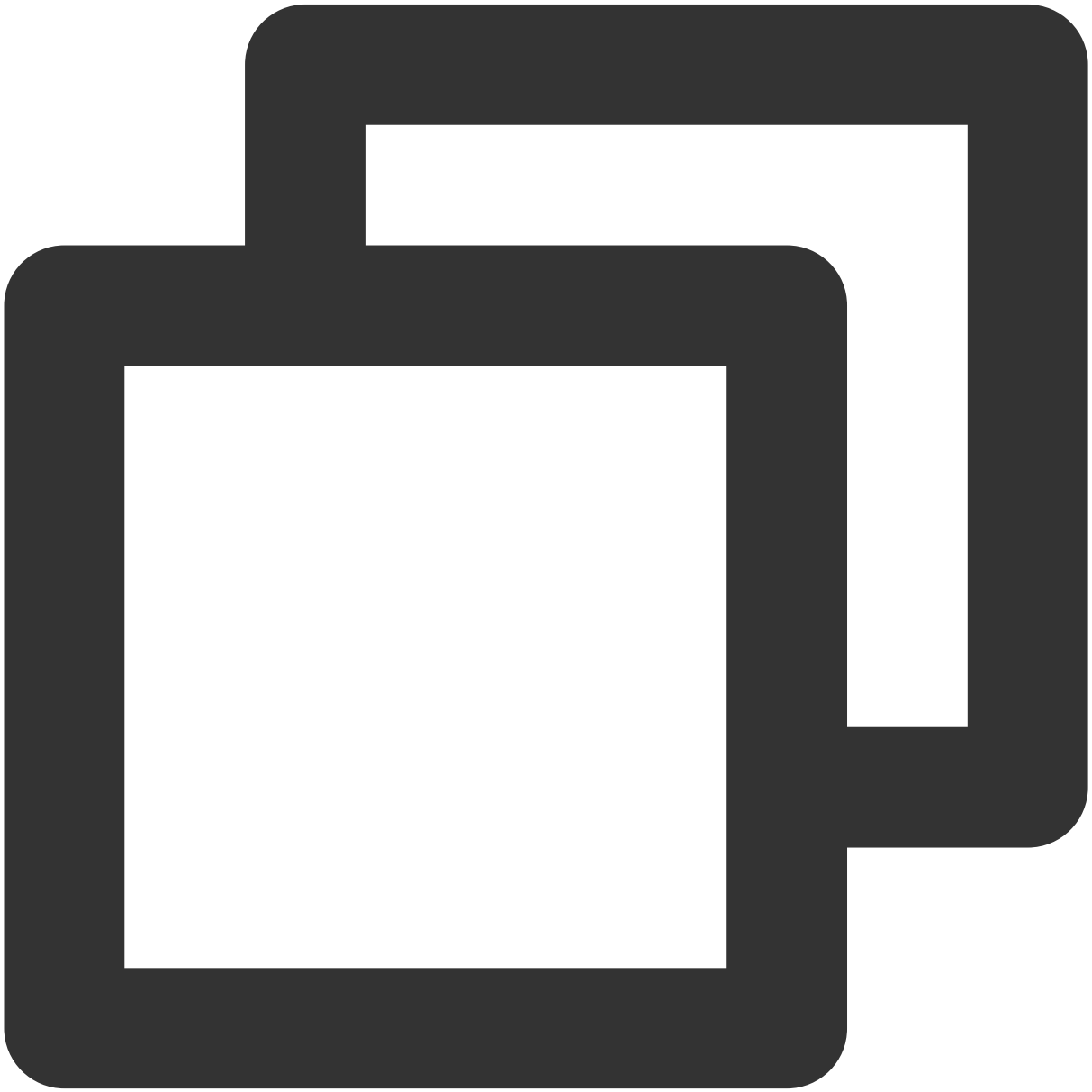
APIインターフェースの例

シンプルアップロードリクエスト

以下はユーザーがオブジェクトをアップロードする場合のリクエストの例です。オブジェクトをアップロードする際、Content-MD5を、Base64エンコードを経たオブジェクト内容のMD5チェックサムに設定します。こうすることで、COSサーバーが受信したオブジェクトのMD5チェックサムとユーザーが設定したContent-MD5が一致した場合のみ、オブジェクトのアップロードが正常に行われるようにし、かつカスタムメタデータx-cos-meta-md5をオブジェクトのチェックサムに設定します。

説明：

例で示したものはMD5検証アルゴリズムによって得られたオブジェクトのチェックサムです。ユーザーはご自身で他の検証アルゴリズムを選択することができます。

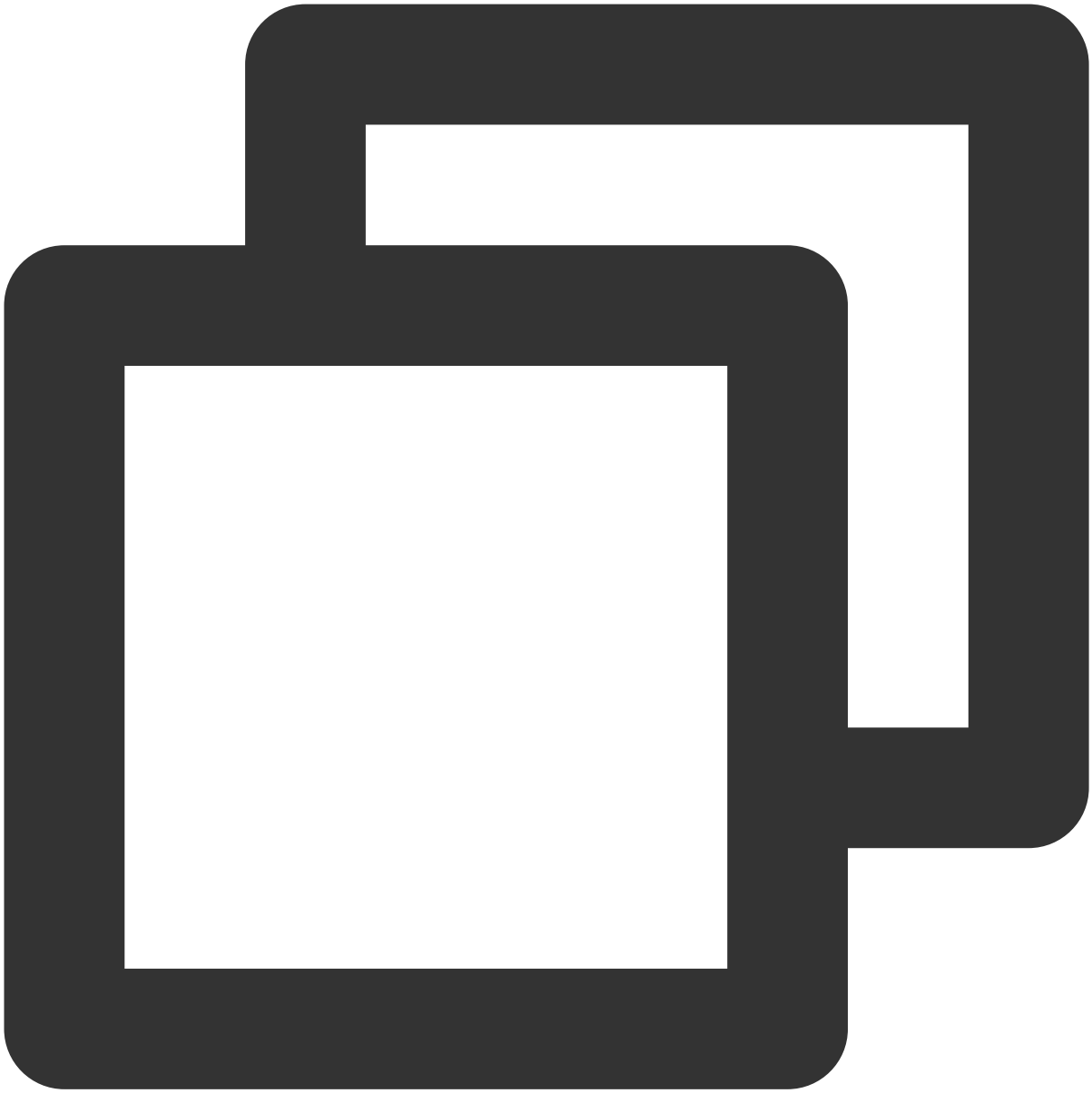


```
PUT /exampleobject HTTP/1.1
```

```
Host: examplebucket-1250000000.cos.ap-beijing.myqcloud.com
Date: Fri, 21 Jun 2019 09:24:28 GMT
Content-Type: image/jpeg
Content-Length: 13
Content-MD5: ti4QvKtVqIJAvZxDbP/c+Q==
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-si
x-cos-meta-md5: b62e10bcab55a88240bd9c436cffdcf9
Connection: close

[Object Content]
```

マルチパートアップロードリクエスト

以下はマルチパートアップロードの初期化リクエストの例です。オブジェクトのパートをアップロードする際、ユーザーはマルチパートアップロードを初期化することによってオブジェクトのカスタムメタデータを設定できます。ここではカスタムメタデータx-cos-meta-md5をオブジェクトのチェックサムに設定します。



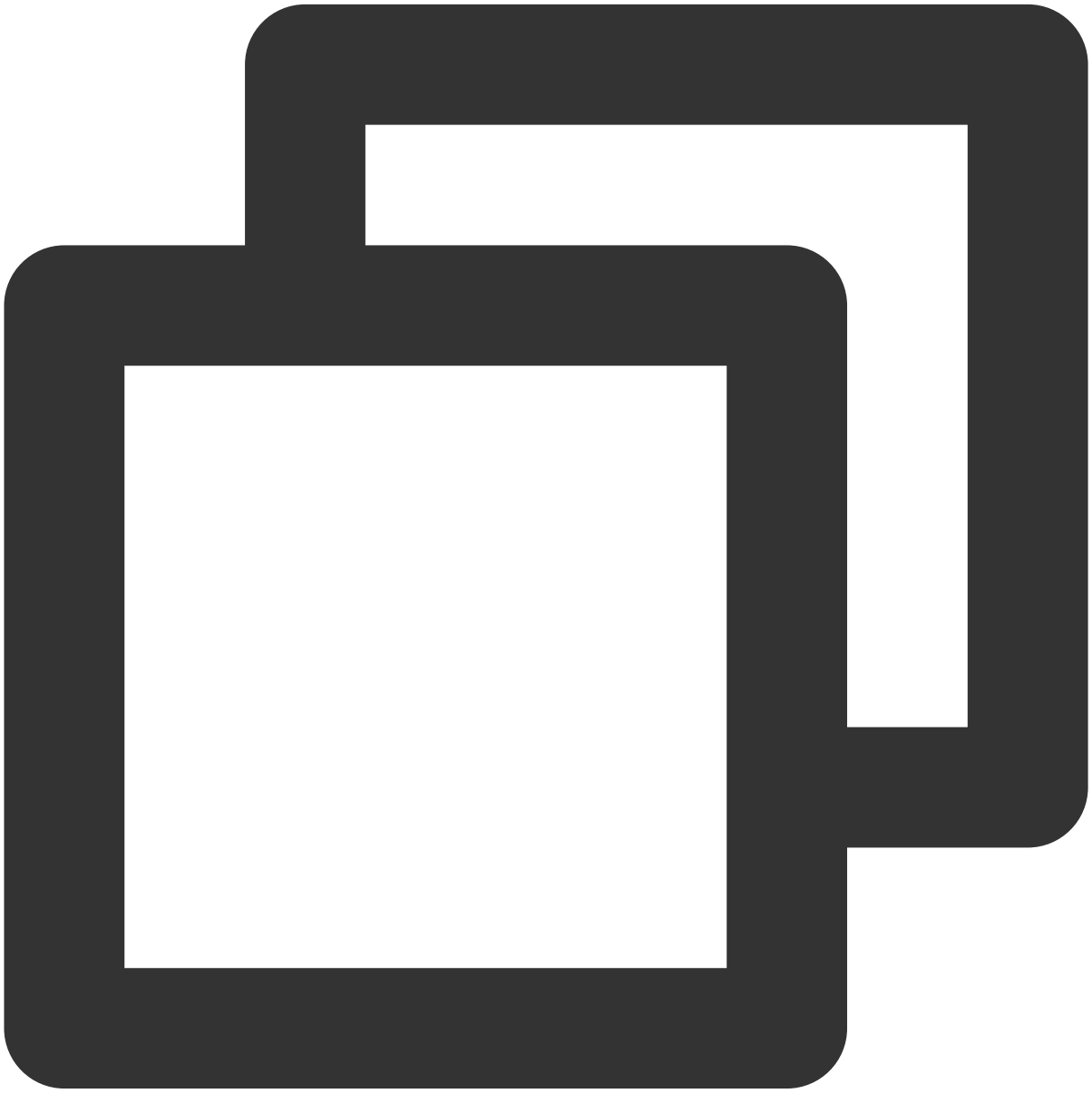
```
POST /exampleobject?uploads HTTP/1.1
Host: examplebucket-1250000000.cos.ap-beijing.myqcloud.com
Date: Fri, 21 Jun 2019 09:45:12 GMT
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-si
x-cos-meta-md5: b62e10bcab55a88240bd9c436cffdcf9
```

注意：

マルチパートアップロードのファイルについて、COSは各パートのMD5値のみを検証し、合体後の完全なファイルのMD5値の計算は行いません。

オブジェクトダウンロードの応答

以下はユーザーがオブジェクトダウンロードリクエストを送信後に取得する応答の例です。ユーザーは応答の中からオブジェクトのカスタムメタデータx-cos-meta-md5を取得することができ、オブジェクトのチェックサムを再計算してそのカスタムメタデータとの比較を行うことで、ダウンロードしたオブジェクトと元のオブジェクトが一致しているかを検証できます。



```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
Connection: close
```

```
Accept-Ranges: bytes
Cache-Control: max-age=86400
Content-Disposition: attachment; filename=example.jpg
Date: Thu, 04 Jul 2019 11:33:00 GMT
ETag: "b62e10bcab55a88240bd9c436cffdcf9"
Last-Modified: Thu, 04 Jul 2019 11:32:55 GMT
Server: tencent-cos
x-cos-request-id: NWQxZGUzZWNfNjI4NWQ2NF9lMWYyXzk1NjFj****
x-cos-meta-md5: b62e10bcab55a88240bd9c436cffdcf9

[Object Content]
```

SDKの例

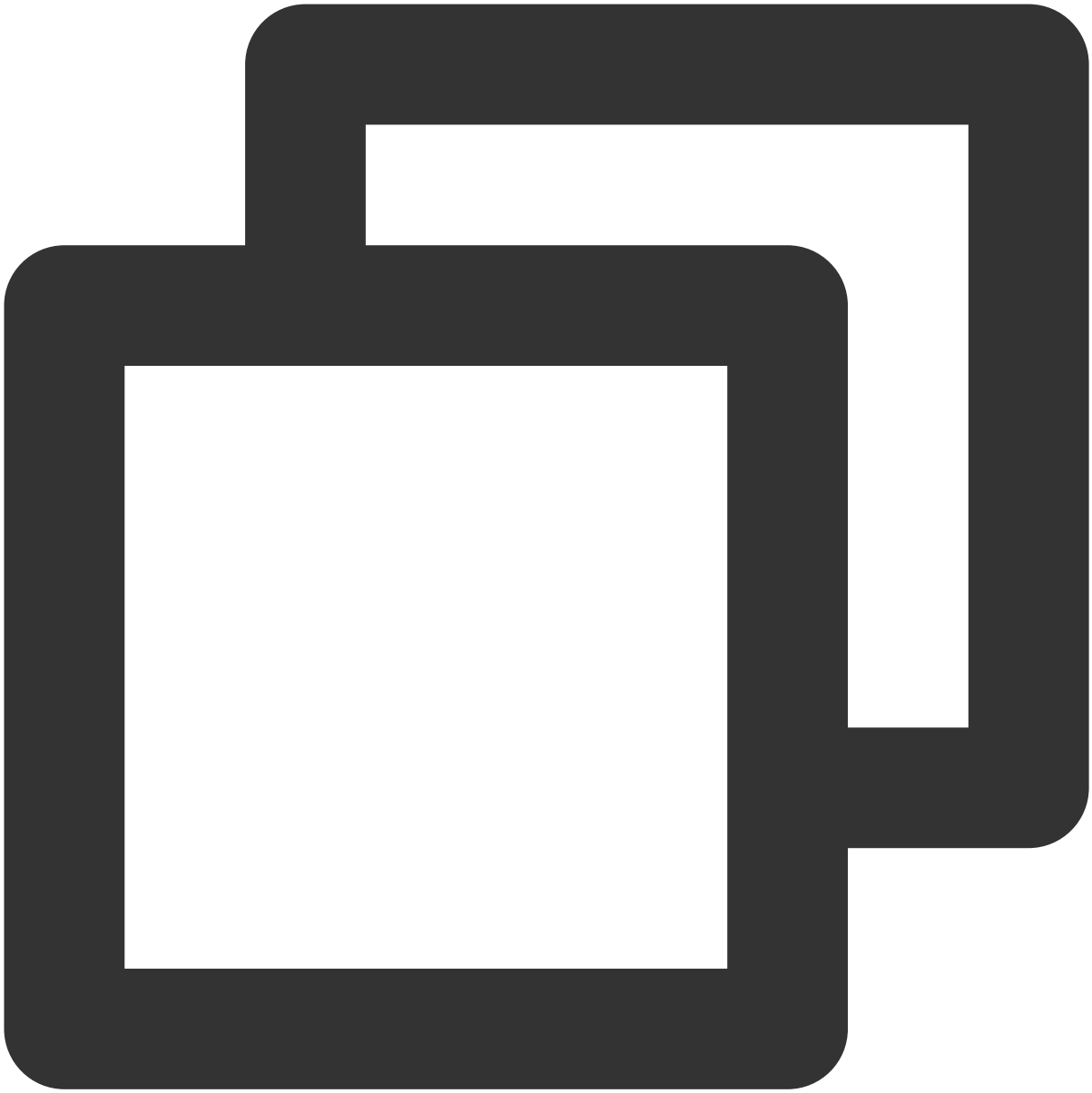
次に、Python SDKを例に、オブジェクトの検証方法についてご説明します。完全なサンプルコードは次のとおりです。

説明：

コードはPython 2.7ベースです。Python SDKの詳細な使用方法については、Python SDKのオブジェクトの操作ドキュメントをご参照ください。

1. 初期化設定

SecretId、SecretKey、Regionを含むユーザー属性を設定し、クライアントオブジェクトを作成します。



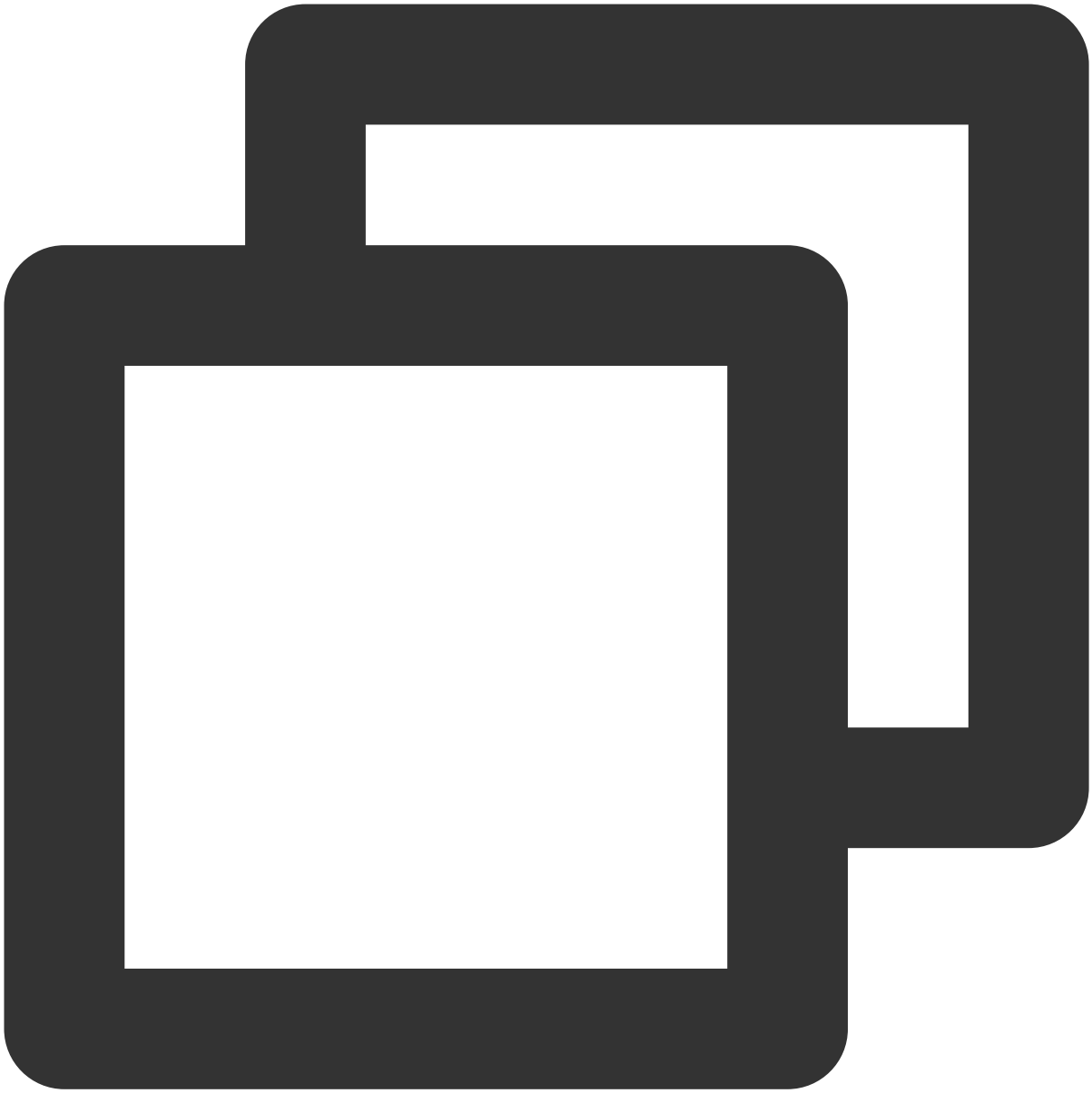
```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
from qcloud_cos import CosServiceError
from qcloud_cos import CosClientError
import sys
import os
import logging
import hashlib

logging.basicConfig(level=logging.INFO, stream=sys.stdout)
```

```
# SecretId、SecretKey、Regionを含むユーザー属性の設定
# APPIDはすでに設定から削除されていますので、パラメータのBucketにAPPIDを含めてください。Bucketに
secret_id = os.environ['COS_SECRET_ID']     # ユーザーのSecretIdです。サブアカウントのキー
secret_key = os.environ['COS_SECRET_KEY']   # ユーザーのSecretKeyです。サブアカウントのキ
region = 'ap-beijing'      # ご自身のRegionに置き換えてください。ここでは北京とします
token = None               # 一時キーのToken。一時キーの生成および使用ガイドについてはhttps:
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

2. シンプルアップロードオブジェクトの検証

（1）オブジェクトのチェックサムの計算

MD5検証アルゴリズムによってオブジェクトのチェックサムを取得します。ユーザーはご自身で他の検証アルゴリズムを選択することができます。

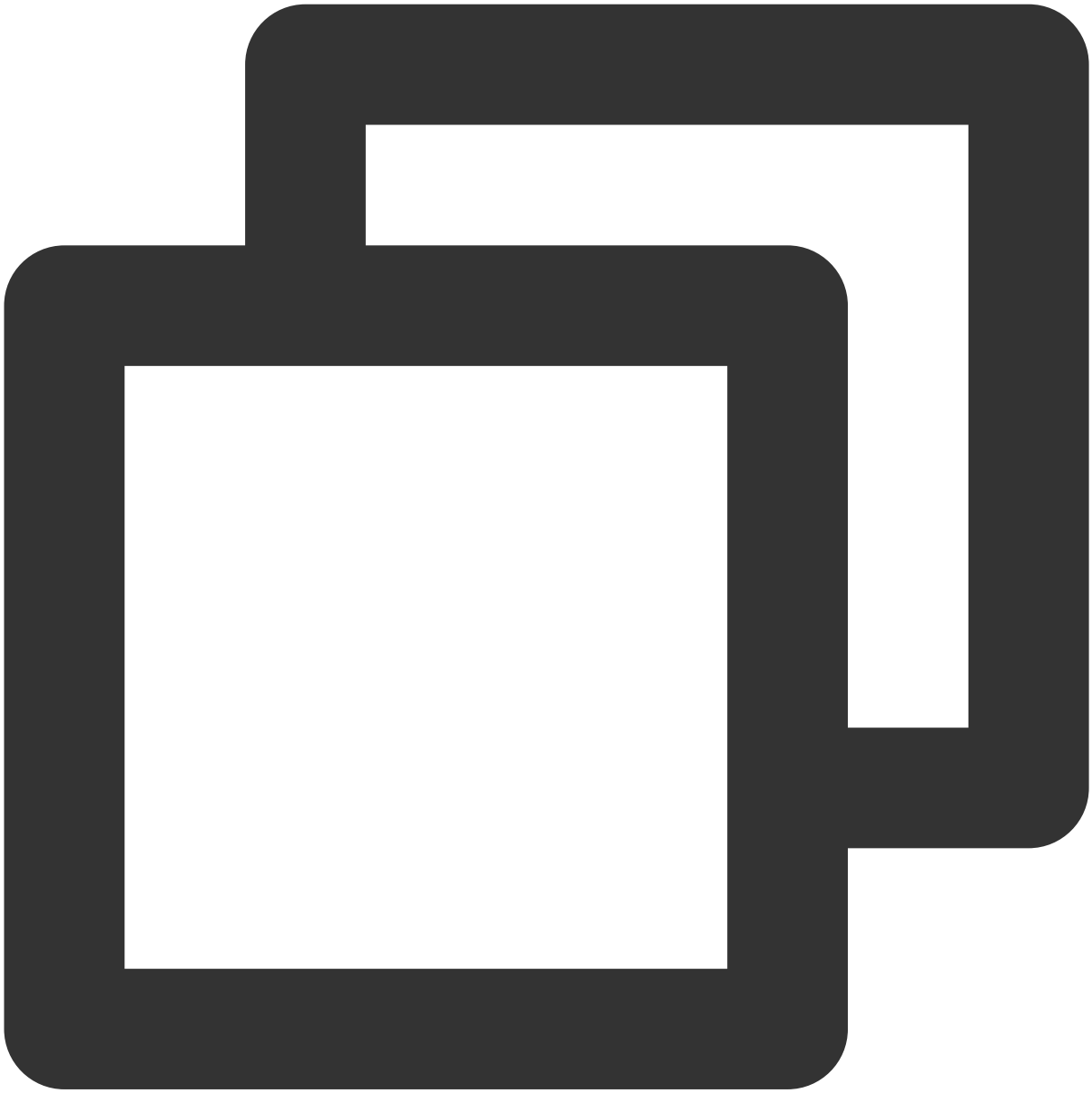


```
object_body = 'hello cos'
#オブジェクトのmd5チェックサムを取得
md5 = hashlib.md5()
md5.update(object_body)
md5_str = md5.hexdigest()
```

（2）オブジェクトのシンプルアップロード

コード内にEnableMD5=Trueと表示されていればMD5検証が有効になっており、Python SDKがContent-MD5を計算します。有効にするとアップロードの消費時間が増加し、COSサーバーが受信したオブジェクトのMD5チェッ

クサムとユーザーが設定したContent-MD5が一致した場合のみ、オブジェクトのアップロードが正常に行われます。
x-cos-meta-md5はユーザー定義のパラメータです（カスタムパラメータ名の形式はx-cos-meta-*）。このパラメータはオブジェクトのMD5チェックサムを表します。

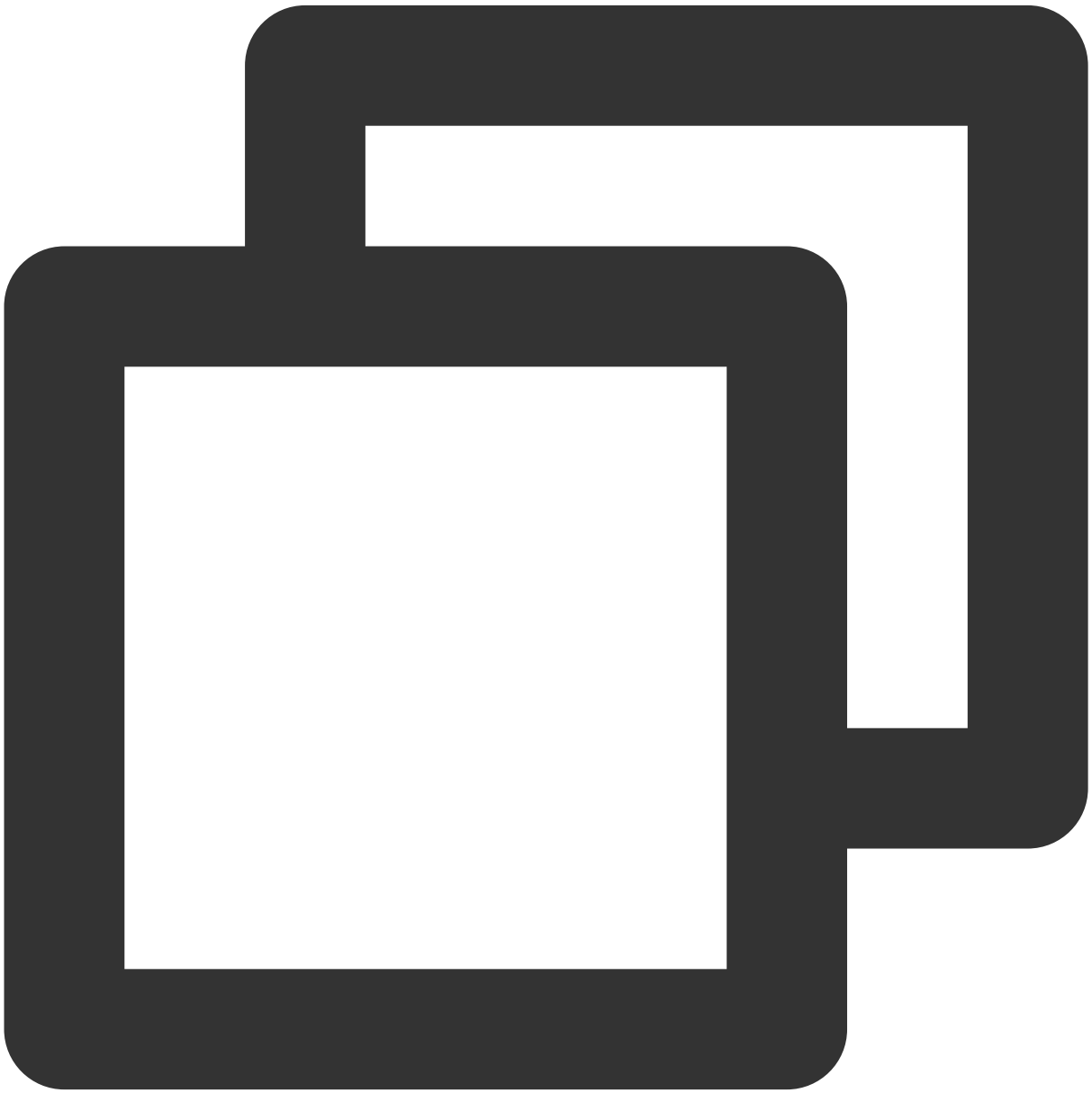


```
#オブジェクトのシンプルアップロードを行い、MD5検証を有効化します
response = client.put_object(
    Bucket='examplebucket-1250000000',      #ご自身のBucket名に置き換えます。examplebuck
    Body='hello cos',                   #アップロードするオブジェクトの内容
    Key='example-object-1',             #アップロードするオブジェクトのKey値に置き換えます
    EnableMD5=True,                     #アップロードのMD5検証を有効化します
```

```
    Metadata={                          #カスタムパラメータを設定し、オブジェクトのMD5チェック
        'x-cos-meta-md5' : md5_str
    }
)
print 'ETag: ' + response['ETag']      # ObjectのEtag値
```

（3）オブジェクトのダウンロード

オブジェクトをダウンロードし、ユーザー定義のパラメータを取得します。

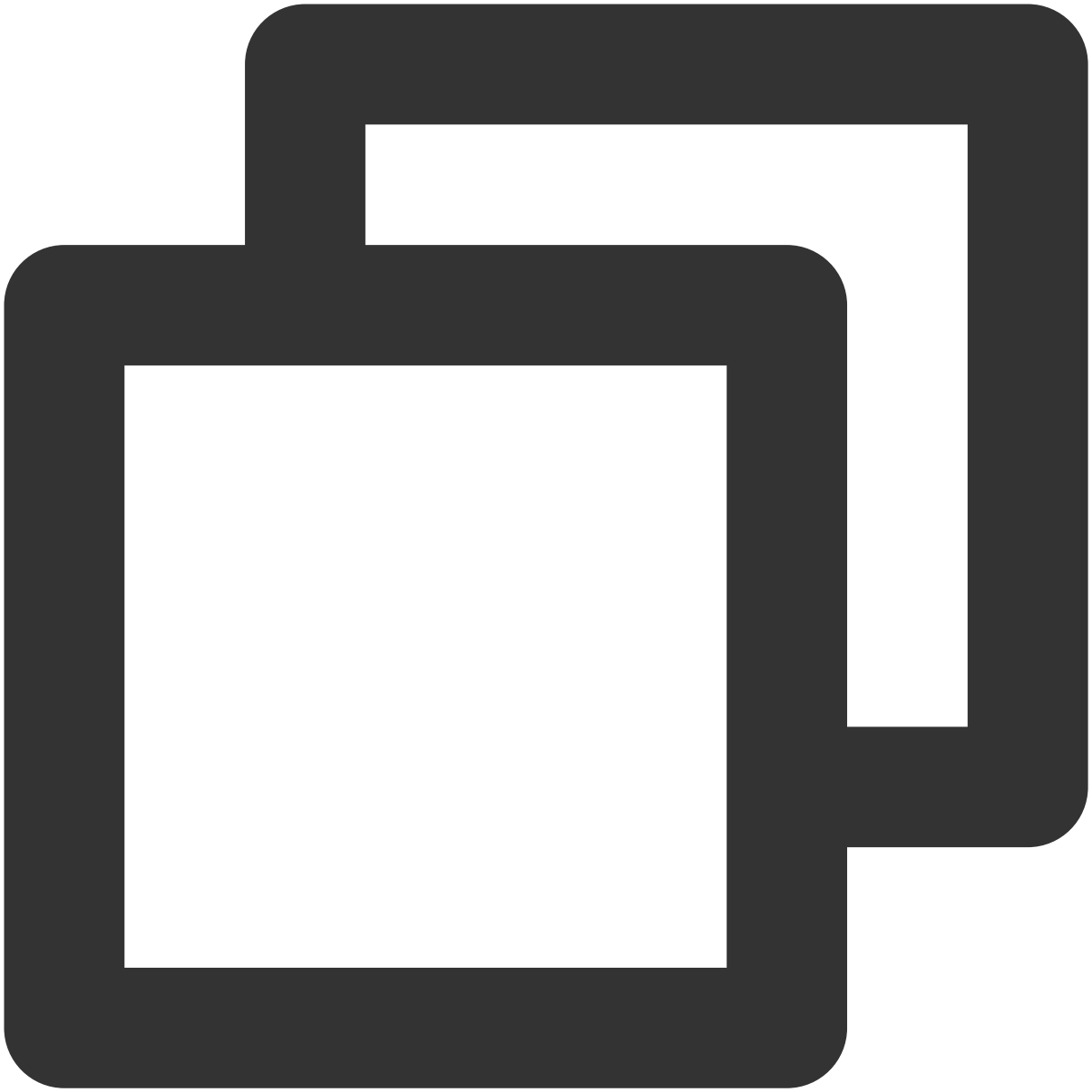


```
#オブジェクトのダウンロード
```

```
response = client.get_object(
    Bucket='examplebucket-1250000000',      #ご自身のBucket名に置き換えます。examplebuck
    Key='example-object-1'                  #ダウンロードしたオブジェクトのKey値
)
fp = response['Body'].get_raw_stream()
download_object = fp.read()                 #オブジェクトの内容を取得
print "get object body: " + download_object
print 'ETag: ' + response['ETag']
print 'x-cos-meta-md5: ' + response['x-cos-meta-md5']    #ユーザー定義のパラメータx-cos-
```

（4） オブジェクトの検証

オブジェクトのダウンロードに成功後、ユーザーはオブジェクトのチェックサムを再計算し（検証アルゴリズムはオブジェクトのアップロード時のものと一致させます）、ユーザー定義のパラメータx-cos-meta-md5と比較し、ダウンロードしたオブジェクトとアップロードしたオブジェクトの内容が一致するかどうかを検証します。

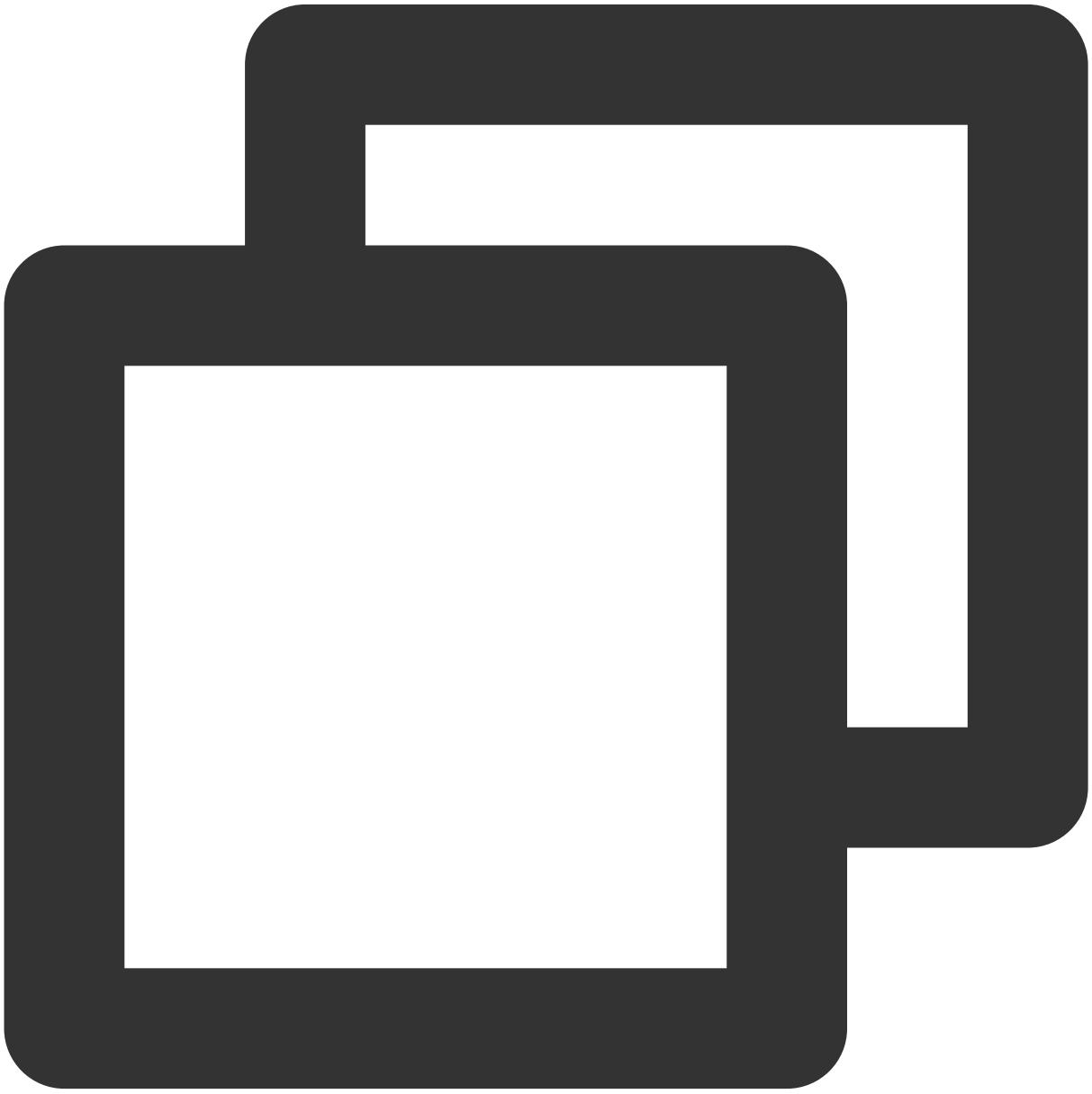


```
#ダウンロードオブジェクトのMD5チェックサムを計算
md5 = hashlib.md5()
md5.update(download_object)
md5_str = md5.hexdigest()
print 'download object md5: ' + md5_str

#ダウンロードオブジェクトのMD5チェックサムとアップロードオブジェクトのMD5チェックサムを比較し、オブ
if md5_str == response['x-cos-meta-md5']:
    print 'MD5 check OK'
else:
    print 'MD5 check FAIL'
```

3. マルチパートアップロードオブジェクトの検証

（1）オブジェクトのチェックサムの計算

オブジェクトの分割をシミュレートし、オブジェクト全体のチェックサムを計算します。以下ではMD5検証アルゴリズムによってオブジェクトのチェックサムを取得しますが、ユーザーはご自身で他の検証アルゴリズムを選択することができます。

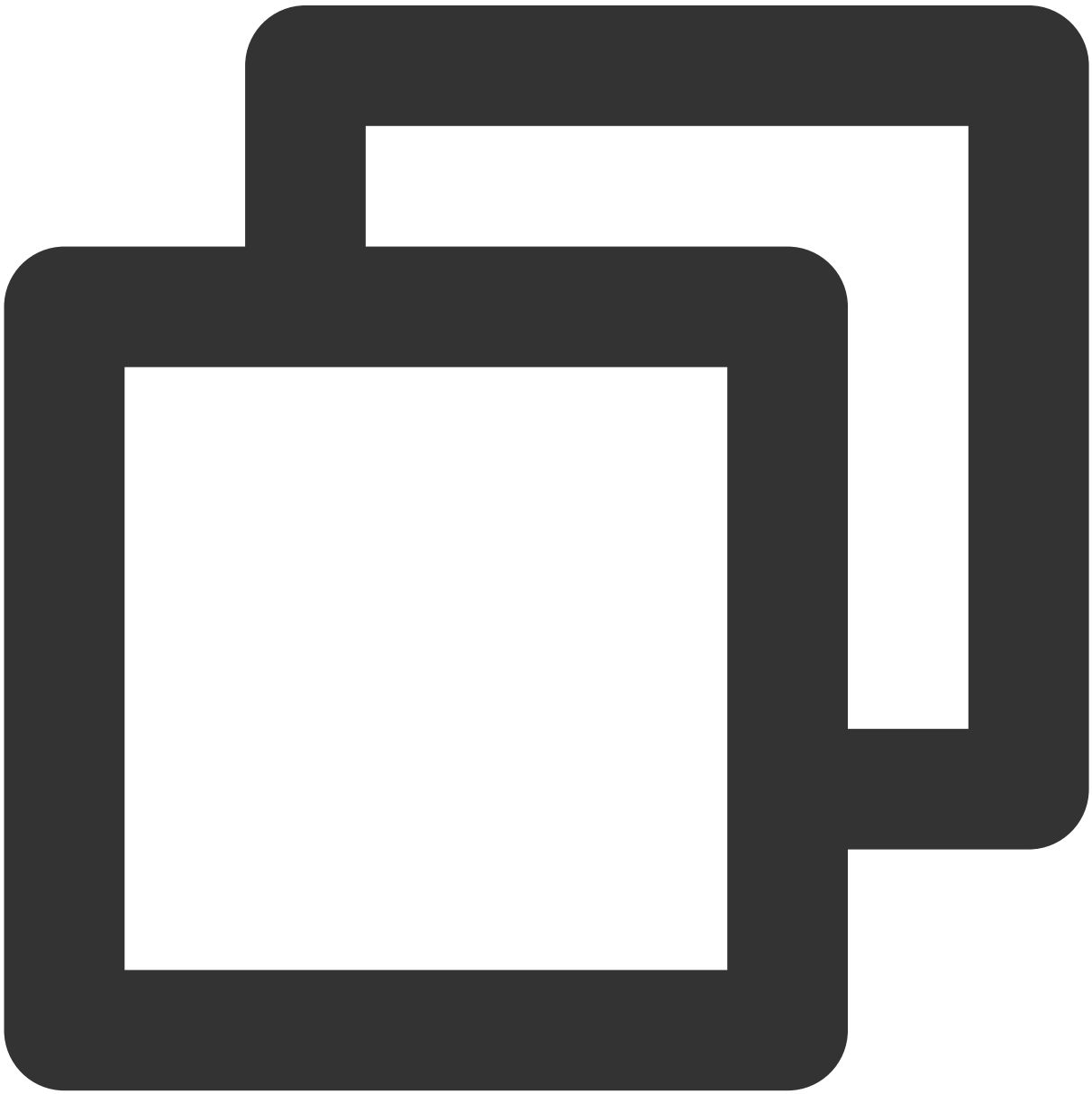


```
OBJECT_PART_SIZE = 1024 * 1024       #シミュレートした各パートのサイズ
OBJECT_TOTAL_SIZE = OBJECT_PART_SIZE * 1 + 123       #オブジェクト全体のサイズ
```

```
object_body = '1' * OBJECT_TOTAL_SIZE        #オブジェクトの内容

#オブジェクト内容全体のMD5チェックサムを計算
md5 = hashlib.md5()
md5.update(object_body)
md5_str = md5.hexdigest()
```

（2）マルチパートアップロードの初期化

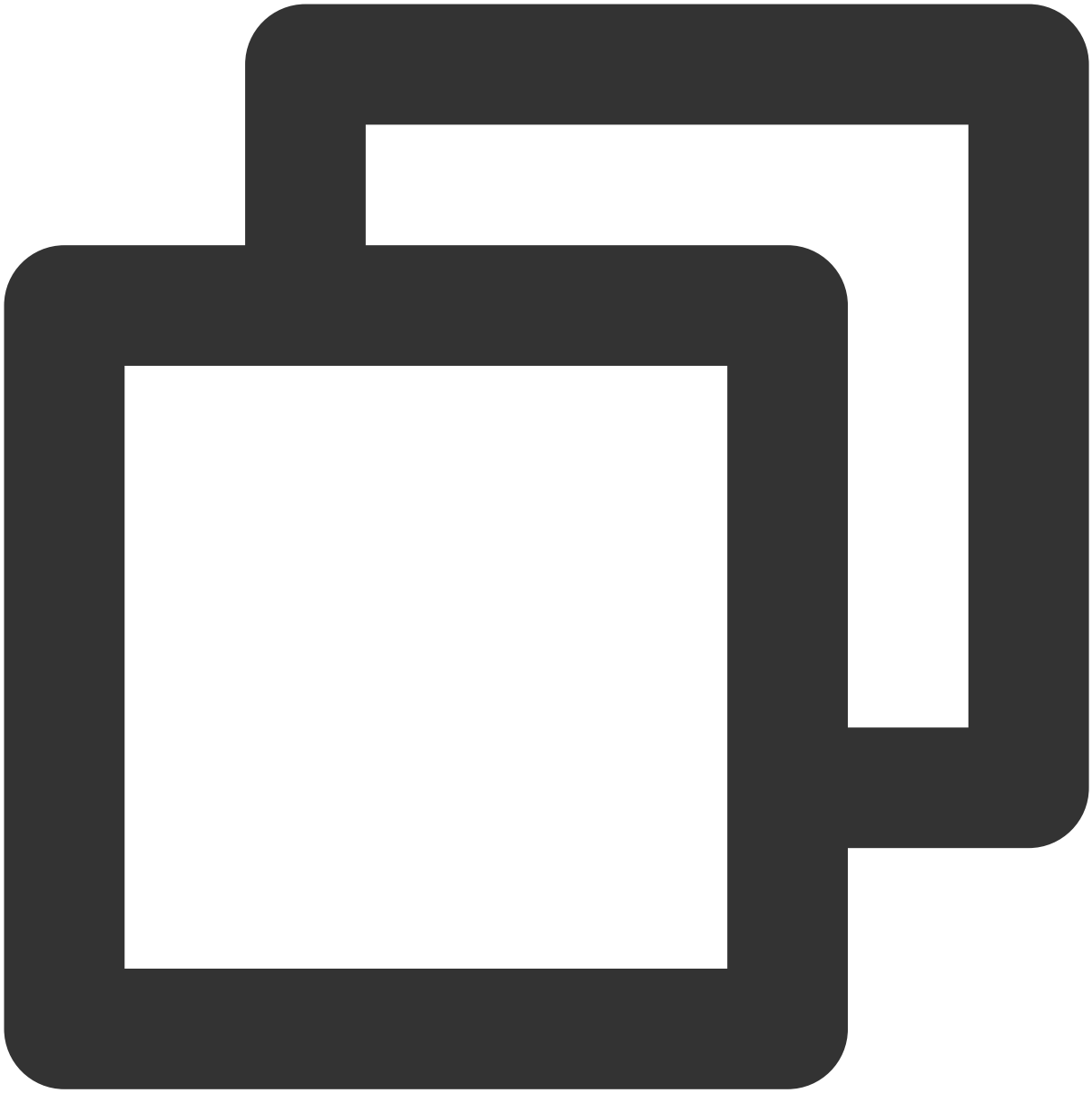
マルチパートアップロードの初期化の際、カスタムパラメータx-cos-meta-md5を設定し、オブジェクト全体のMD5チェックサムをパラメータの内容とします。



```
#マルチパートアップロードの初期化
response = client.create_multipart_upload(
    Bucket='examplebucket-1250000000',  #ご自身のBucket名に置き換えます。examplebucketは
    Key='exampleobject-2',              #アップロードするオブジェクトのKey値に置き換えます
    StorageClass='STANDARD',            #オブジェクトのストレージクラス
    Metadata={
        'x-cos-meta-md5' : md5_str      #カスタムパラメータの設定、MD5チェックサムに設定
    }
)
#マルチパートアップロードのUploadIdを取得
upload_id = response['UploadId']
```

（3）オブジェクトのマルチパートアップロード

オブジェクトのマルチパートアップロードは、オブジェクトを複数のパートに分割してアップロードを行うもので、最大10000パートの分割をサポートします。各パートのサイズは1MB～5GBで、最後のパートは1MB未満とすることができます。マルチパートアップロードの際は、各パートのPartNumber（番号）を設定する必要があります。パートの検証はEnableMD5=Trueで有効化でき、有効にするとアップロードの消費時間が増加します。このときPython SDKは各パートのContent-MD5を計算し、COSサーバーが受信したオブジェクトのMD5チェックサムとContent-MD5が一致した場合のみ、パートのアップロードを正常に行います。アップロードに成功すると、各パートのETagが返されます。

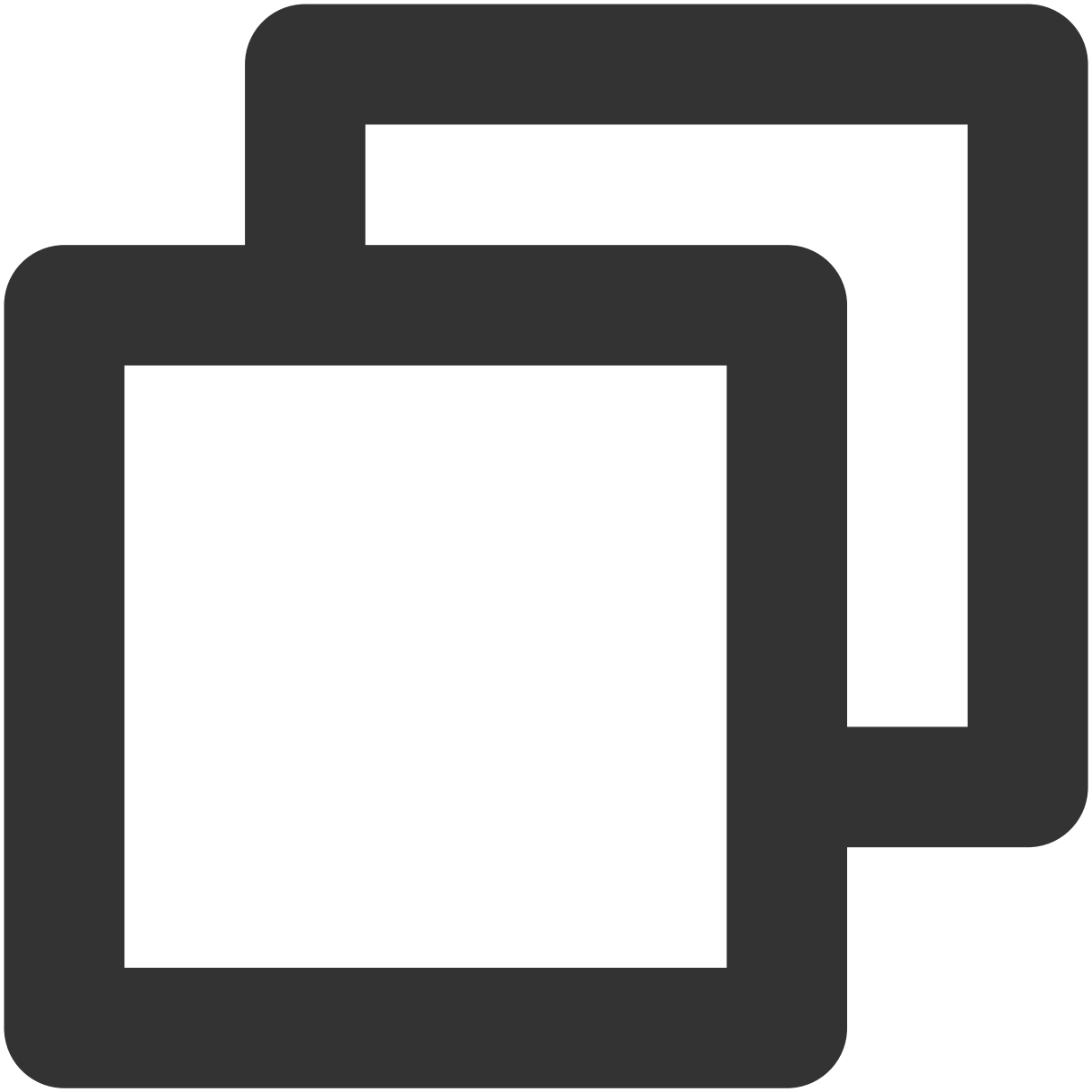


```
#マルチパートアップロードオブジェクト。各パートのサイズがOBJECT_PART_SIZEであり、最後のパートはO
part_list = list()
position = 0
left_size = OBJECT_TOTAL_SIZE
part_number = 0
while left_size > 0:
    part_number += 1
    if left_size >= OBJECT_PART_SIZE:
        body = object_body[position:position+OBJECT_PART_SIZE]
    else:
        body = object_body[position:]
```

```
        position += OBJECT_PART_SIZE
        left_size -= OBJECT_PART_SIZE

        #パートアップロード
        response = client.upload_part(
            Bucket='examplebucket-1250000000',   #ご自身のBucket名に置き換えます。examplebuck
            Key='exampleobject-2',               #オブジェクトのKey値
            Body=body,
            PartNumber=part_number,
            UploadId=upload_id,
            EnableMD5=True        #パート検証を有効化します。COSサーバーが各パートに対しMD5検証を行
        )
        etag = response['ETag']      #ETagは各パートのMD5値を表します
        part_list.append({'ETag' : etag, 'PartNumber' : part_number})
        print etag + ', ' + str(part_number)
```

（4）マルチパートアップロードの完了

すべてのパートのアップロードが完了した後、マルチパートアップロード完了操作を行う必要があります。各パートのETagとPartNumberをそれぞれ対応させ、COSサーバーを使用してパートの正確性を検証します。マルチパートアップロードの完了後に返されるETagは合体後のオブジェクトの一意のタグ値を表すもので、オブジェクト内容全体のMD5チェックサムを表すものではありません。このためオブジェクトをダウンロードする際は、カスタムパラメータによって検証を行ってください。

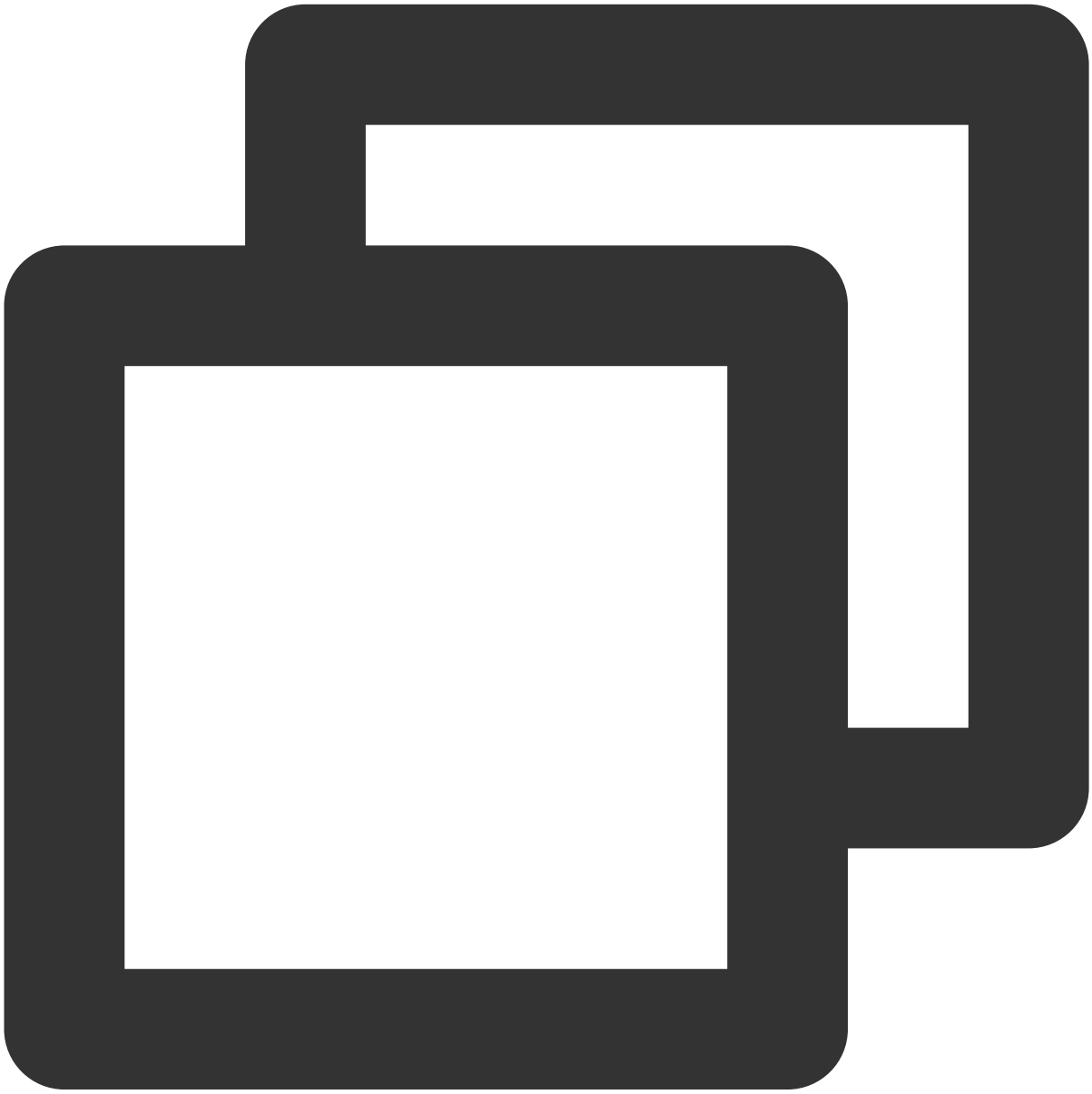


```
#マルチパートアップロードの完了
response = client.complete_multipart_upload(
    Bucket='examplebucket-1250000000',  #ご自身のBucket名に置き換えます。examplebucketは
    Key='exampleobject-2',              #オブジェクトのKey値
    UploadId=upload_id,
    MultipartUpload={       #各パートのETagとPartNumberそれぞれの対応が必要
        'Part' : part_list
    },
)

#ETagは合体後のオブジェクトの一意のタグ値を表すもので、この値はオブジェクトのMD5チェックサムではな
```

```
print "ETag: " + response['ETag']
print "Location: " + response['Location']    #URLアドレス
print "Key: " + response['Key']
```

（5）オブジェクトのダウンロード

オブジェクトをダウンロードし、ユーザー定義のパラメータを取得します。

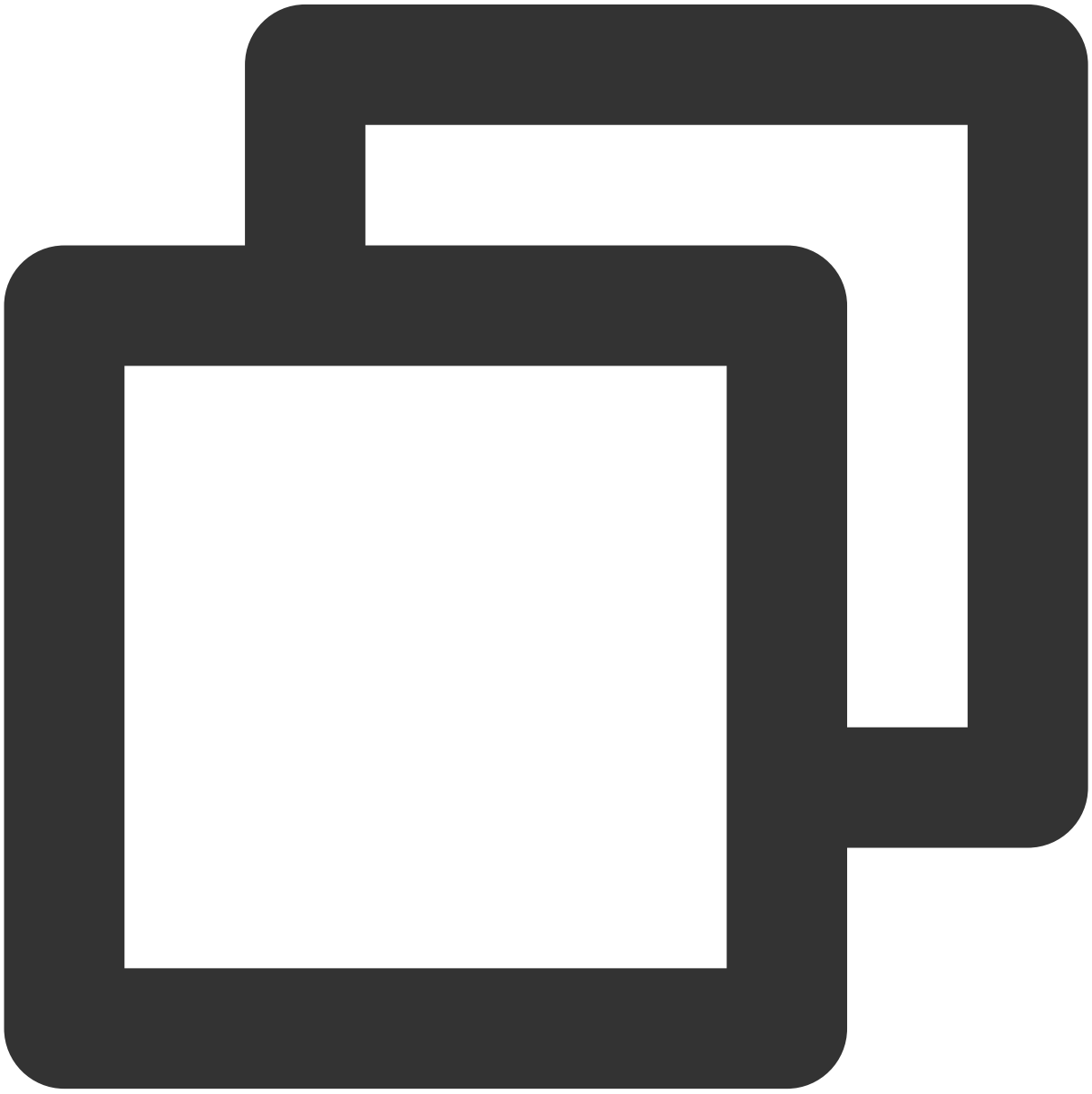


```
# オブジェクトのダウンロード
response = client.get_object(
    Bucket='examplebucket-1250000000',  #ご自身のBucket名に置き換えます。examplebucketは
```

```
    Key='exampleobject-2'                #オブジェクトのKey値
)
print 'ETag: ' + response['ETag']                          #オブジェクトのETagはオブジェク
print 'x-cos-meta-md5: ' + response['x-cos-meta-md5']      #ユーザー定義のパラメータx-cos
```

（6）オブジェクトの検証

オブジェクトのダウンロードに成功後、ユーザーはオブジェクトのMD5チェックサムを再計算し、ユーザー定義のパラメータx-cos-meta-md5と比較し、ダウンロードしたオブジェクトとアップロードしたオブジェクトの内容が一致するかどうかを検証します。



```
#ダウンロードオブジェクトのMD5チェックサムを計算
fp = response['Body'].get_raw_stream()
DEFAULT_CHUNK_SIZE = 1024*1024
md5 = hashlib.md5()
chunk = fp.read(DEFAULT_CHUNK_SIZE)
while chunk:
    md5.update(chunk)
    chunk = fp.read(DEFAULT_CHUNK_SIZE)
md5_str = md5.hexdigest()
print 'download object md5: ' + md5_str

#ダウンロードオブジェクトのMD5チェックサムとアップロードオブジェクトのMD5チェックサムを比較し、オブ
if md5_str == response['x-cos-meta-md5']:
    print 'MD5 check OK'
else:
    print 'MD5 check FAIL'
```

サードパーティアプリケーションでのCOSの使用
WordPressリモート添付ファイルをCOSに保存する

最終更新日：：2024-06-26 10:47:37

概要

WordPressは、PHP言語を使用して開発されたブログプラットフォームです。ユーザーは、PHPとMySQLデータベースをサポートするサーバーに、自分のウェブサイトを設置することができます。また、WordPressをコンテンツ管理システム(CMS)として使用することも可能です。
WordPressは強力な機能と拡張性を備えています。それは主にプラグインの多さによるもので、機能を拡充しやすく、ウェブサイトが備えるべき機能を基本的に完備しており、サードパーティプラグインによってすべての機能を実現することができます。
ここでは、プラグインを使用してリモート添付ファイルを実装し、WordPressのメディアライブラリ添付ファイルをTencent CloudのCloud Object Storage（COS）に保存する方法についてご説明します。
COSは高い拡張性、低コスト、高い信頼性と安全性などの特徴を備えています。メディアライブラリ添付ファイルをCOSに保存すると次のようなメリットがあります。
添付ファイルの信頼性が高まります。
サーバーが添付ファイルのために追加のストレージスペースを用意する必要がありません。
ユーザーが画像添付ファイルを確認する際は直接COSサーバーに接続するため、サーバーのダウンストリーム帯域幅/トラフィックを占有せず、ユーザーのアクセス速度がより速くなります。
Tencent CloudのContent Delivery Network（CDN）と併用することで、ユーザーの画像添付ファイル表示速度がさらに早くなり、ウェブサイトのアクセス速度が最適化されます。

前提条件

1. COSバケットがすでにあること。ない場合はバケットの作成操作ガイドをご参照ください。
2. サーバーを作成済みであること。例えばCloud Virtual Machine（CVM）などです。関連ガイドについてはCVM製品ドキュメントをご参照ください。

実践手順

Wordpressのデプロイ

Tencent CVMをベースにしてスピーディーにWordPressを構築するには、イメージによるデプロイと手動デプロイという2つの方法があります。業務ウェブサイトに比較的高い拡張性が求められる場合は、手動での構築が可能です。詳細については次のガイドをご参照ください。

WordPress個人サイトの手動構築（Linux）

WordPress個人サイトの手動構築（Windows）

ここではイメージによるWordPressのデプロイについてご説明します。イメージデプロイは便利かつスピーディーな方法です。操作手順は次のとおりです。

1. イメージからWordPressを起動します。

1.1 CVMコンソールにログインし、インスタンス管理ページの**新規作成**をクリックします。

1.2 ページのプロンプトに従ってモデルを選択し、**インスタンス構成 > イメージ**で**イメージマーケットプレイス**をクリックし、**イメージマーケットプレイスから選択**を選択します。

1.3 「イメージマーケットプレイス」のポップアップウィンドウで、**基本ソフトウェア**を選択し、**wordpress**と入力して検索します。

1.4 必要なイメージを選択します。例として**WordPressブログプログラミング_v5.5.3(CentOS | LAMP)を選択し、無料で使用**をクリックします。

1.5 購入完了後、CVMコンソールにログインし、先ほど作成したインスタンスにセキュリティグループをバインドします。ポート80を許可するインバウンドルールを追加する必要があります。

2. インスタンスの管理ページで、このCVMインスタンスの**パブリックIP**をコピーし、ローカルブラウザでアドレス `http://パブリックIP/wp-admin` にアクセスし、次のようにWordPressウェブサイトのインストールを開始します。

2.1 Wordpressの言語を選択し、**Continue**をクリックします。

2.2 必要に応じてWordPressのサイトのタイトル、管理者ユーザー名、管理者パスワード、メールアドレスを入力します。

2.3 **WordPressをインストールする**をクリックします。

2.4 **ログイン**をクリックします。

3. WordPressを新バージョンの6.0.2にアップグレードします。

ダッシュボード左側のメニューをクリックし、「更新」メニューに進み、バージョン6.0.2に更新します。

COSバケットの作成

1. **パブリック読み取り・プライベート書き込み**のバケットを作成します。バケットのリージョンはWordPressブログプラットフォームのCVMの実行リージョンと同一にすることをお勧めします。作成の詳細については、バケットの作成のドキュメントをご参照ください。

2. 作成したバケットを**バケットリスト**から見つけ、そのバケット名をクリックし、バケットのページに進みます。

3. 左側ナビゲーションバーで**概要**をクリックし、アクセスドメイン名を確認して記録します。

プラグインのインストールと設定

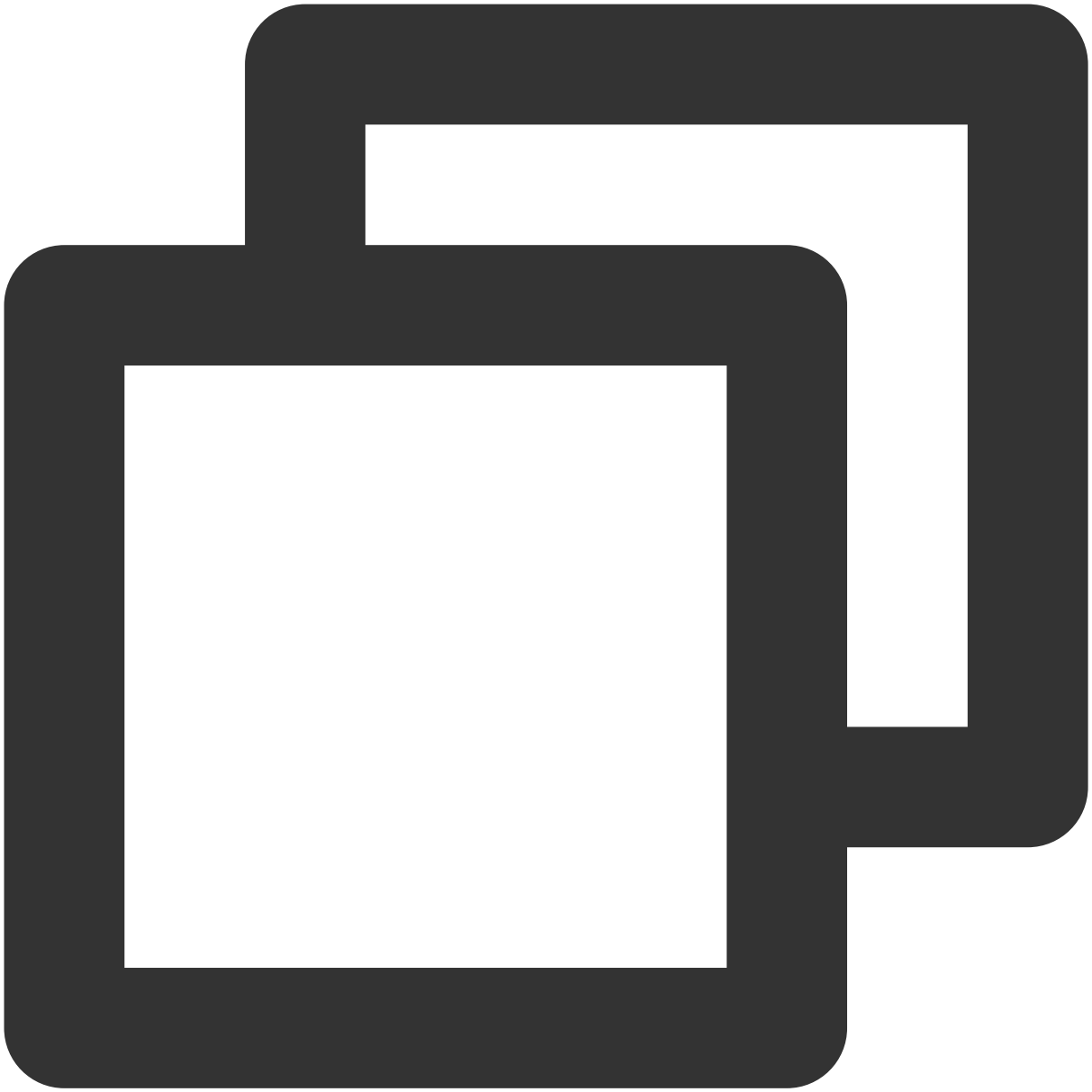
プラグインのインストール方法には、プラグインライブラリからのインストールとソースコードからのインストールがあります。

プラグインライブラリからのインストール（推奨）

WordPressバックエンドで**プラグイン**をクリックし、**tencentcloud-cos**プラグインを直接検索し、**今すぐインストール**をクリックすればインストールできます。

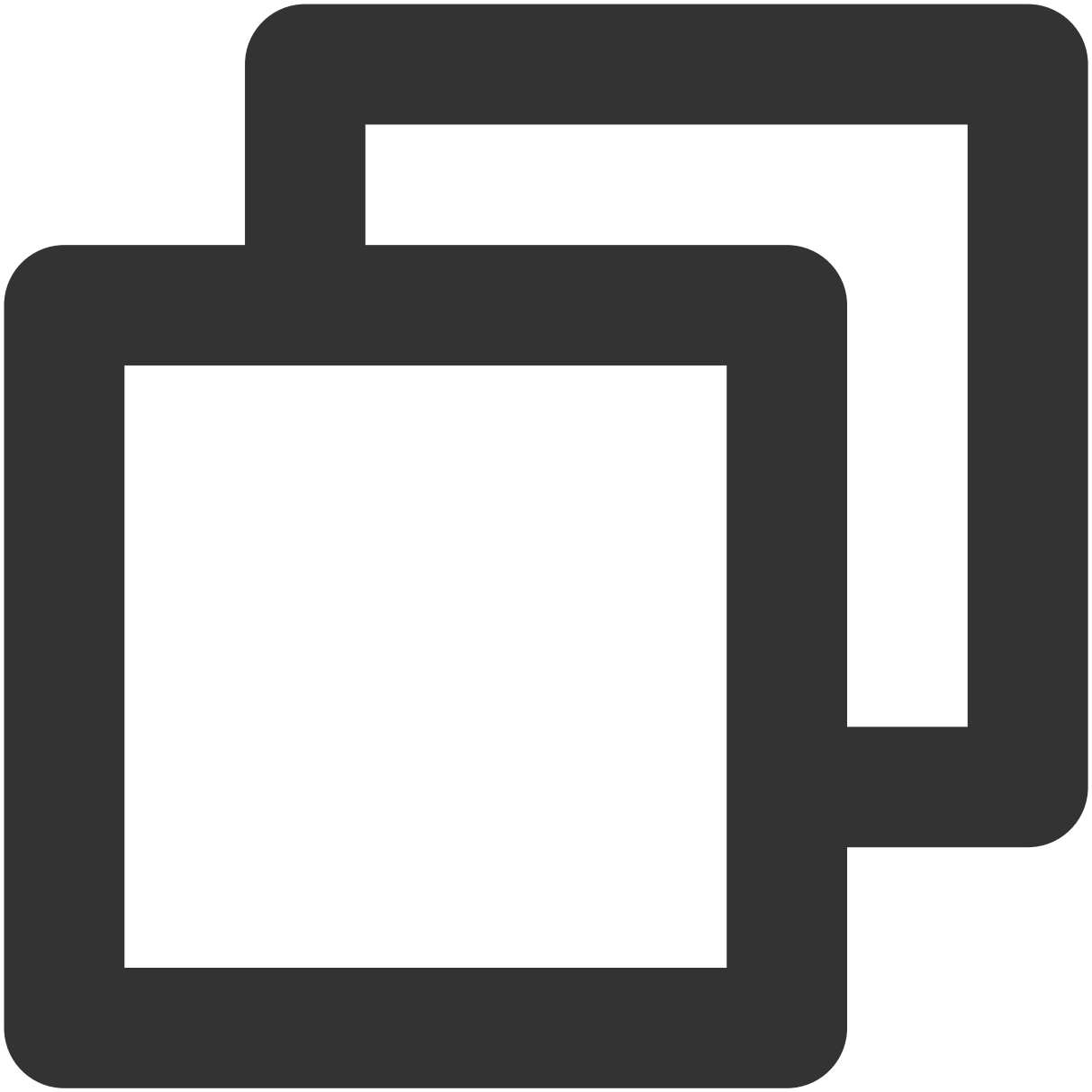
ソースコードからのインストール

まずプラグインのソースコードをダウンロードした後、プラグインのソースコードをWordPressプラグインディレクトリ `wp-content/plugins` にアップロードし、最後にバックエンドで起動します。
以下はUbuntuでのプラグインインストールの例です。
1. wp-contentの親ディレクトリに進みます。



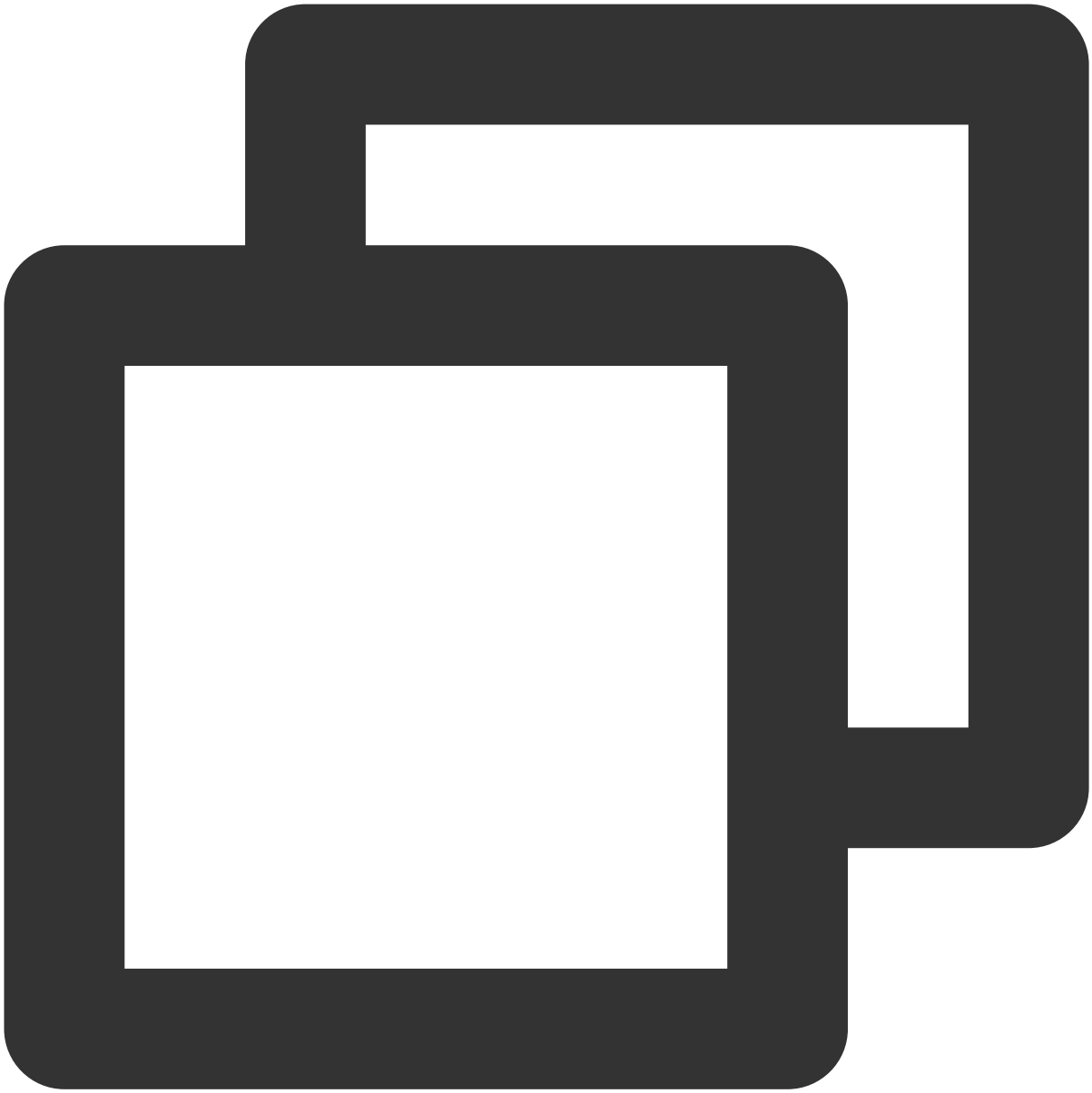
```
cd /var/www/html
```

2. 権限を追加します。



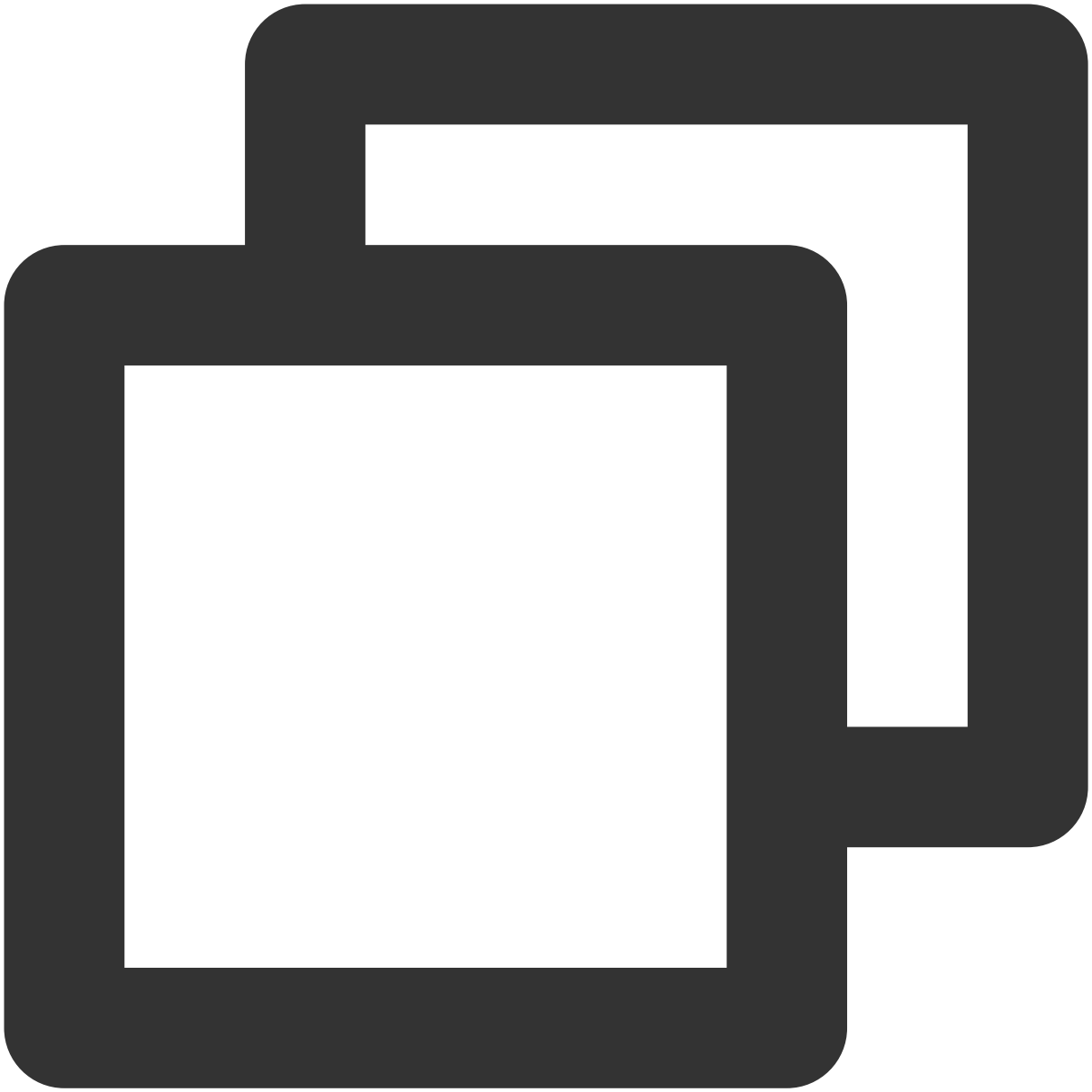
```
chmod -R 777 wp-content
```

3. プラグインディレクトリを作成します。



```
cd wp-content/plugins/
mkdir tencent-cloud-cos
cd tencent-cloud-cos
```

4. プラグインをプラグインディレクトリにダウンロードします。



```
wget https://cos5.cloud.tencent.com/cosbrowser/code/tencent-cloud-cos.zip
unzip tencent-cloud-cos.zip
rm tencent-cloud-cos.zip -f
```

5. 「プラグイン」の左側のメニューをクリックすると、このプラグインが見つかります。このプラグインをクリックして起動します。

プラグインの設定

プラグインtencent-cloud-cosにCOSバケットの情報を設定します。

1. 「設定」ボタンをクリックし、プラグインtencent-cloud-cosを設定します。

2. ページ内でCOSの関連情報を設定します。設定の説明については次の表をご参照ください。

設定項目	設定値
SecretId、SecretKey	アクセスキー情報。Tencent Cloud APIキーに進んで作成と取得を行うことができます
所属リージョン	バケット作成時に選択したリージョン
スペース名	バケット作成時にカスタマイズしたバケット名。例：examplebucket-1250000000
アクセスドメイン名	COSのデフォルトのバケットドメイン名であり、ユーザーのバケット作成時に、システムがバケット名およびリージョンに基づいて自動的に生成します。COSコンソール に進み、バケットの概要>ドメイン名情報で確認することができます
自動リネーム	ファイルをCOSにアップロードすると自動的にリネームされ、同名ファイルとの競合を防止します。リネームは指定の形式で行うことができます
ローカルに保存しない	有効にすると、ソースファイルをローカルに保存しません
リモートファイルの維持	有効にすると、ファイルを削除したときにローカルファイルのレプリカだけを削除し、リモートCOSバケット内のファイルレプリカは維持されるため、復元に便利です
サムネイル禁止	有効にすると、対応するサムネイルファイルをアップロードしません
Cloud Infinite	Cloud Infiniteサービスを有効にすると、画像の編集、圧縮、形式変換、ウォーターマーク追加などの操作を行うことができます。詳細については、Cloud Infinite製品の紹介をご参照ください
ファイル審査	ファイル審査を有効にすると、画像、ビデオ、オーディオ、テキスト、ドキュメント、ウェブページなどのマルチメディアのコンテンツセキュリティに対してインテリジェント審査サービスを行うことができます。ユーザーが、ポルノ・低俗、違法・不正、不快感を与えるなどの禁止コンテンツを効果的に識別し、運営リスクを回避できるようにサポートします。詳細についてはコンテンツ審査の概要をご参照ください
ドキュメントプレビュー	ドキュメントプレビューを有効にすると、ファイルを画像、PDFまたはHTML5ページにトランスコードすることができ、ドキュメントコンテンツのページ表示の問題を解決することができます。詳細についてはドキュメントプレビューの概要をご参照ください
デバッグ	エラー、異常、警告情報を記録します

3. 設定完了後に、**設定を保存**をクリックすれば完了です。

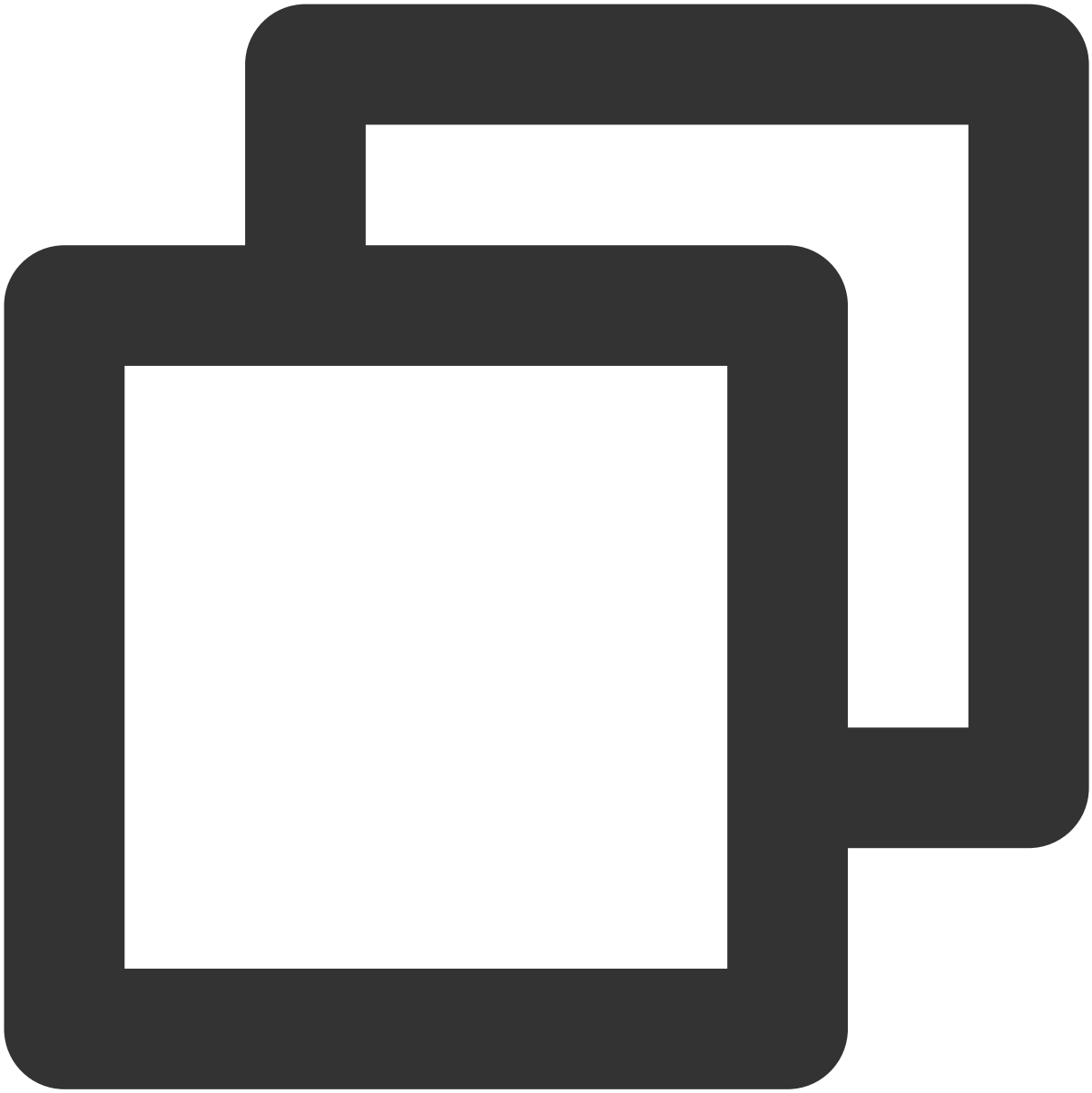
Wordpress添付ファイルのCOSへの保存を検証する

Wordpressで画像付きの記事を1件作成し、画像がCOSに保存されるかどうかを確認します。
1. 画像付きの記事を1件作成します。Wordpressのダッシュボードで、「記事」の左側のメニューをクリックします。
2. WordPressでデフォルトで生成される「Hello world!」の記事を編集します。
3. 右側の「+」ボタンをクリックします。
4. 画像を1枚選択してアップロードします。
5. アップロード完了後、アップロード済みの画像のURLを表示し、画像のアドレスがCOSのアドレスになっていることを確認します。例えば `https://wd-125000000.cos.ap-nanjing.myqcloud.com/2022/10/立夏-1200x675.jpeg` などで、形式が `https://<BucketName-APPID>.cos.<Region>.myqcloud.com/<ObjectKey>` となっていれば、画像がCOSバケットにアップロードされていることを表します。
6. COSコンソールにログインすると、先ほどアップロードした画像がCOSバケット内にあることを確認できます。
説明：
上記のテストに成功した後、必要があれば続いて古いリソースをCOSバケットに同期させます（COSCMDツールまたはCOS Migrationツールを使用できます）。**これを行わなければ、古いリソースをバックエンドで正常にプレビューできなくなります**。同期が完了すると、back-to-origin設定を有効化できます。下記のback-to-originの設定をご参照ください。

拡張

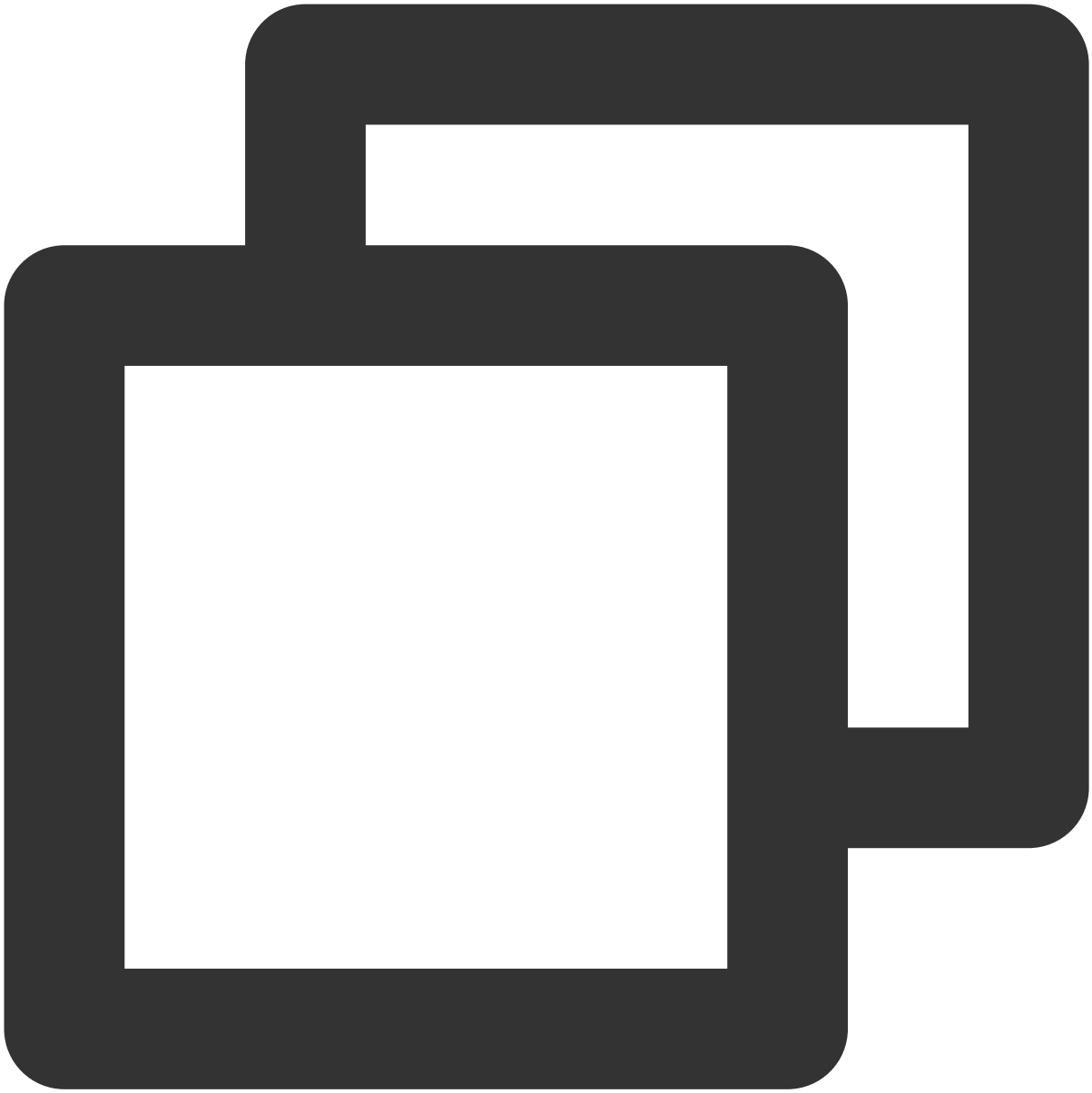
1. CDNアクセラレーションを使用したアクセス
バケットにCDNアクセラレーションを設定したい場合は、CDNアクセラレーションの設定のドキュメントをご参照ください。プラグイン設定で、URLのプレフィックスをデフォルトのCDNアクセラレーションドメイン名またはカスタムアクセラレーションドメイン名に変更するだけで設定できます。
2. データベース内のリソースアドレスの置き換え
新規作成のサイトを除き、データベースには必ず古いリソースリンクアドレスが存在するため、リソースアドレスの置き換えを行う必要があります。プラグインで置き換え機能を提供しています。最初に置き換えを行う前にはバックアップを忘れないようにしてください。
古いドメイン名に元のリソースドメイン名を入力します（例：`https://example.com/`）
新しいドメイン名に現在のリソースドメイン名を入力します（例：`https://img.example.com/`）

3. クロスドメインアクセスの設定
対応するリソースリンクを記事に引用する際、コンソールがクロスドメインエラー `No 'Access-Control-Allow-Origin' header is present on the requested resource` を表示することがあります。これはheaderを追加していないことが原因です。クロスドメインアクセスCORSの設定でHTTP Header追加の設定を行う必要があります。設定を行うための2つの手段を次でご説明します。
COSコンソールで設定
説明：
クロスドメイン設定の操作手順に関しては、クロスドメインアクセスの設定のドキュメントをご参照ください。
CDNコンソールで設定
全ドメイン名を許可する場合、設定は次のようになります。



```
Access-Control-Allow-Origin: *
```

個人のドメイン名のアクセスのみを許可する場合、設定は次のようになります。



```
Access-Control-Allow-Origin: https://example.com
```

4. back-to-originの設定

WordPressバックエンドメディアライブラリにリソースをアップロードしない場合は、back-to-origin有効化の設定をお勧めします。詳細な手順については、back-to-originの設定のドキュメントをご参照ください。

back-to-origin設定を有効化すると、クライアントがCOSソースファイルに初めてアクセスした際、COSはオブジェクトにヒットしないことを検出し、クライアントに対しHTTPステータスコード302を返してback-to-originアドレスに対応するアドレスにリダイレクトします。このときオブジェクトはオリジンサーバーからクライアントに提供され、アクセスが保証されます。同時に、COSはオリジンサーバーからこのファイルをコピーして、バケッ

トの対応するディレクトリに保存します。2回目のアクセスの際は、COSがオブジェクトに直接ヒットし、クライアントに返します。

Ghostブログアプリケーションの添付ファイルをCOSに保存する

最終更新日：：2024-06-26 10:47:37

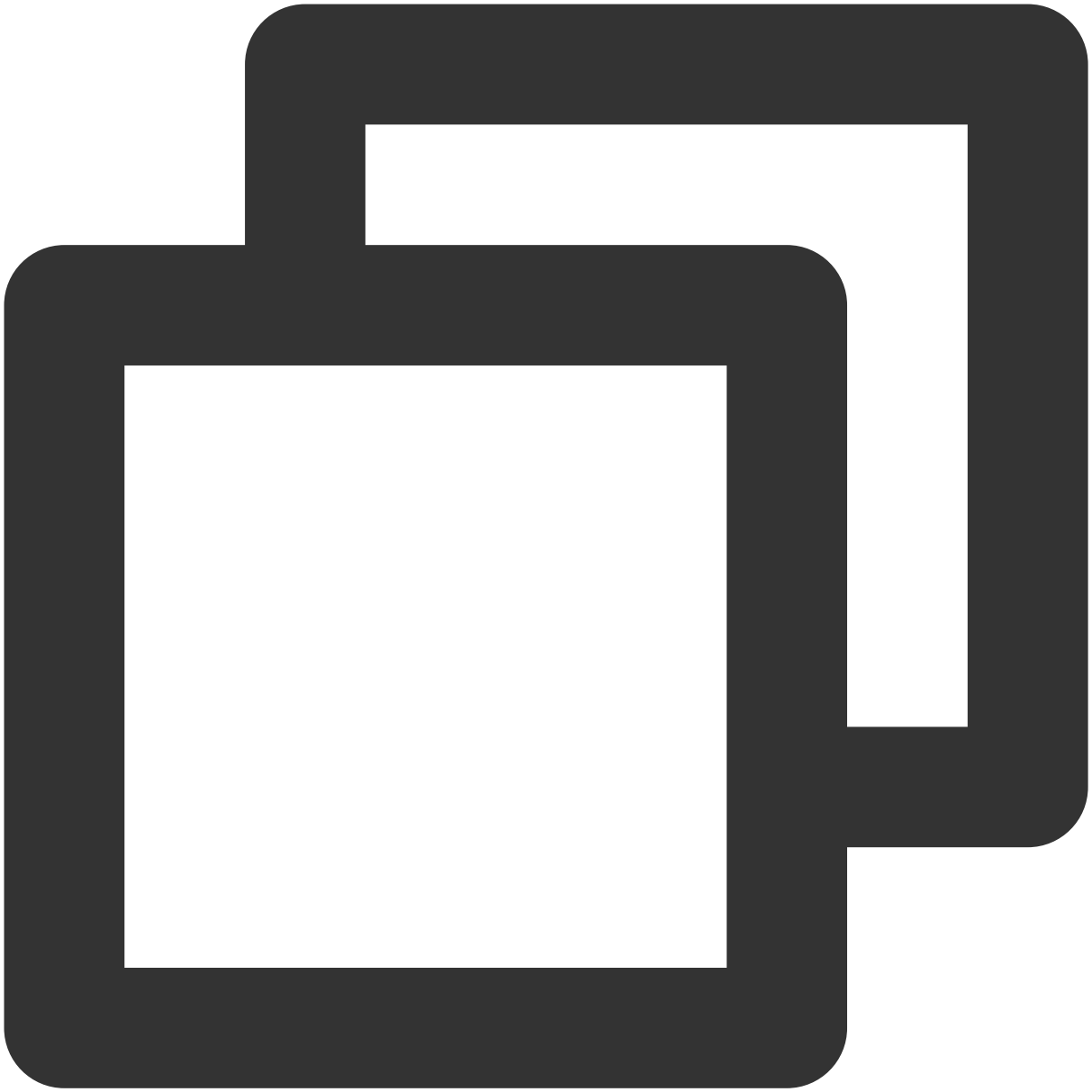
概要

GhostはNode.jsをベースにした、ブログ系のウェブサイトをスピーディーに構築できるフレームワークです。開発者はGhostの公式cliツールによって個人ウェブサイトをワンクリックで生成し、CVMやDocker上にデプロイすることができます。
添付ファイルのアップロードはブログ系ウェブサイトにとって欠かせない機能です。Ghostは添付ファイルをデフォルトでローカルに保存します。ここではプラグインによって添付ファイルをTencent Cloud Object Storage（COS）に保存する方法についてご説明します。フォーラムの添付ファイルをCOSに保存すると次のようなメリットがあります。
添付ファイルの信頼性が高まります。
サーバーがフォーラム添付ファイルのために追加のストレージスペースを用意する必要がありません。
ユーザーが画像添付ファイルを確認する際は直接COSサーバーに接続するため、サーバーのダウンストリーム帯域幅/トラフィックを占有せず、ユーザーのアクセス速度がより速くなります。
Tencent Cloud Content Delivery Network（CDN）と併用することで、フォーラムユーザーの画像添付ファイル表示速度がさらに早くなります。

準備作業

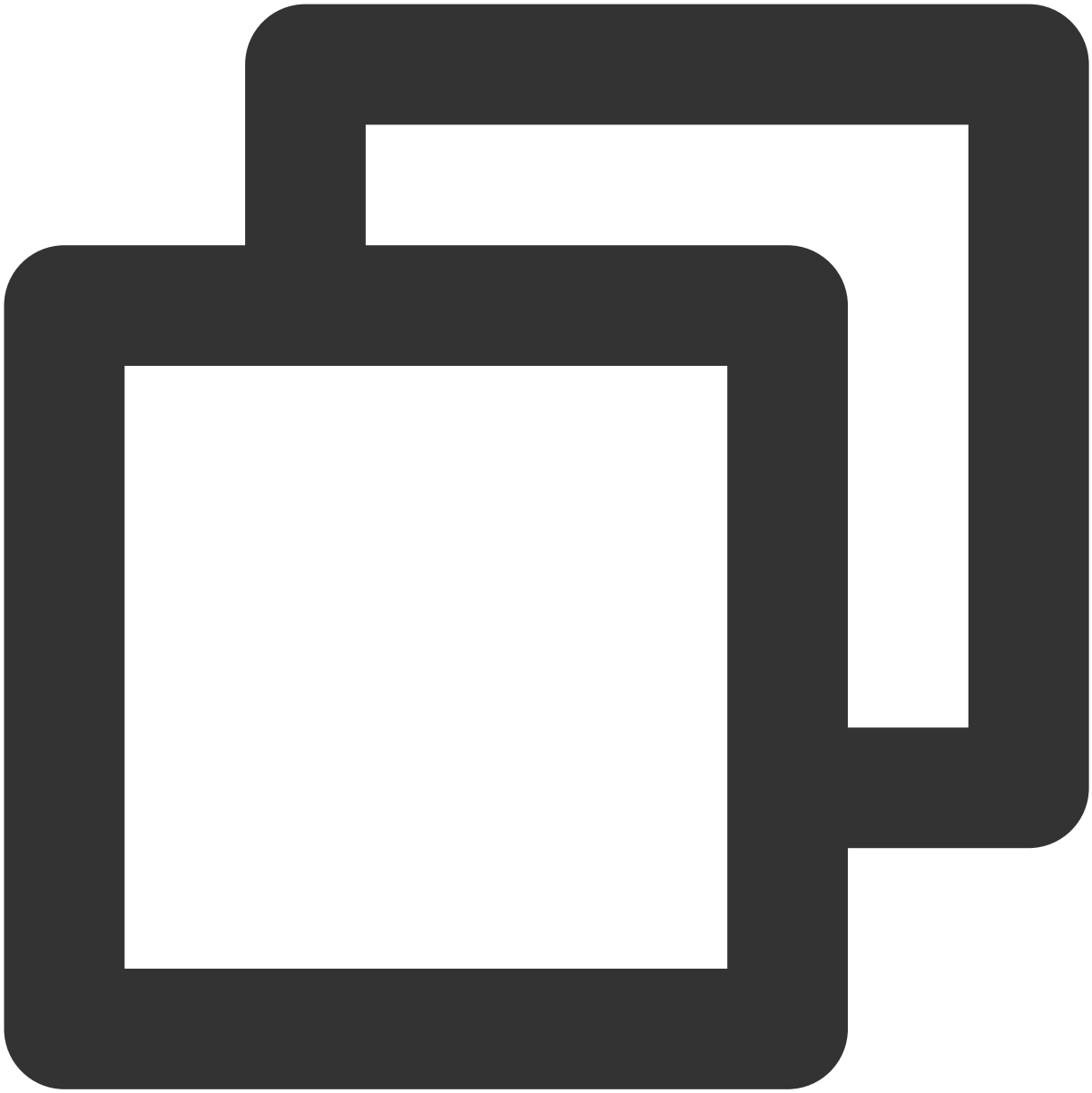
Ghostウェブサイトの構築

1. Node.js環境をインストールします。
2. ghost-cliをインストールします。



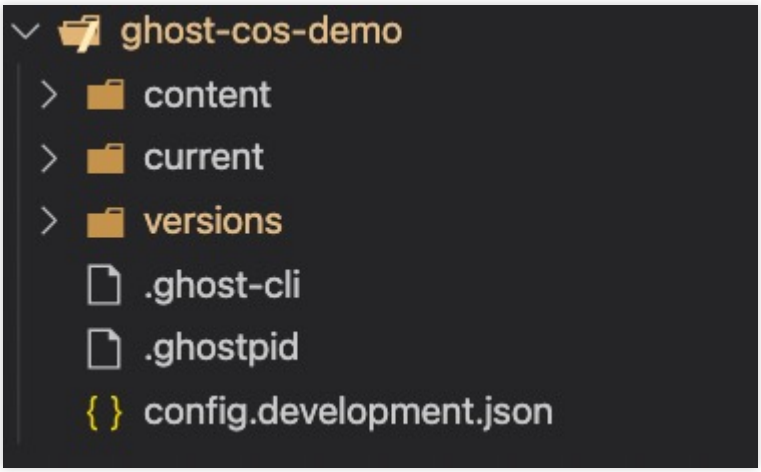
```
npm install ghost-cli@latest -g
```

3. プロジェクトを作成し、このプロジェクトのルートディレクトリで次のコマンドを実行します。

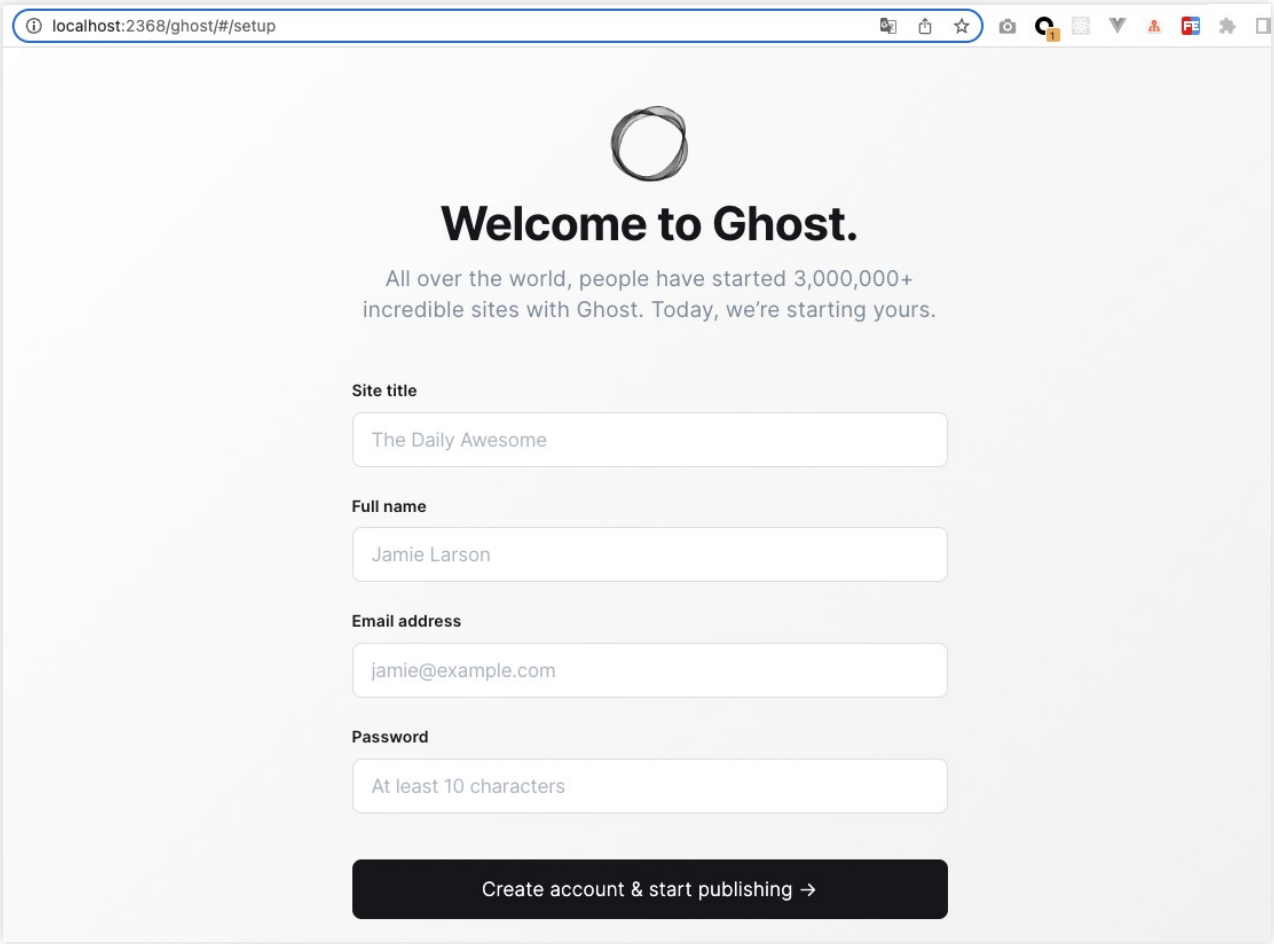


```
ghost install local
```

作成に成功すると、プロジェクトの構造は下図のようになります。



4. ブラウザを開き、localhost:2368に進むと登録ページが表示されます。登録し、管理バックエンドに進みます。



COSバケットの作成

1. COSコンソールで、アクセス権限が**パブリック読み取り・プライベート書き込み**のバケットを作成します。操作ガイドについてはバケットの作成をご参照ください。

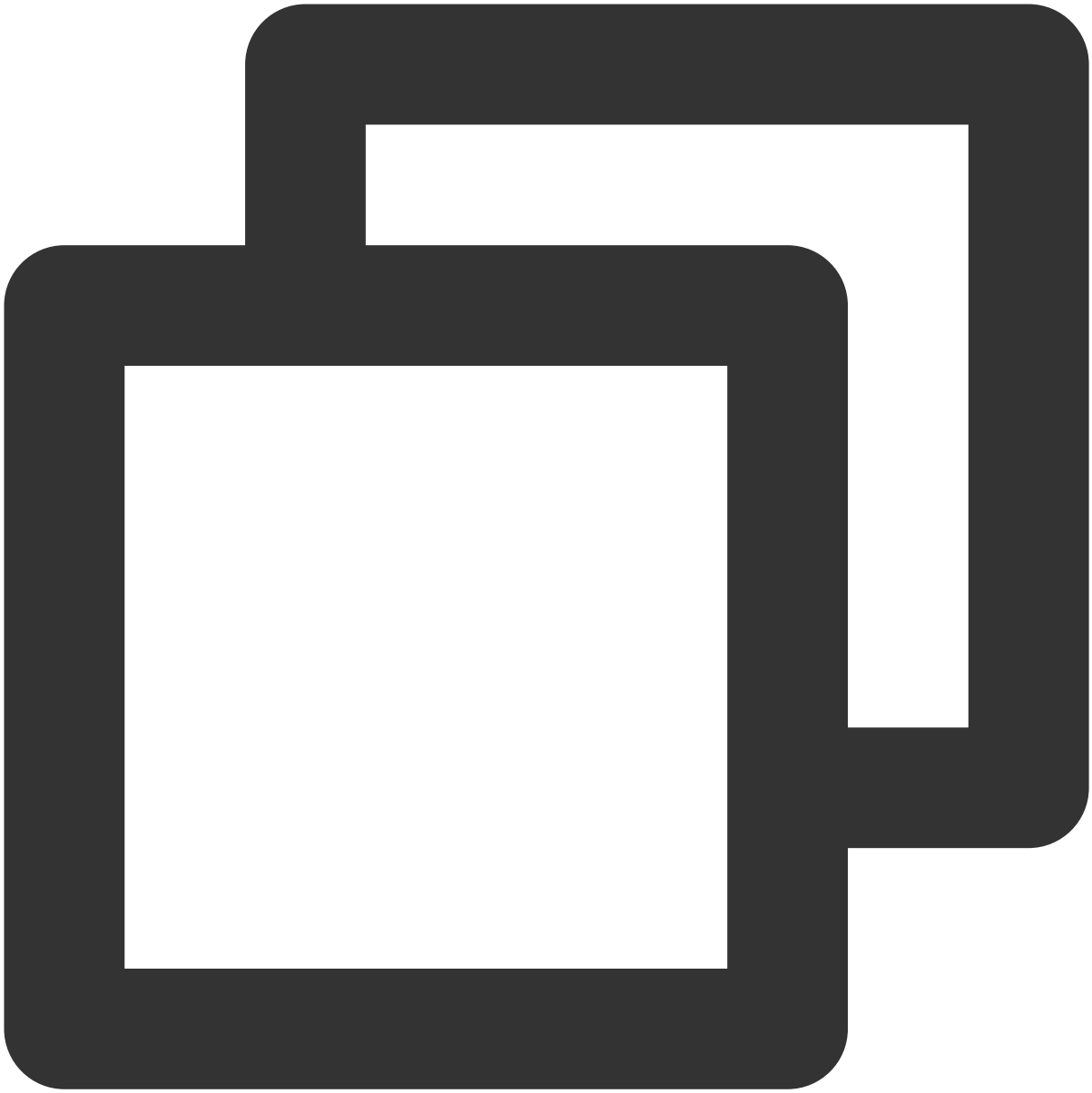
2. **セキュリティ管理>クロスドメインアクセスCORS設定**をクリックし、クロスドメイン設定を1行追加します。デバッグの利便性のため、以下の設定を使用できます。操作ガイドについては、クロスドメインアクセスの設定をご参照ください。

GhostをCOSバケットにバインドする

注意：

サブアカウントキーを使用し、最小権限ガイドに従うことで、使用上のリスクを低減させることをお勧めします。サブアカウントキーの取得については、サブアカウントのアクセスキー管理をご参照ください。

1. Ghostプロジェクトのルートディレクトリ下のconfig.development.json設定ファイルを変更し、次の設定を追加します。

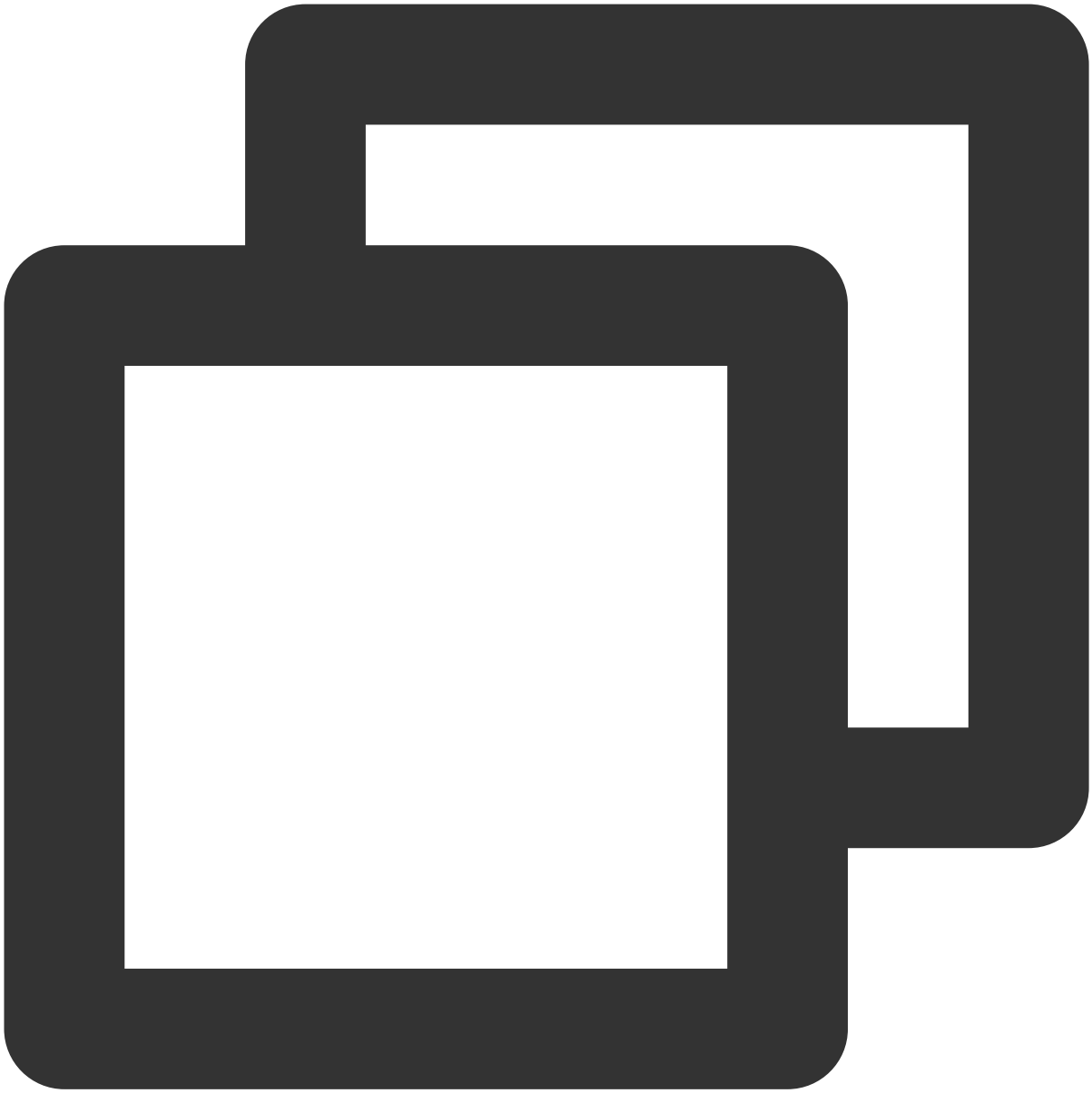


```
 "storage": {
"active": "ghost-cos-store",
"ghost-cos-store": {
  "BasePath": "ghost/", // ご自身のディレクトリ名に変更できます。入力しない場合はデフォルトで
  "SecretId": "AKID*************",
  "SecretKey": "***************",
  "Bucket": "xxx-125********",
  "Region": "**-*******"
}
 }
```

パラメータの説明は次のとおりです。

設定項目	設定値
BasePath	ファイルの保存先のCOSパスはご自身で変更できます。入力しない場合はデフォルトでルートディレクトリになります
SecretId	アクセスキー情報。Tencent Cloud APIキーに進んで作成と取得を行うことができます
SecretKey	アクセスキー情報。Tencent Cloud APIキーに進んで作成と取得を行うことができます。
Bucket	バケット作成時にカスタマイズした名前。例：examplebucket-1250000000。
Region	バケット作成時に選択したリージョン。

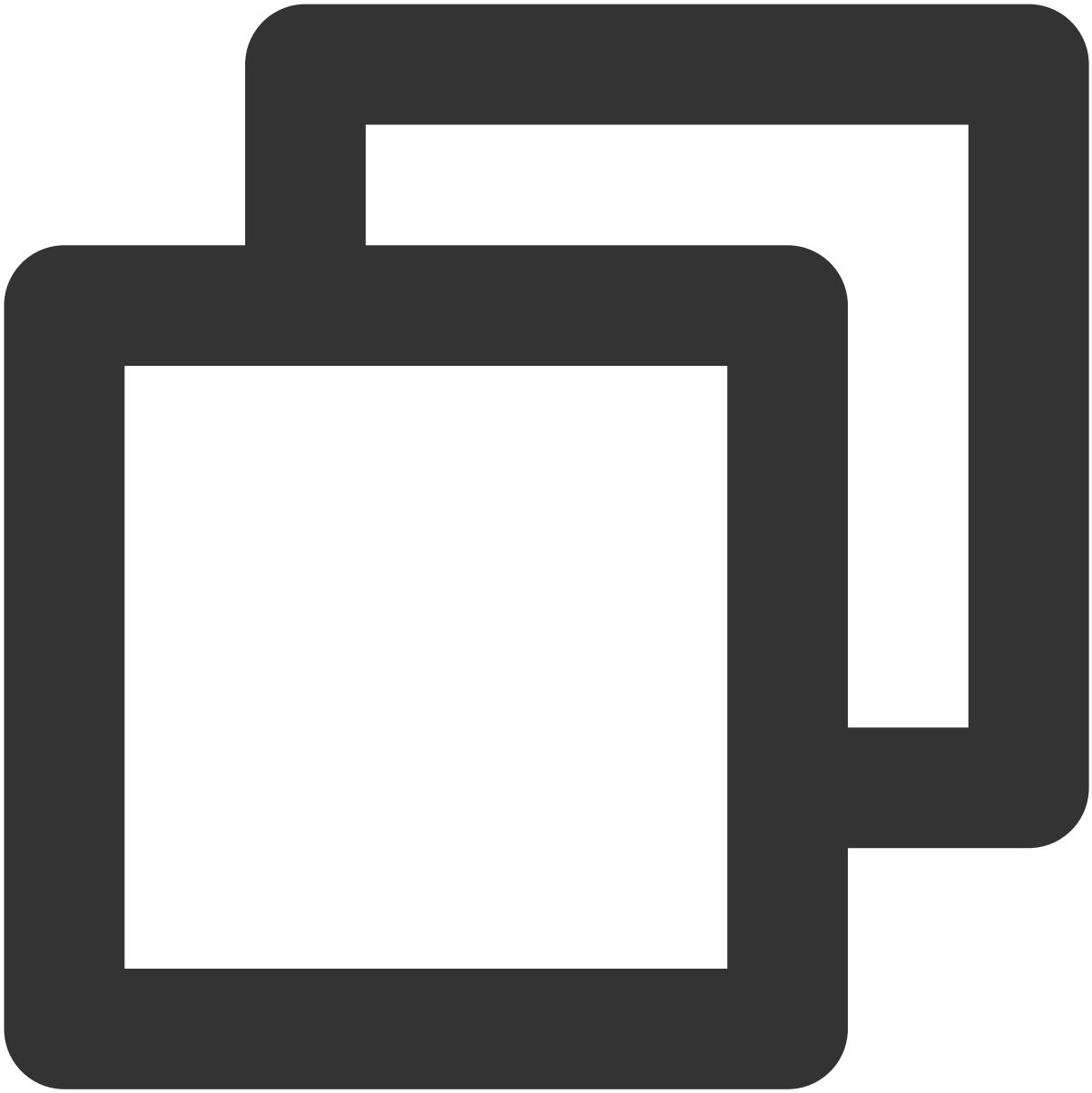
2. カスタムストレージディレクトリを作成し、このプロジェクトのルートディレクトリで次を実行します。



```
mkdir -p content/adapters/storage
```

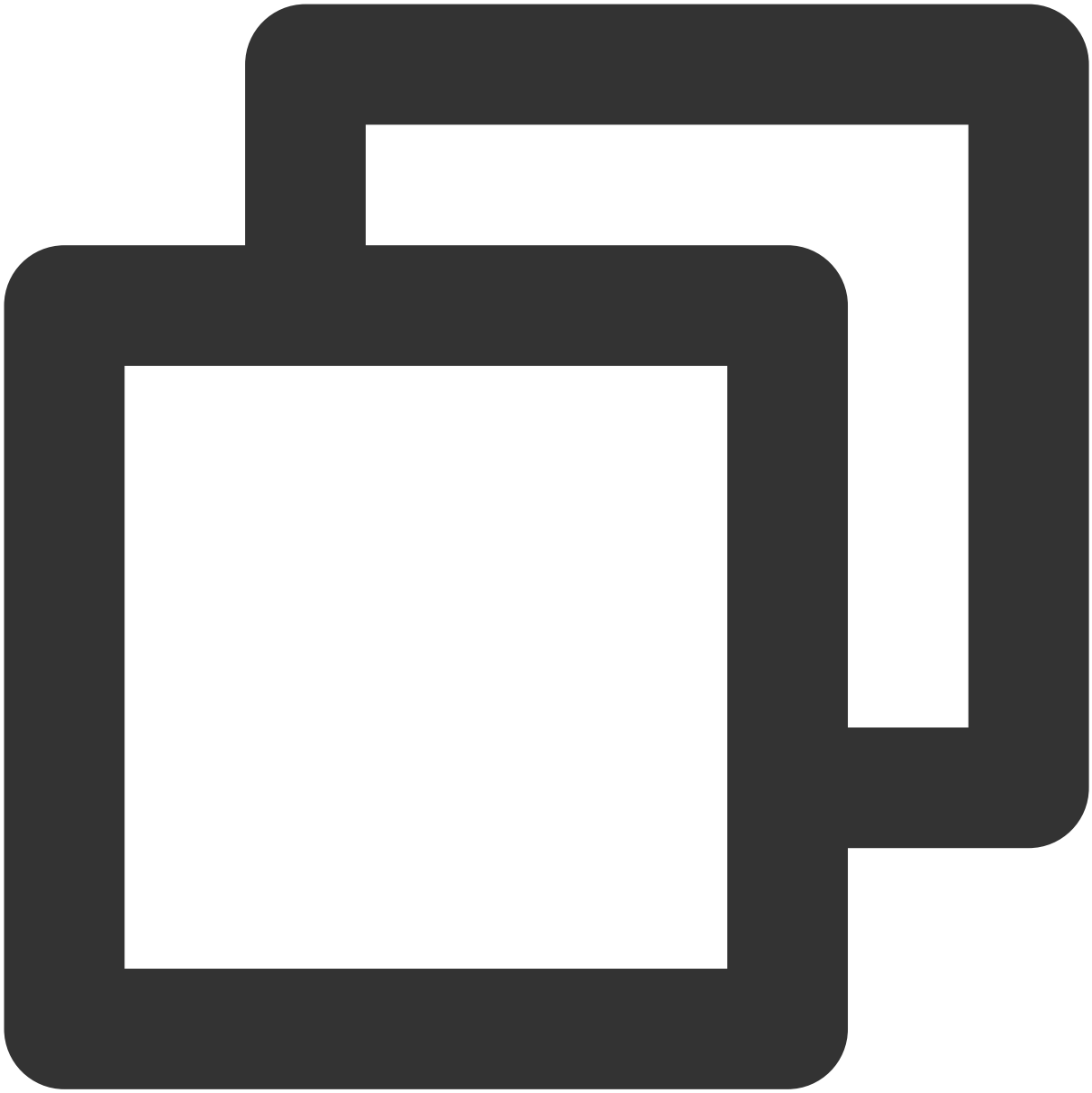
3. Tencent Cloud公式が提供するghost-cos-storeプラグインをインストールします。

3.1 npmによってインストールします。



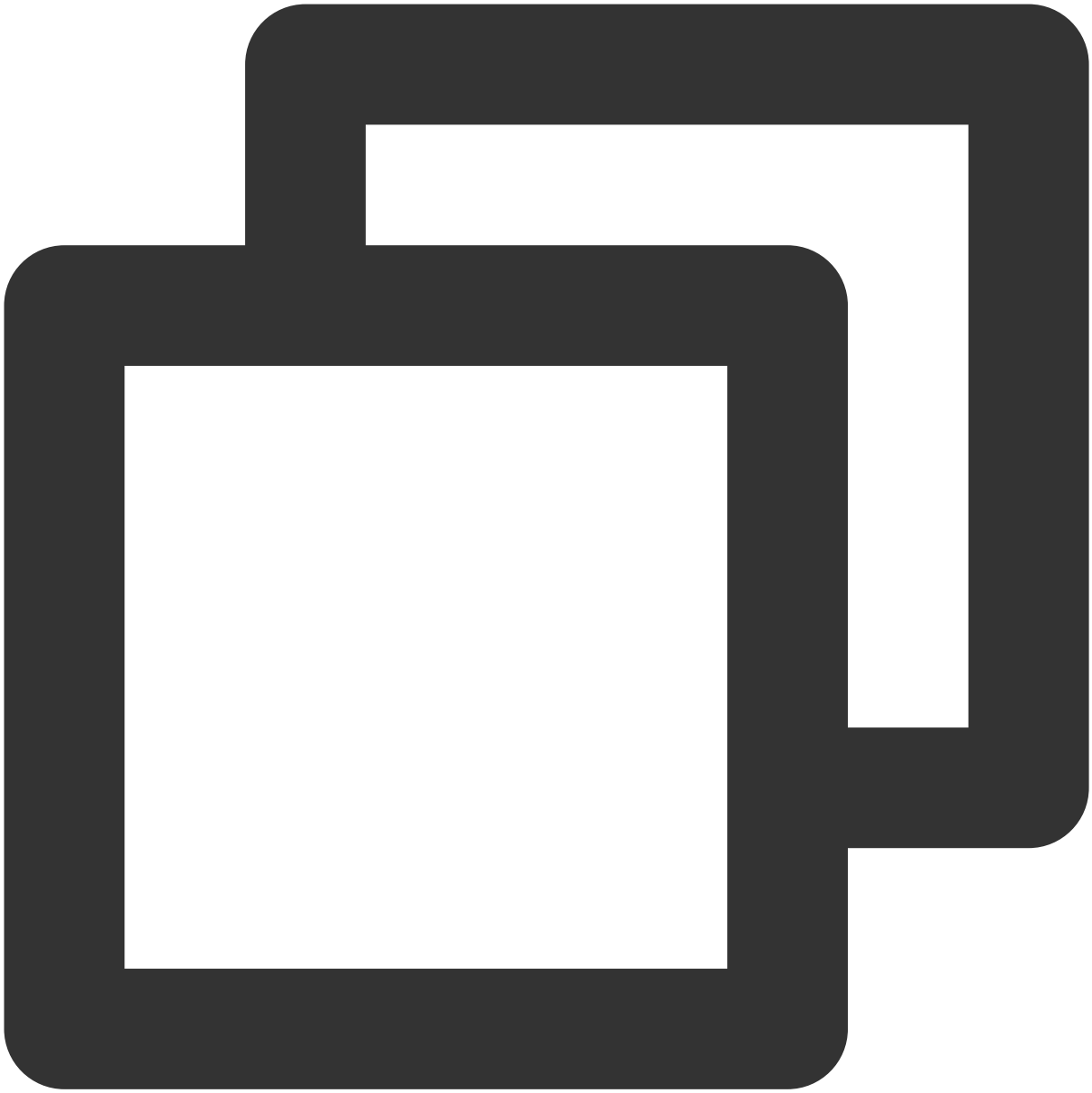
```
npm install ghost-cos-store
```

3.2 storageディレクトリ下にghost-cos-store.jsファイルを作成します。内容は次のとおりです。



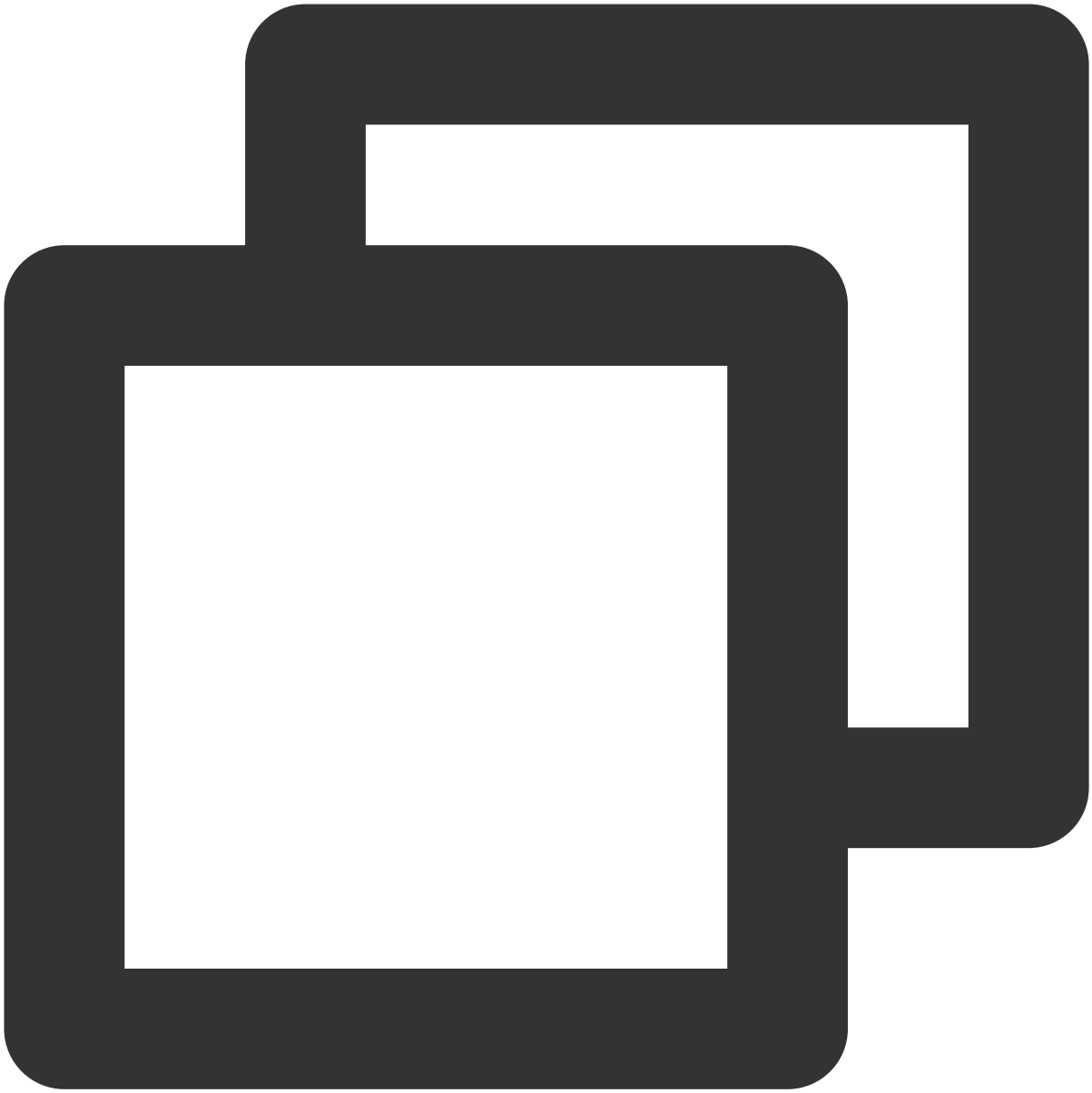
```
//  content/adapters/storage/ghost-cos-store.js
module.exports = require('ghost-cos-store');
```

3.3 git cloneによってインストールします。



```
cd content/adapters/storage
git clone https://github.com/tencentyun/ghost-cos-store.git
cd ghost-cos-store
npm i
```

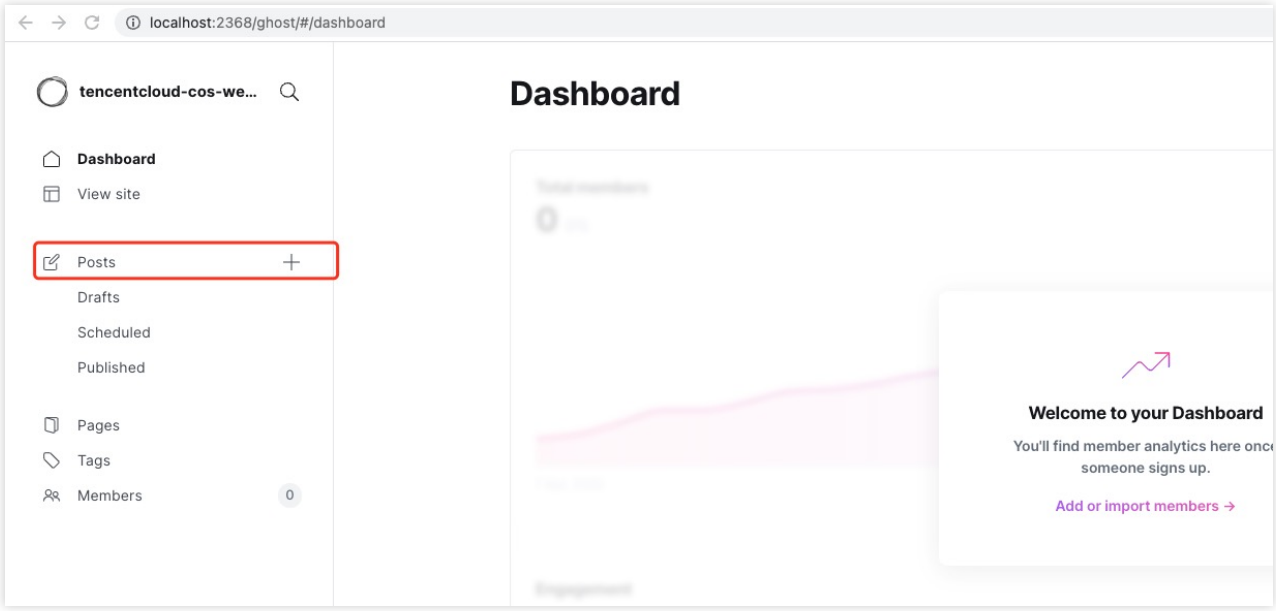
3.4 インストール完了後にGhostの再起動が必要です。



```
ghost restart
```

投稿してアップロードテストを行う

1. Ghost管理バックエンドに進み、クリックして記事を1件投稿します。



2. 画像のアップロードをクリックします。ブラウザでパケットキャプチャを行うとuploadリクエストの成功が確認でき、画像に対応したCOSリンクが返されます。

NextCloud + COSを使用した個人オンラインストレージの構築

最終更新日：：2024-06-26 10:47:37

まえがき

NextCloudはオンラインストレージの作成に用いられるオープンソースクライアントおよびサーバーソフトウェアです。このソフトウェアを利用することで、誰でも個人のオンラインストレージサービスを構築することができます。
NextCloudのサーバーはPHPで作成し、基盤ストレージはデフォルトでサーバーのローカルディスクに保存されます。NextCloudの構成を変更することで、Tencent Cloud Object Storage（COS）を基盤ストレージとして使用することができ、COSのより安価なストレージコスト、より高い信頼性と障害復旧機能、ならびに無限のストレージスペースを利用することが可能になります。
ここではNextCloudのサーバーが依存する環境についてご説明するとともに、ローカルストレージとCOSの違いについて分析比較し、最後に個人オンラインストレージ構築の実践について解説します。
注意：
既存のNextCloudサーバーインスタンスをローカルストレージからTencent Cloud COSの使用に切り替えると、既存のファイルが見えなくなる可能性があります。既存のインスタンスのストレージ方式を変更したい場合は、このチュートリアルに従って新しいNextCloudサーバーを構築し、Tencent Cloud COSを使用するよう設定した後、古いインスタンスのデータを新しいインスタンスに移行することをお勧めします。

NextCloudサーバー環境の概要

NextCloudサーバーはPHPで作成し、データベースはSQLite、MySQL、MariaDB、PostgreSQLを使用できます。このうちSQLiteはパフォーマンス上の制限があるため、通常実際のユースケースで使用することはお勧めしません。PHP、MySQLおよび関連のサーバーソフトウェアにはすべてWindows版がありますが、NextCloudコミュニティからのフィードバックによると、WindowsでNextCloudサーバーを実行すると文字コードなどの問題が生じることがあるため、公式はWindowsでのNextCloudサーバーのデプロイをサポートしていないとアナウンスしています。

サーバーの設定

Tencent Cloud Virtual Machine（CVM）には現在複数のインスタンスファミリーがあり、各インスタンスファミリーはそれぞれ複数のサブタイプに分かれています。インスタンスファミリーごとに特化する点（例えば大容量メモリや高IOなど）が異なります。NextCloudの位置づけは個人、家庭、中小企業ユーザー向けであり、各ハード

ウェアリソースへの要件はどれも高くないため、各リソースのバランスの取れた標準型を選択することでニーズに対応できます。通常、最新のサブタイプはよりコストパフォーマンスが高いため、一般的には最新のサブタイプを選択するとよいでしょう。
インスタンスファミリーとサブタイプを決定した後、具体的なvCPUとメモリ仕様も選択する必要があります。vCPUとメモリ仕様ごとに、対応するプライベートネットワーク帯域幅とネットワーク送受信パケットが異なります。PHPはOPcacheによってパフォーマンスを向上させることができ、NextCloudサーバーもAPCuメモリキャッシュの使用によるパフォーマンス向上をサポートしています。このため、仕様を選択する際は容量が大きめのメモリを選択することをお勧めします。
CVMは購入後に構成の調整が可能なため、例えばvCPU1コアとメモリ4GBなどの比較的低構成の仕様をまず購入し、構築完了後に実際にオンラインで使用する際に、ユーザー数、ファイル数、CVMの関連モニタリングデータなどに基づいて、パフォーマンス向上のための仕様アップグレードが必要かどうかを判断することができます。例えばご家庭または中小企業などの複数ユーザーによる使用を想定している場合は、複数ユーザーでの使用に足る十分なパフォーマンスを提供するため、2コア8GBから4コア16GBの構成の購入をお勧めします。

サーバーOS

主要なLinuxディストリビューションであれば、どれもNextCloudサーバーを問題なく実行できます。ソフトウェアパッケージインストールをする際に使用するコマンド（パッケージ管理ツール）がシステムによって異なる点を除き、設定作業に違いはありません。
説明：
ここではCentOS OS 7.7のCVMを例として説明します。

データベース

上記で述べたとおり、実際のユースケースでは通常MySQLとPHPをセットで使用します。MariaDBはMySQLの「復刻」版であり、MySQLとの間で高度な互換性を有します。このためMySQL 5.7+またはMariaDB 10.2+はどちらも問題なくNextCloudサーバーと組み合わせて使用することができます。
Tencent CloudがホストするTencentDB for MySQLとTencentDB for MariaDBは、CVM上の自作データベースと異なり、デフォルトで1マスター1スレーブの高可用性モードを採用し、より高い信頼性を有するほか、自動バックアップなどの便利な運用保守操作をご提供します。このため、実際のユースケースではこちらのクラウドデータベースのご使用を強く推奨します。
説明：
ここではTencentDB for MySQL 5.7を例として説明します。

WebサーバーとPHPランタイム

NextCloudサーバーは `.htaccess` によって一部の設定を指定するため、Apacheサーバーソフトウェアを使用する際、NextCloudサーバー自体の設定項目をそのまま使用することができます。Nginxは近年急成長中のWebサーバーソフトウェアであり、Apacheに比べてインストール設定が簡単、リソース占有が少ない、ロードバランシング機能がより強力であるなどのメリットがあります。NextCloudサーバー内の `.htaccess` 設定をNginxの設定と

して移行しても、NextCloudサーバーの実行は問題なくサポートされます。ここではNginxサーバーソフトウェアを使用するとともに、完全なNginx設定の例を参考までに示します。
PHPランタイムは現在PHP 7にバージョンアップしています。保守されている主なバージョンは7.2、7.3、7.4であり、この3バージョンはすべてNextCloudサーバーをサポートしています。最新の7.4の使用をお勧めします。また、NextCloudはPHPの一部拡張モジュールにも依存しています。続いて拡張モジュール要件の詳細についてご説明します。

Tencent Cloudネットワーク環境

Tencent Cloudでは現在、基幹ネットワークとVirtual Private Cloud（VPC）環境をご提供しています。基幹ネットワークとはTencent Cloud上のすべてのユーザーのための公共のネットワークリソースプールです。すべてのCVMのプライベートIPアドレスはTencent Cloudが一元的に割り当てており、ネットワークセグメントの区分、IPアドレスのカスタマイズは行えません。VPCとは、ユーザーがTencent Cloud上に構築した、論理的に隔離されているネットワークスペースです。VPC内では、ユーザーはネットワークセグメントの区分、IPアドレスおよびルーティングポリシーを自由に定義できます。現在基幹ネットワークは、リソース不足かつ機能の拡張ができないなどの理由により、新規登録アカウントおよび一部の新規アベイラビリティーゾーンでは今後サポートを行わないことになっています。このため、これ以降の説明はVPCを例として行います。
説明：
VPCに関するより詳しい説明については、Virtual Private Cloud製品概要をご参照ください。

CBSとCOSの比較

CVMでは、Cloud Block Storage（CBS）をCVM内のローカルディスクの形式でOSにマウントします。NextCloudはデフォルトでファイルシステムを使用してオンラインストレージのデータを保存するため、NextCloudのデータはそのままOS内のCBSに保存することができます。CBSと比較した、COSを使用する上でのメリットについて、次のいくつかの点から解説します。

ユースケース

CBS

CBSはブロックストレージの一種であり、CVMのOSに直接マウントしてハードディスクとして使用することができます。通常はOSによって専有されるため、1台のCVMにしかマウントできませんが、読み取り書き込みパフォーマンスが高いため、高IO、低遅延かつ他のCVMとの共有が必要ないケースに適しています。

COS

COSはHTTPプロトコルによって外部に提供される読み取り書き込みインターフェースであり、プログラミング方式でCOSのストレージオブジェクト（ファイル）にアクセスする必要があります。COSはオブジェクトキー（Key、ファイルパスであると理解できます）をインデックスとして使用するもので、ストレージ容量の制限がありません。ネットワーク伝送を使用するため、速度と遅延は比較的大きくなりますが、操作がオブジェクトレベ

ルのため、1つのソフトウェアで1つのオブジェクトの操作を行った後、別のソフトウェアですぐに同じオブジェクトの操作を行えることから、パフォーマンスに対する要件が厳しくなく、低コストで大容量のストレージが必要な場合、または共有アクセスが必要なケースに適しています。オンラインストレージの使用自体はネットワーク伝送を通じて行われるため、遅延に対する要件は厳しくなく、かつオンラインストレージクライアントからオンラインストレージサーバーを経由してCOSに至るリンクにおいて、速度と遅延に影響する要因は主にクライアントのネットワーク環境にあり、COS自体の速度制限ではないことから、COSの方がよりオンラインストレージとの併用に適していると言えます。

メンテナンス

CBS

CBSは容量が固定であり、コンソールまたはTencent Cloud APIによるスケーリングが必要です。スケーリング後はOSでパーティションの拡張も行わなければならず、かつパーティションの拡張時にパーティションエラーが発生するリスクも一定程度あり、ある程度のメンテナンスコストがかかります。

COS

COSは必要に応じて使用でき、総容量の制限がなく、オブジェクト数（ファイル数）の制限もありません。完全メンテナンスフリーです。

データセキュリティ

CBSとCOSはどちらもマルチレプリカなどの手段を使用してデータの信頼性を保証しています。

NextCloudサーバー実行環境の構築

NextCloudサーバーが依存するクラウド製品の準備

CVM

スタート操作については、Cloud Virtual Machine（CVM）クイックスタートをご参照ください。

TencentDB for MySQL

スタート操作については、TencentDB for MySQLクイックスタートをご参照ください。

COS

1. COSコンソールを開いてログインし（初回使用前にCOSサービスをアクティブ化する必要があります）、**バケットリスト**に進み、**バケットの作成**をクリックし、下表の説明に従って設定します。

設定項目	値

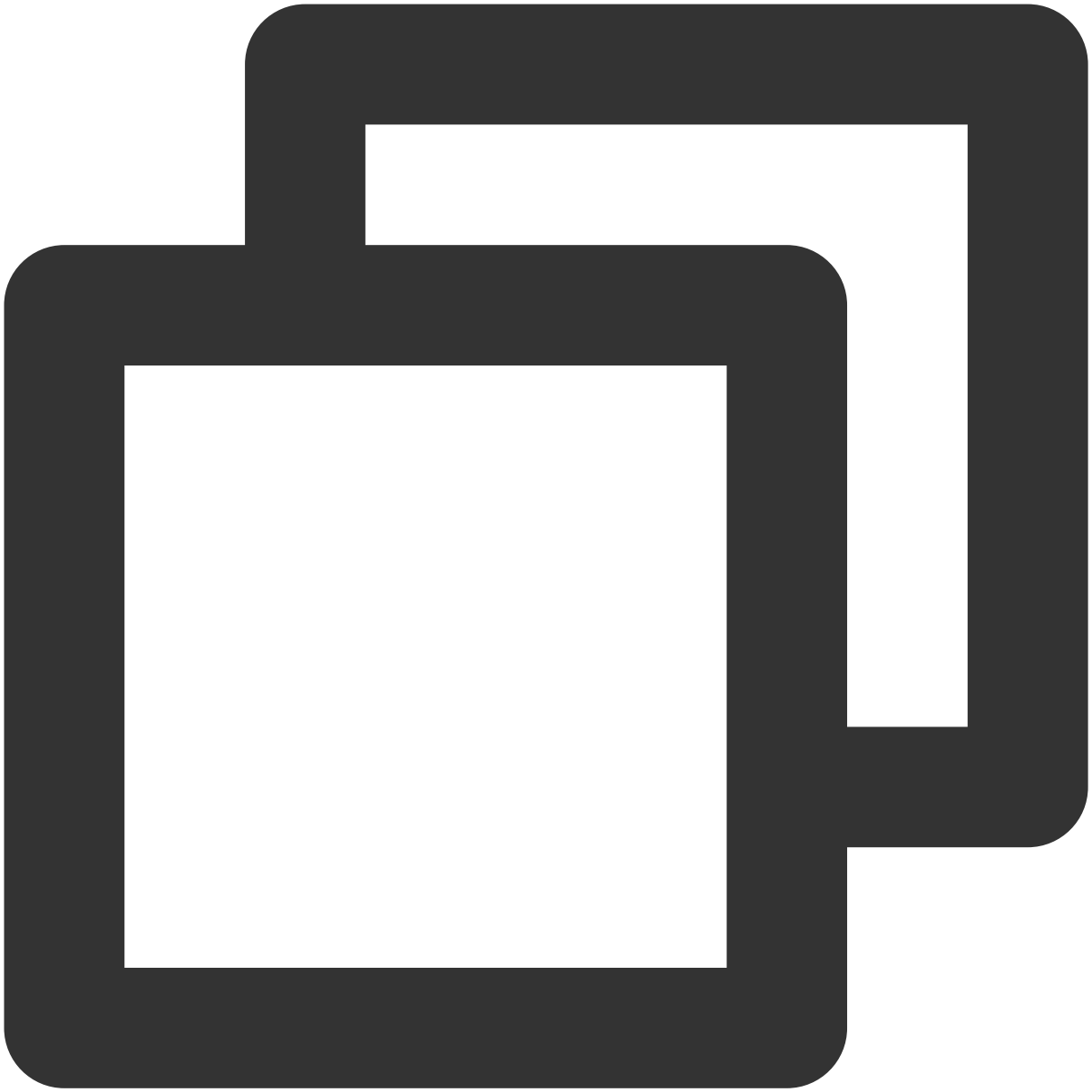
名前	カスタマイズしたバケット名（例：nextcloud）を入力します。この名前は決定すると変更できませんのでご注意ください
所属リージョン	購入したCVMの所属リージョンと同じものにします
その他	デフォルトを維持します

2. 上記の設定完了後、**OK**をクリックして作成を完了します。

Webサーバーおよびランタイムのインストールと設定

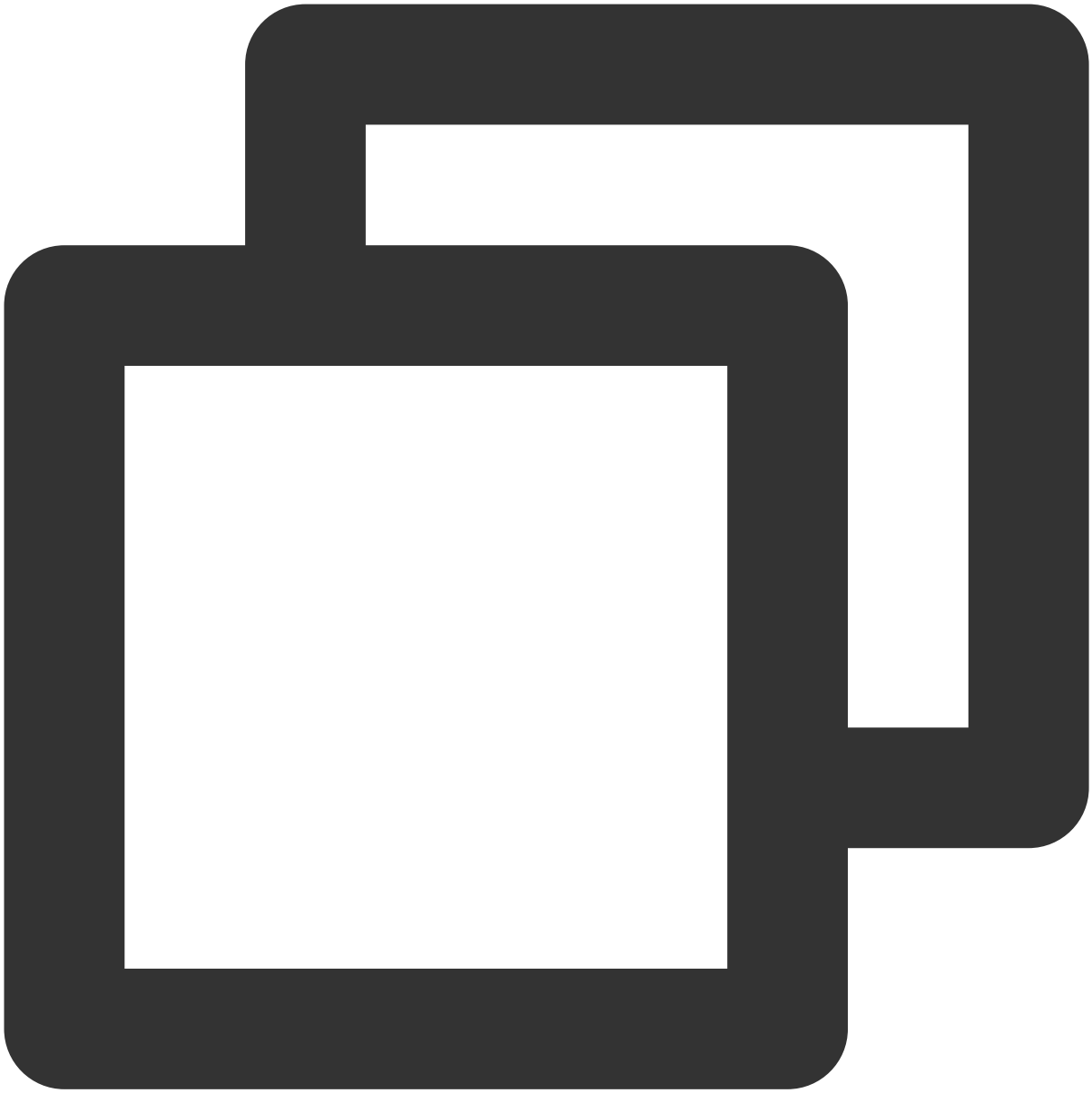
Nginxのインストール

1. SSHツールを使用して新規購入サーバーにログインします。
2. 次のコマンドを実行してNginxをインストールします。



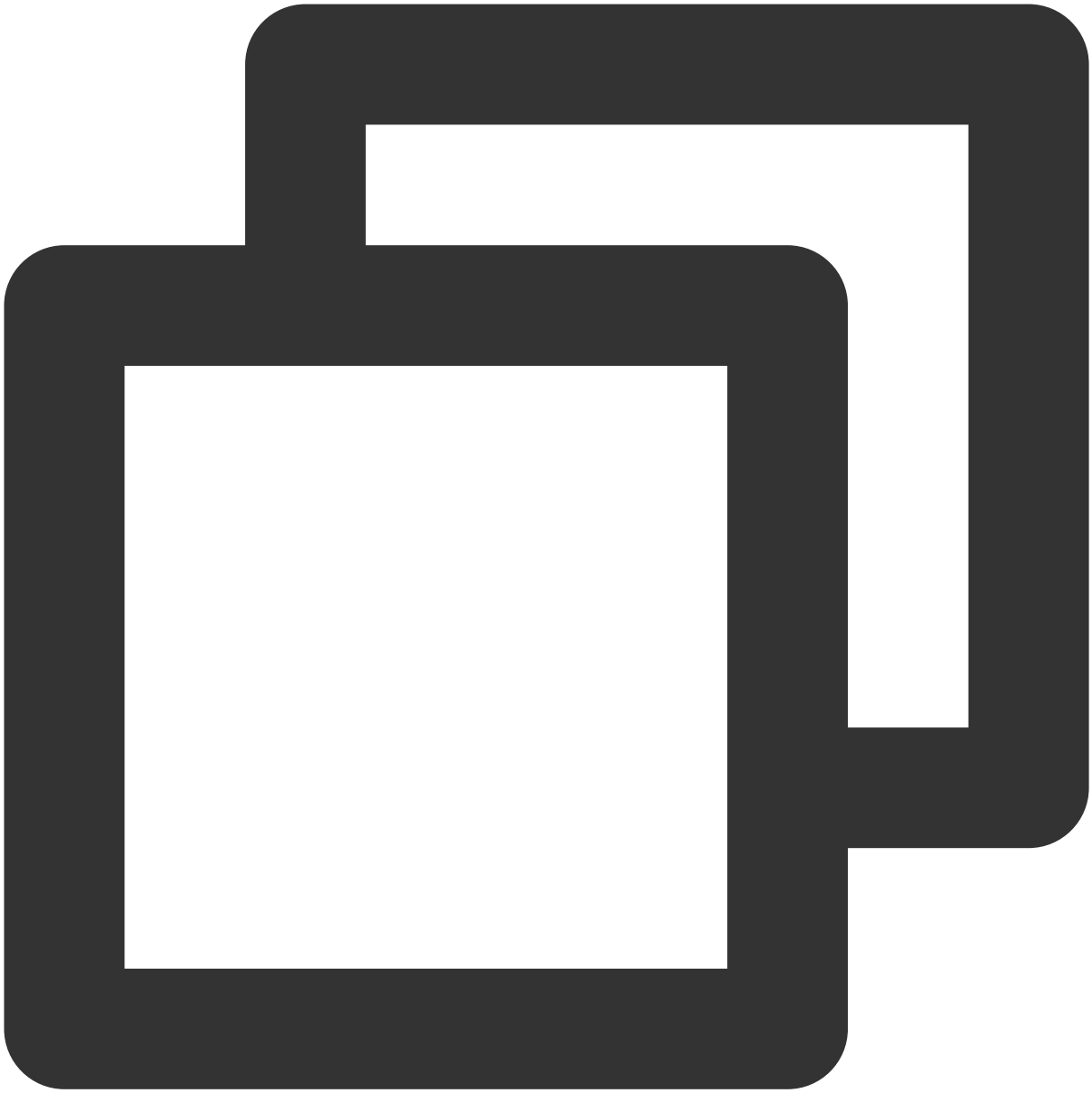
```
yum install nginx
```

次のメッセージが表示された場合は、 `Y` を入力してエンターを押し、インストールを確認します（以下同様）。



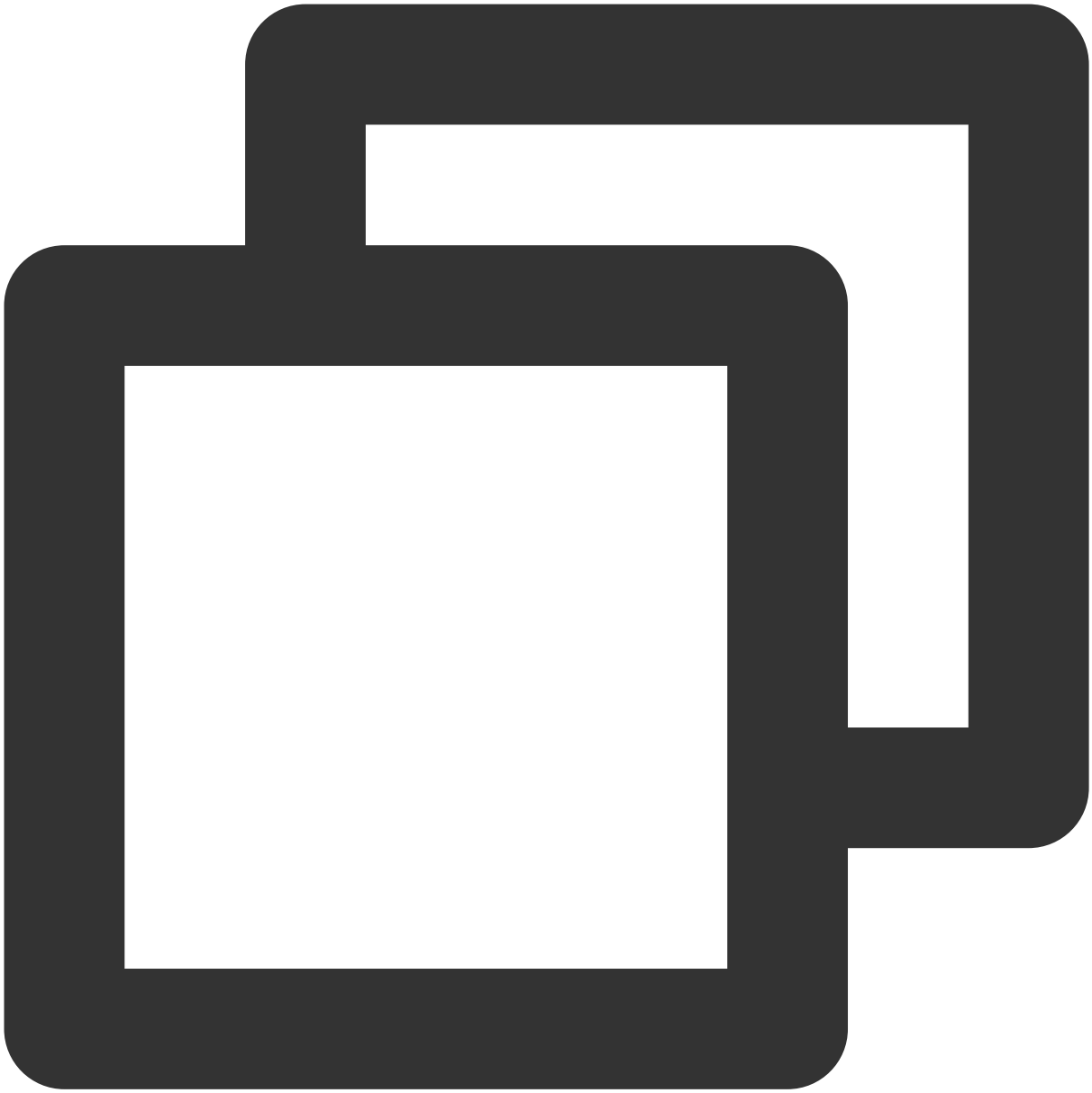
```
Is this ok [y/d/N]:
```

3. 次のメッセージが表示された場合、インストールは完了しています。



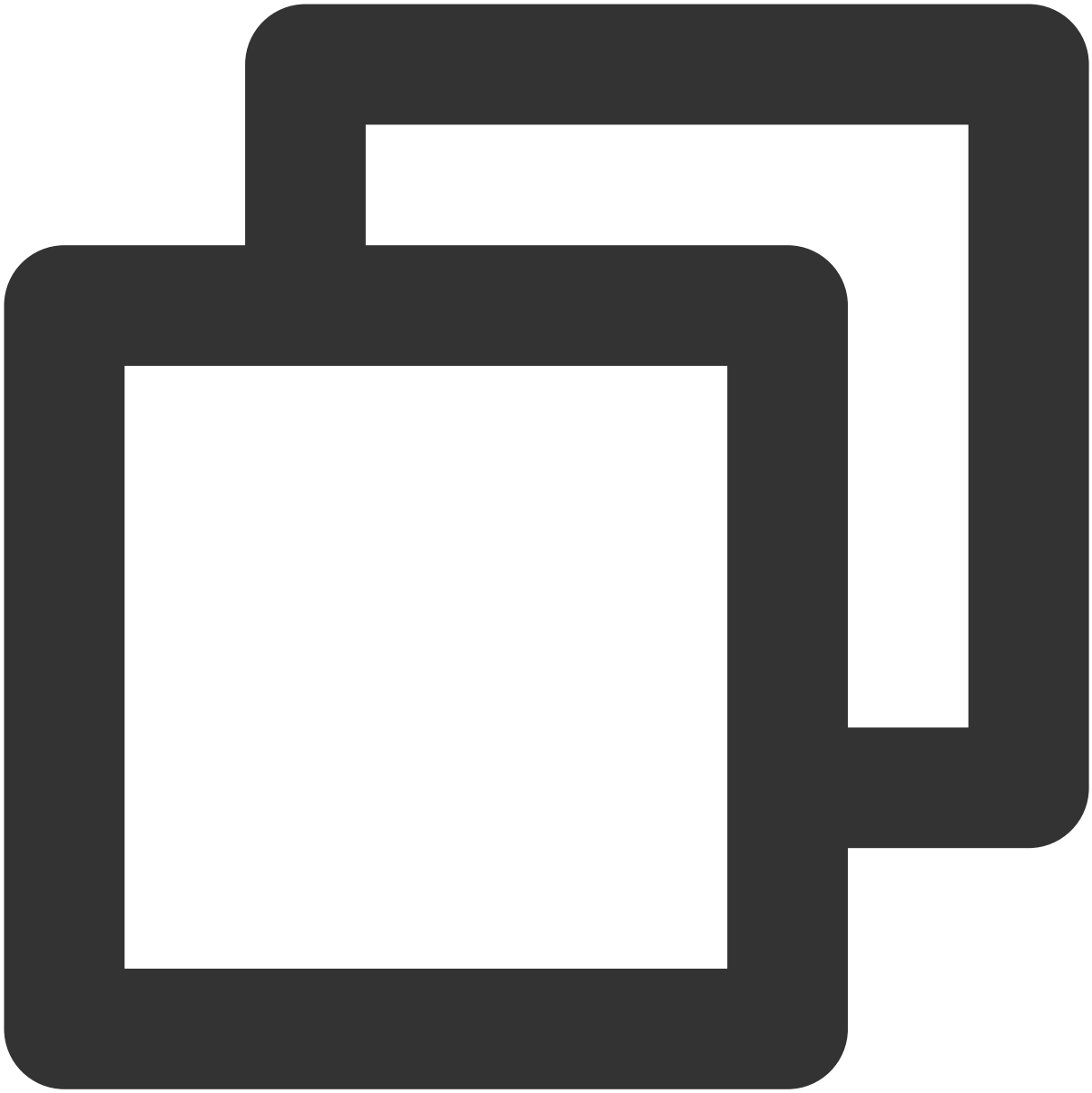
```
Complete!
[root@VM-0-10-centos ~]#
```

4. 続いて次のコマンドを実行し、Nginxのバージョンが正常に表示されるかどうかを検証します。



```
nginx -v
```

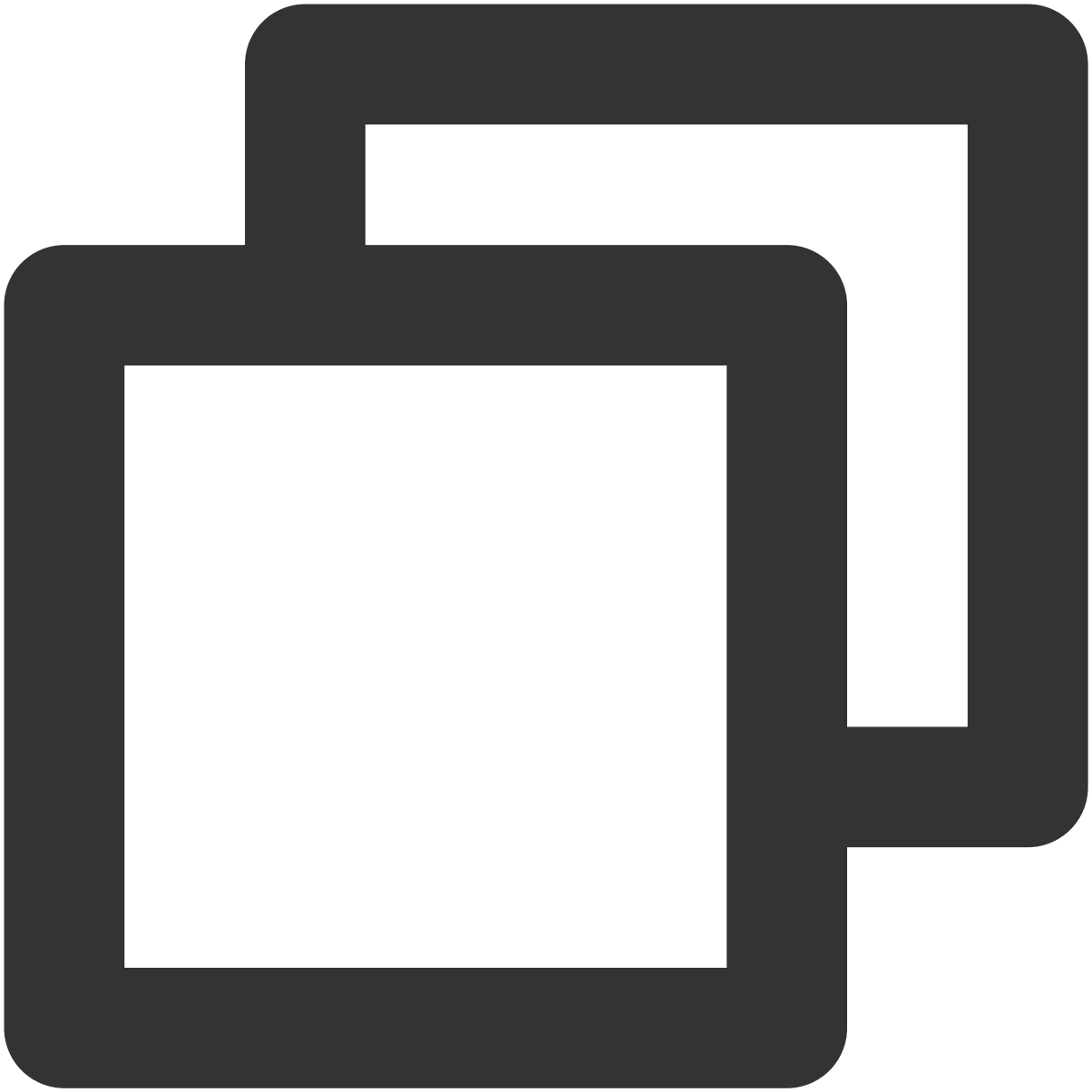
次のメッセージが表示された場合、インストールの完了が検証されました。



```
nginx version: nginx/1.16.1
```

PHPのインストール

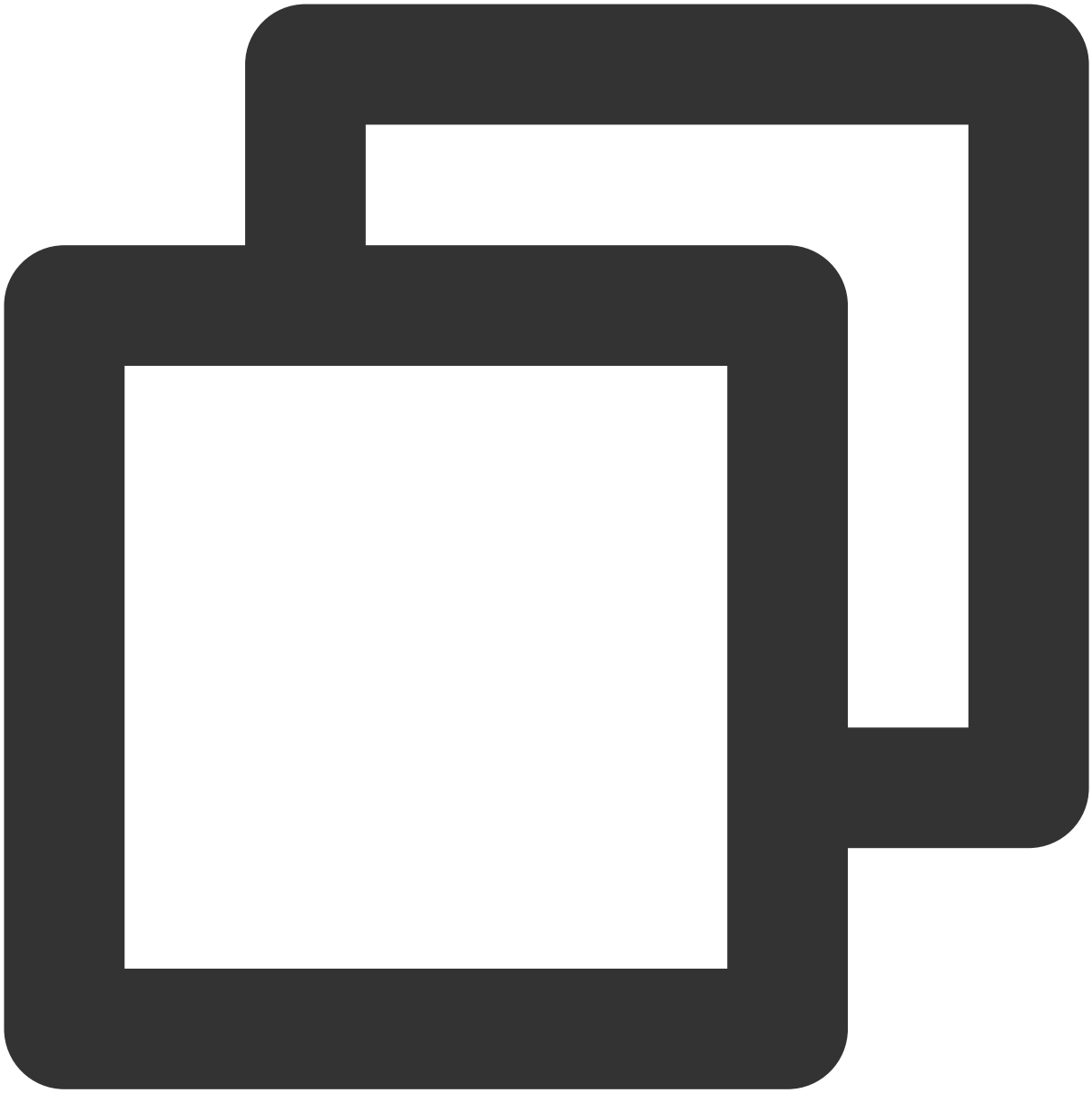
1. SSHツールを使用して新規購入サーバーにログインします。
2. 次のコマンドを実行し、PHP 7.4のインストールを開始します。



```
yum install epel-release yum-utils
```

3. 次のコマンドを順に実行します。

コマンド1：

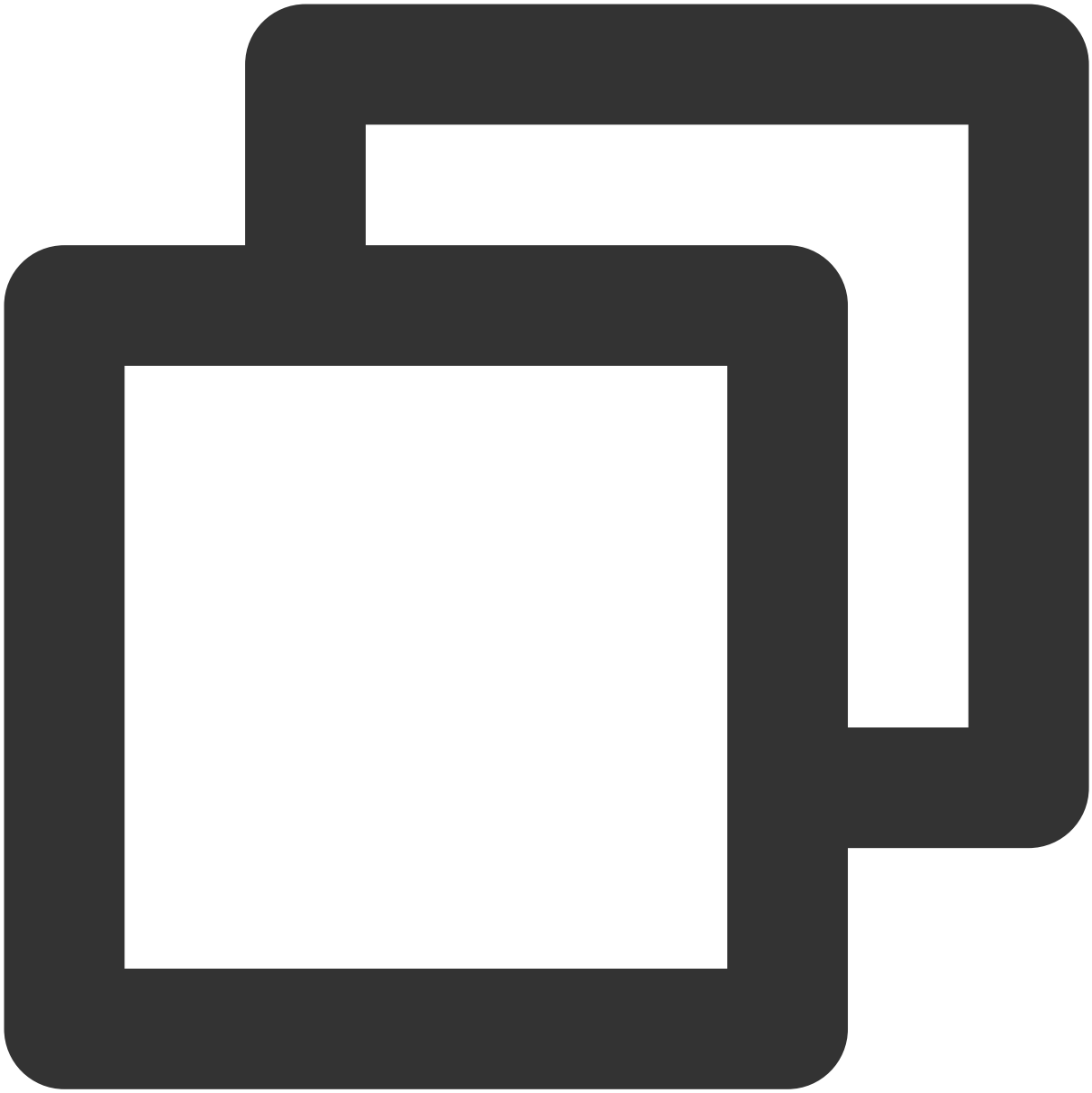


```
yum install http://rpms.remirepo.net/enterprise/remi-release-7.rpm
```

説明：

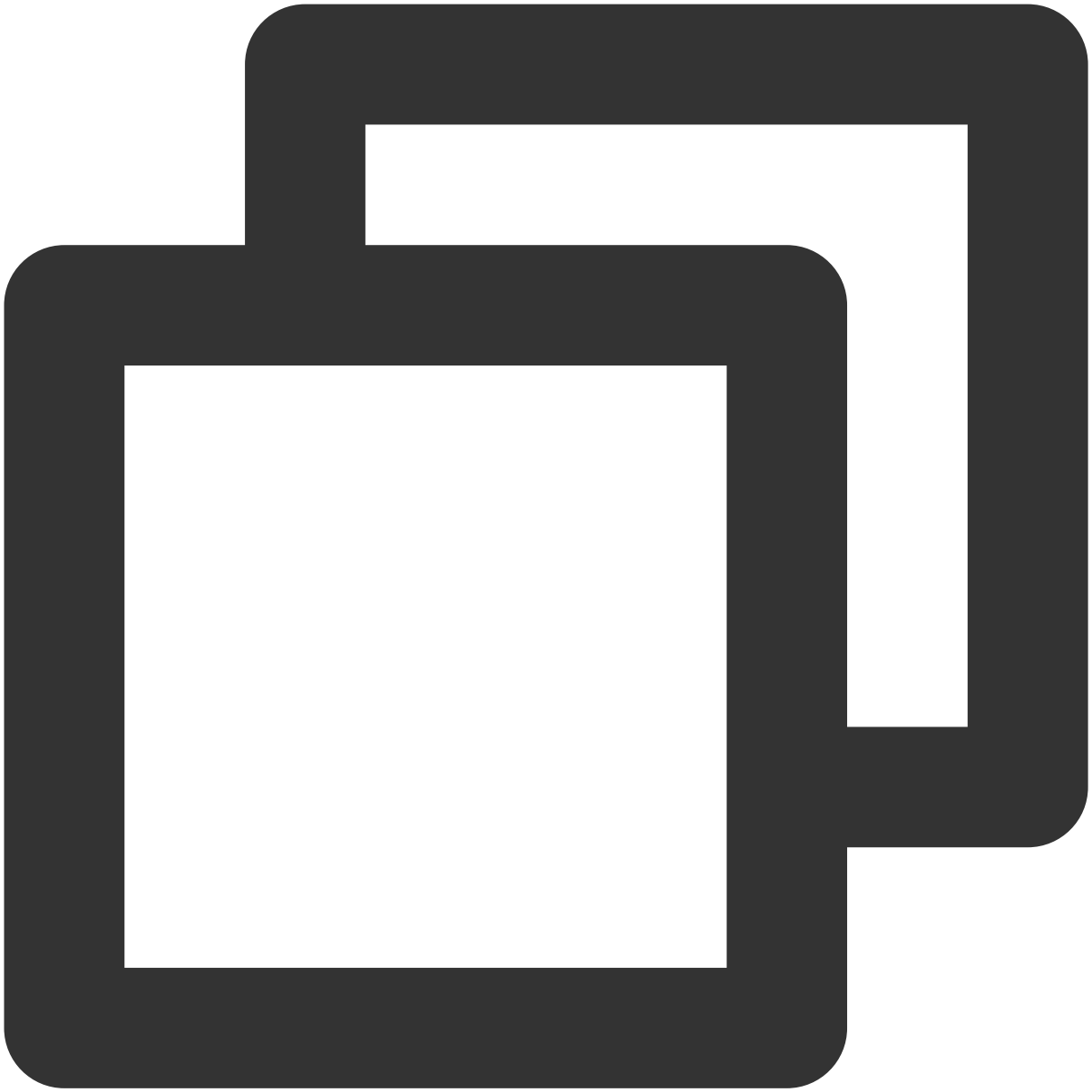
上記のコマンドの実行時に、速度が遅すぎる、長時間進捗しないなどの問題があった場合は、`Ctrl-C` でキャンセルし、再度このコマンドを実行することができます（以下同様）。

コマンド2：



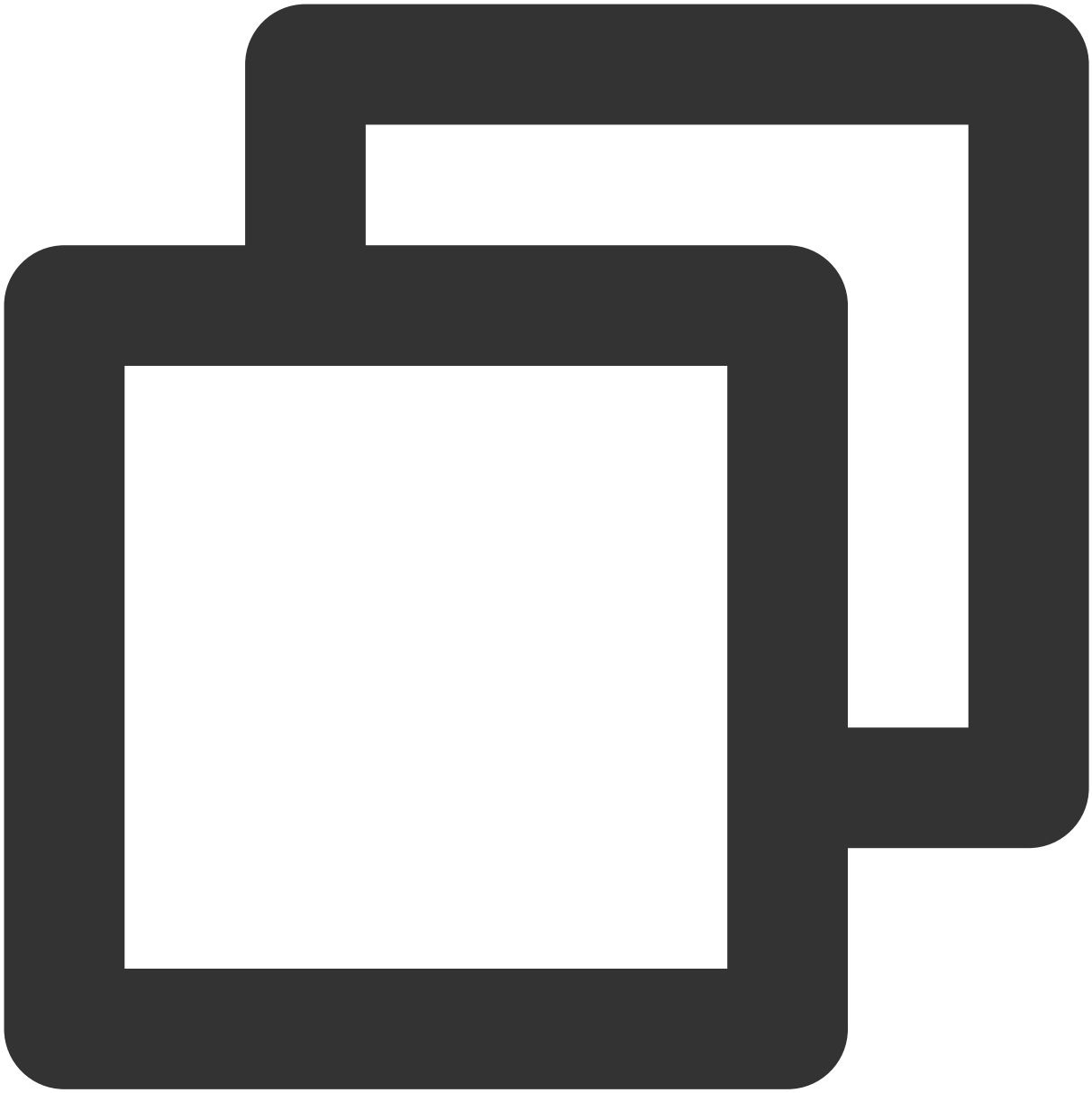
```
yum-config-manager --enable remi-php74
```

コマンド3：



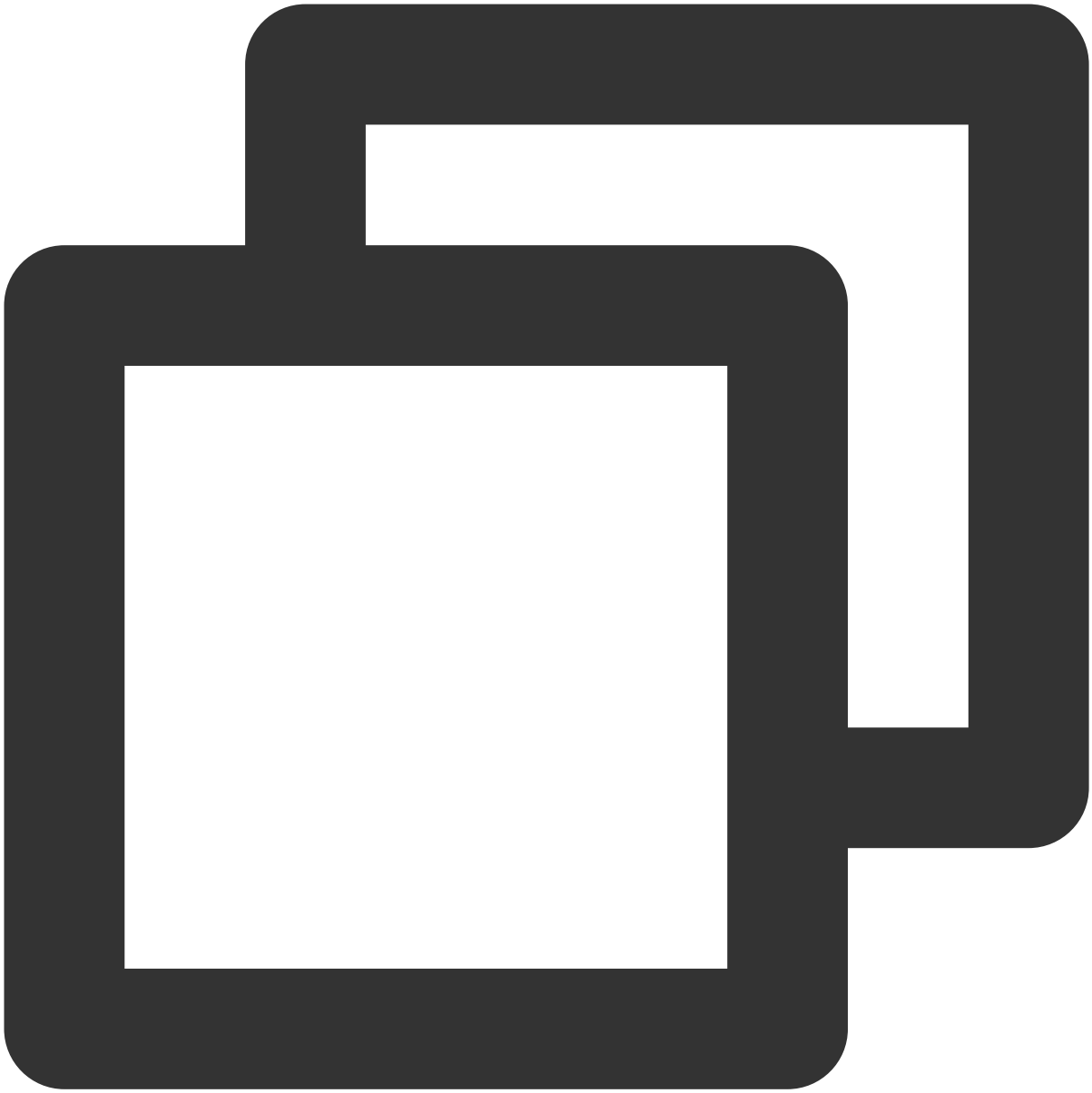
```
yum install php php-fpm
```

4. インストールの完了後に次のコマンドを実行し、PHPのバージョンが正常に表示されるかどうかを検証します。



```
php -v
```

次のメッセージが表示された場合、インストールの完了が検証されました。



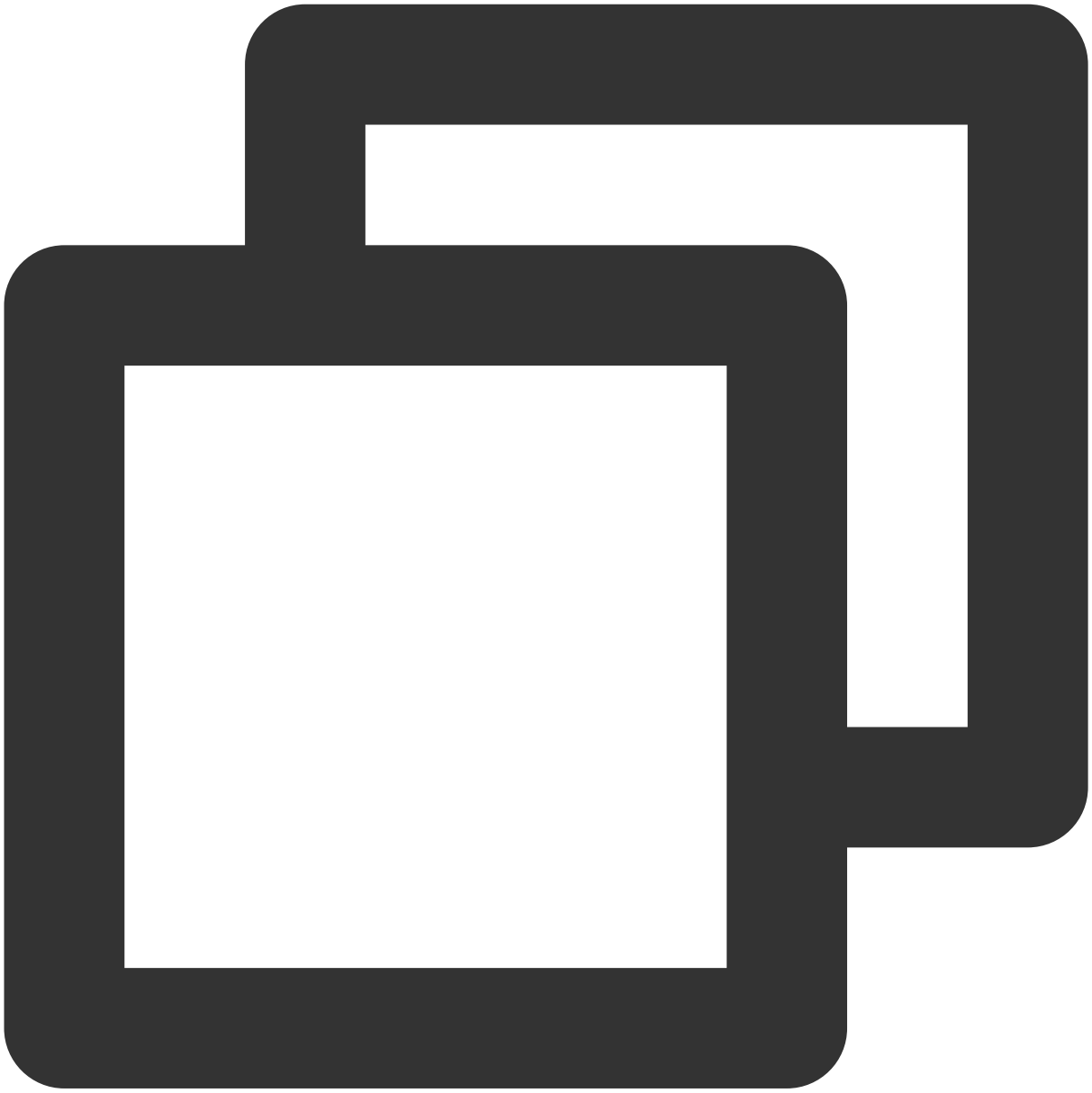
```
PHP 7.4.8 (cli) (built: Jul 9 2020 08:57:23) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
```

PHPモジュールのインストール

基本のPHP以外に、NextCloudは他のPHPモジュールにも依存して一部の機能を実装しています。NextCloudの依存するモジュールについての詳細情報は、NextCloud公式ドキュメントをご参照ください。
このチュートリアルではNextCloudに必須のPHPモジュールをインストールします。その後、NextCloudのその他のオプション機能を使用するご予定がある場合は、ご注意の上、依存する他のPHPモジュールをご自身でインス

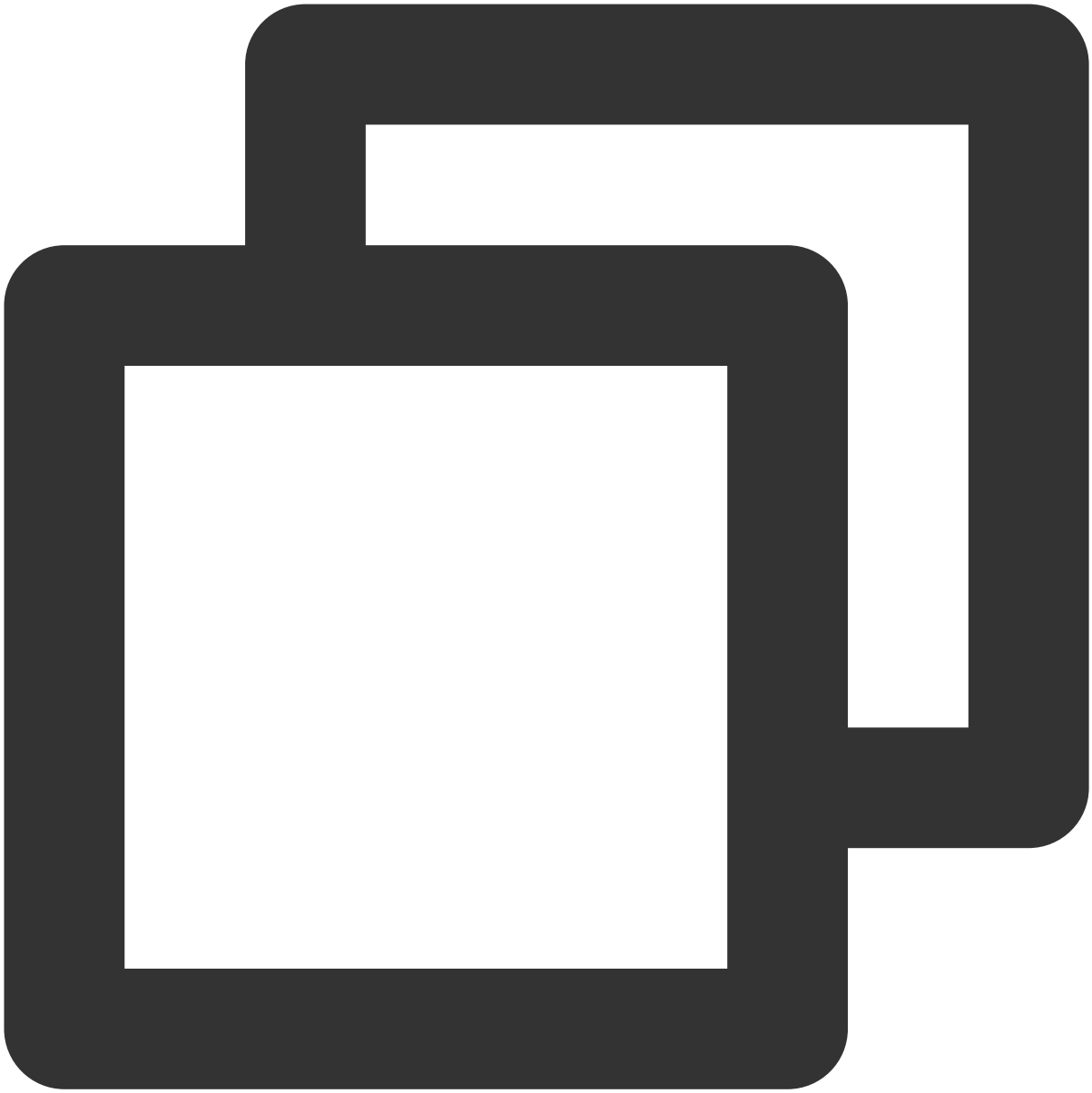
トールしてください。

1. SSHツールを使用して新規購入サーバーにログインします。
2. 次のコマンドを実行し、PHPモジュールをインストールします。



```
yum install php-xml php-gd php-mbstring php-mysqlnd php-intl php-zip
```

3. インストールの完了後、次のコマンドを実行して、インストール済みのPHPモジュールを表示します。



```
php -m
```

4. 他のモジュールもインストールする必要がある場合は、`yum install <php-module-name>` を繰り返し実行します。

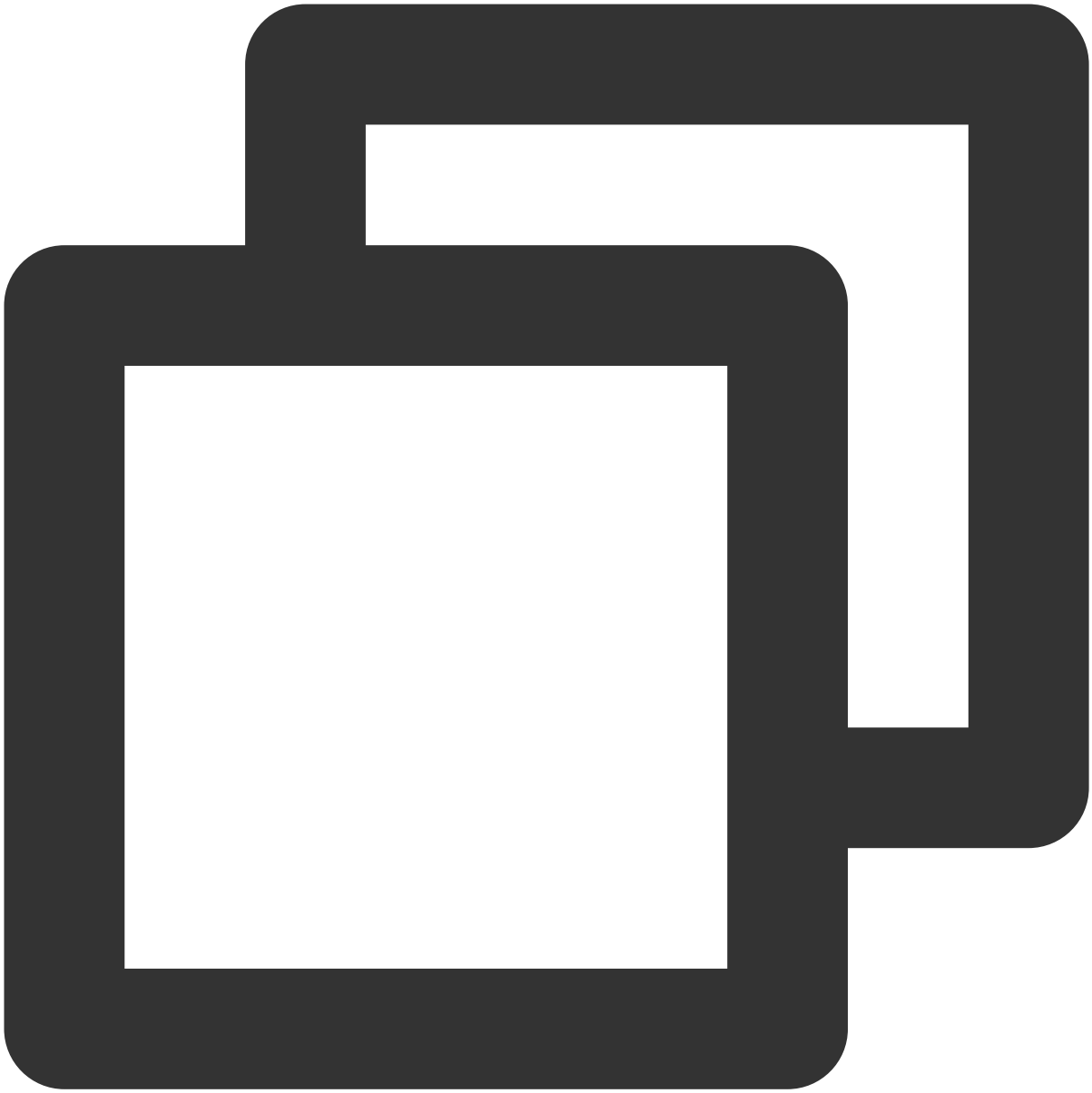
NextCloudサーバーコードのアップロードと解凍

1. NextCloud公式サイトからNextCloudサーバーの最新版のインストールパッケージをダウンロードし、サーバーの `/var/www/` ディレクトリ下にアップロードします。アップロードは以下の方法で行うことができます。

1. `wget` コマンドを使用して、インストールパッケージのダウンロードとサーバーへのアップロードを直接行います。例えば `/var/www/` ディレクトリに進んだ後、コマンド `wget https://download.nextcloud.com/server/releases/nextcloud-19.0.1.zip` を実行します。
2. ローカルコンピュータにダウンロードした後、SFTPまたはSCPなどのソフトウェアによって、インストールパッケージを `/var/www/` ディレクトリにアップロードします。
3. ローカルコンピュータにダウンロードした後、lrzszを使用してアップロードします。方法は次のとおりです。
3.1 SSHツールを使用して新規購入サーバーにログインします。
3.2 `yum install lrzsz` を実行してlrzszをインストールします。
3.3 `cd /var/www/` を実行して目的のディレクトリに進みます。
3.4 `rz -bye` を実行し、続いてSSHツールから、ローカルにダウンロードするNextCloudサーバーインストールパッケージを選択します（この操作はSSHツールによって多少異なります）。
4. SSHツールを使用して新規購入サーバーにログインします。
5. `unzip nextcloud-<version>.zip` を実行してインストールパッケージを解凍します。例えば `unzip nextcloud-19.0.1.zip` などとします。

PHPの設定

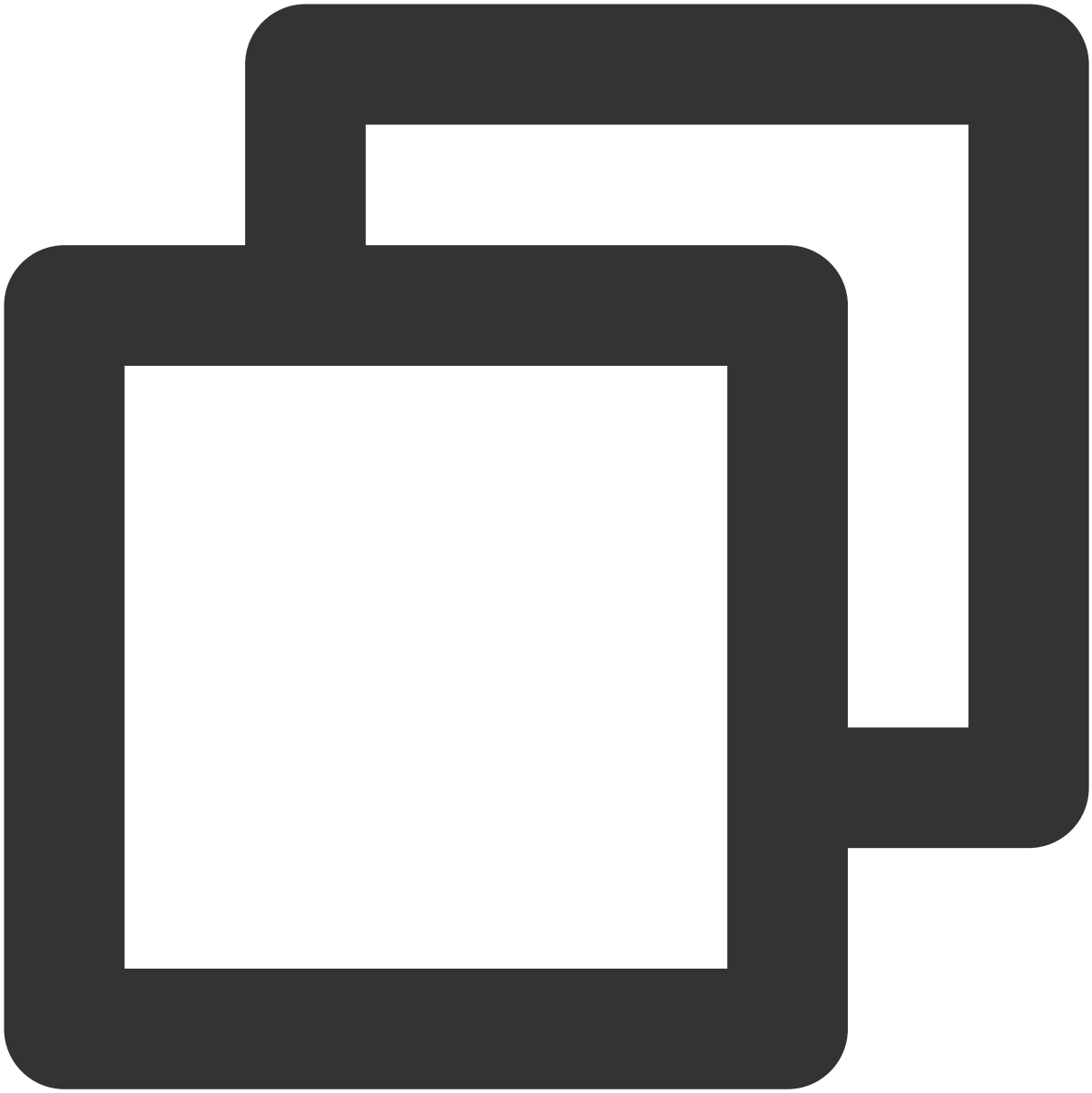
1. SSHツールを使用して新規購入サーバーにログインします。
2. `vim /etc/php-fpm.d/www.conf` を実行してPHP-FPMの設定ファイルを開き、設定項目を順に変更します（vimの具体的な使用方法については関連資料をご参照ください。この設定ファイルの変更は他の方法でも行うことができます）。
1. `user = apache` を `user = nginx` に変更します。
2. `group = apache` を `group = nginx` に変更します。
3. 変更完了後、 `:wq` を入力してファイルを保存して終了します（vimの詳細な操作ガイドについては関連ドキュメントをご参照ください）。
4. 次のコマンドを実行してディレクトリ所有者を変更し、PHPをNginxで使用できるようにします。



```
chown -R nginx:nginx /var/lib/php
```

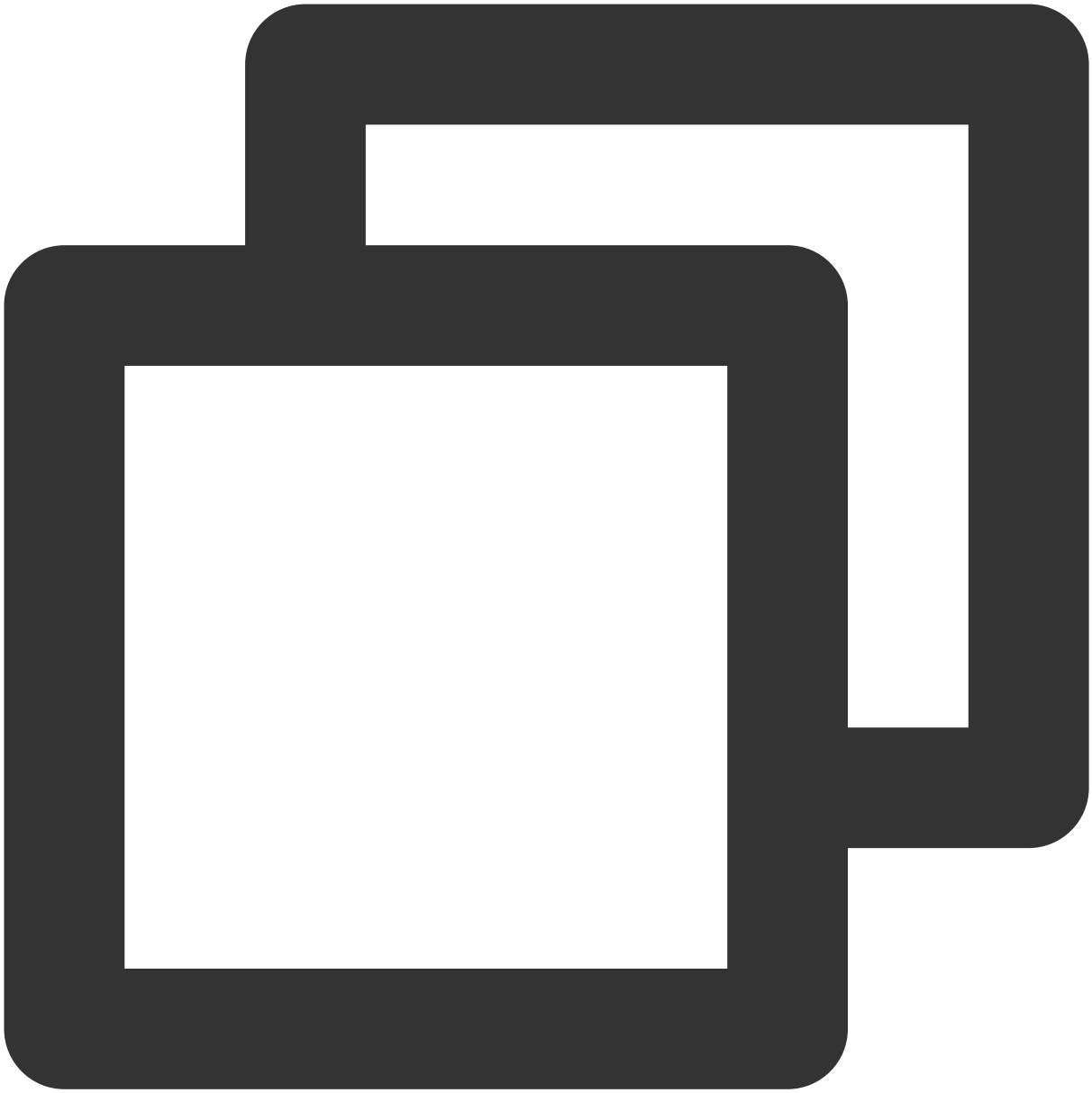
5. 次のコマンドを順に実行し、PHP-FPMサービスを起動します。

コマンド1：



```
systemctl enable php-fpm   # コマンド1
```

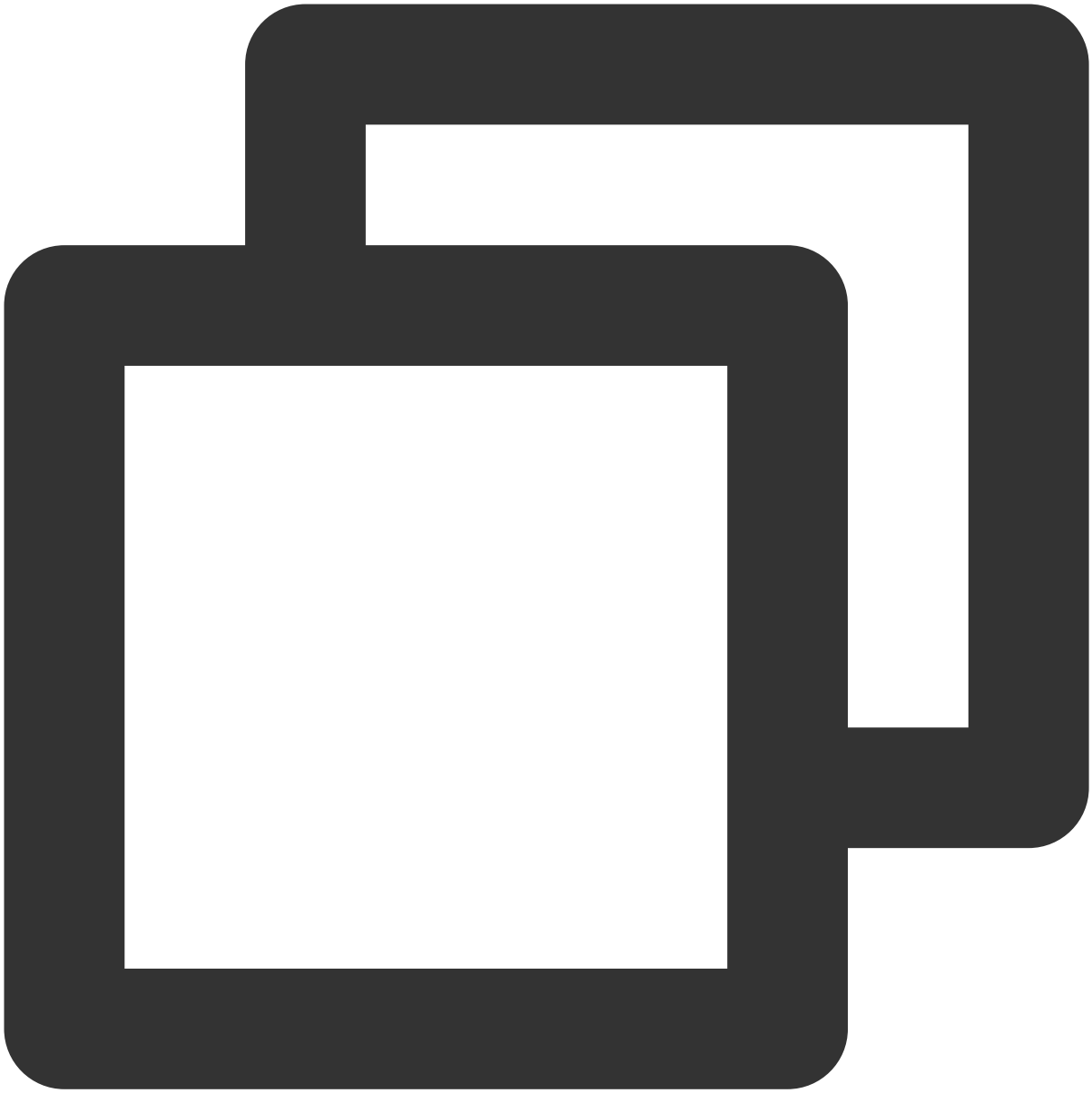
コマンド2：



```
systemctl start php-fpm   # コマンド2
```

Nginxの設定

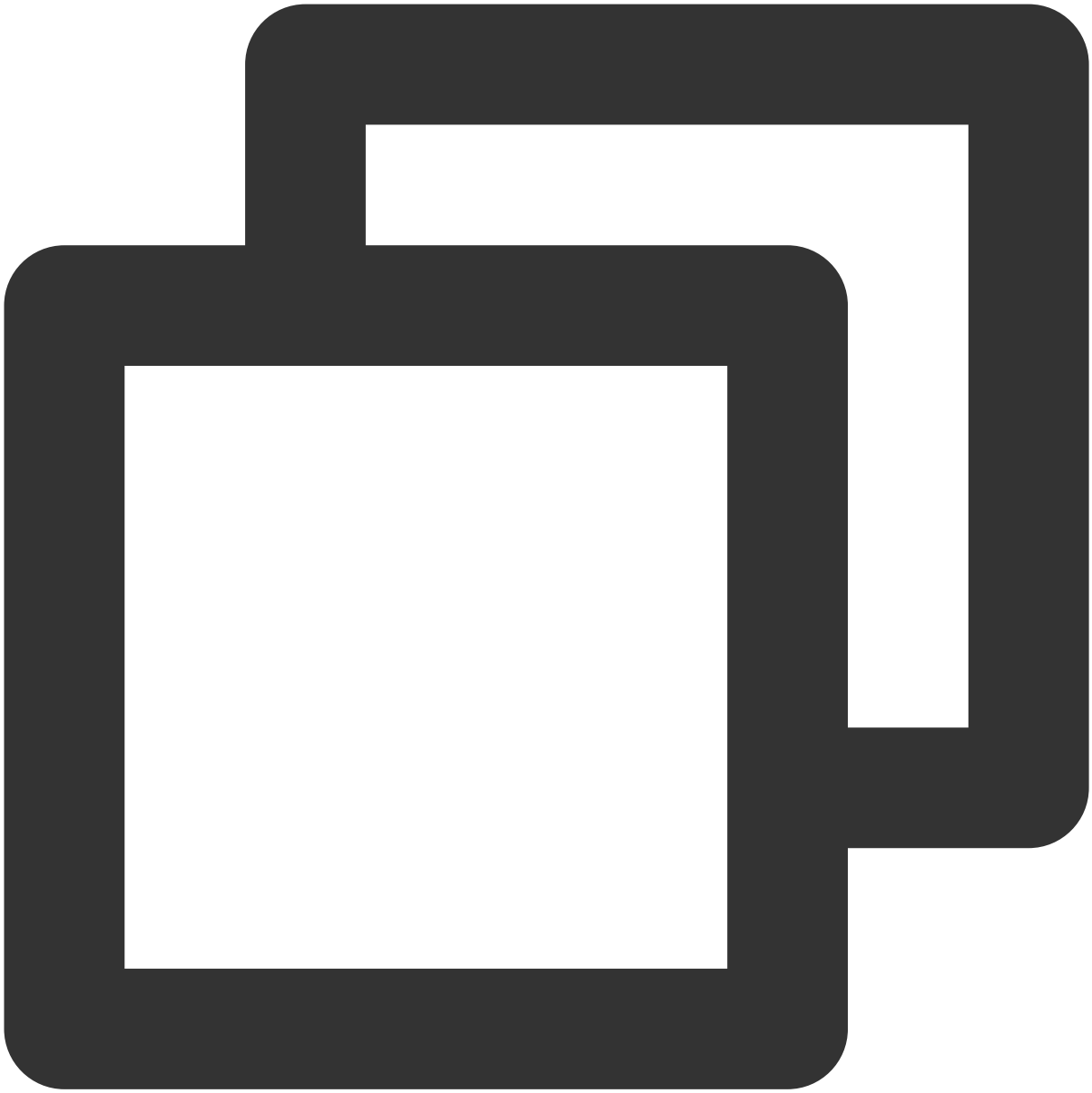
1. SSHツールを使用して新規購入サーバーにログインします。
2. 次のコマンドを実行し、ウェブサイトのディレクトリ所有者を変更します。



```
chown -R nginx:nginx /var/www
```

3. 現在のNginx設定ファイル `/etc/nginx/nginx.conf` をバックアップします。次の方法で行うことができます。

1. `cp /etc/nginx/nginx.conf ~/nginx.conf.bak` を実行し、現在の設定ファイルをホーム（HOME）ディレクトリにバックアップします。
2. SFTPまたはSCPなどのソフトウェアを使用して、現在の設定ファイルをローカルコンピュータにダウンロードします。
3. `/etc/nginx/nginx.conf` を変更するか、または次の内容に置き換えます。



```
# For more information on configuration, see:
#   * Official English Documentation: http://nginx.org/en/docs/
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;
```

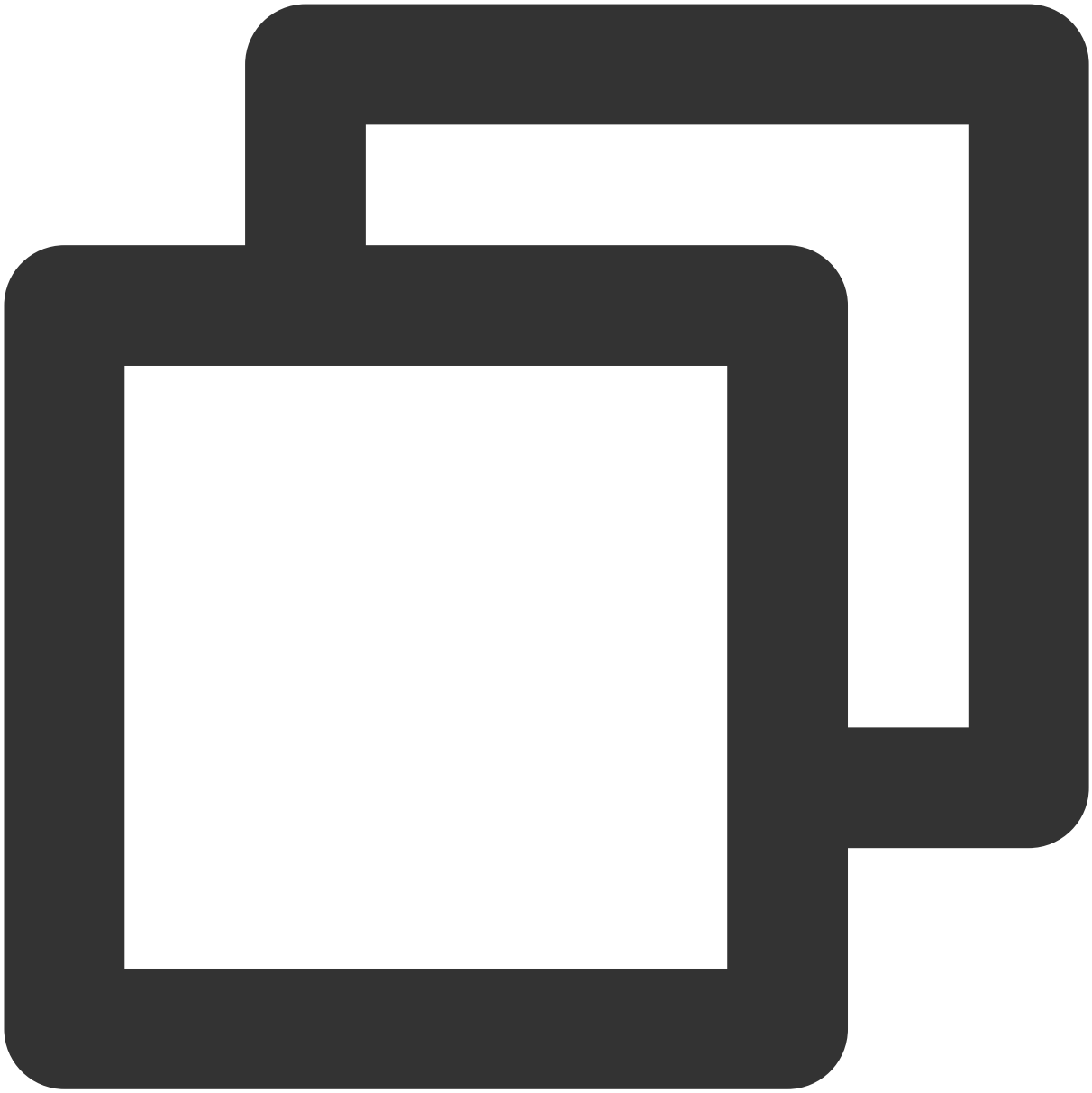
```
events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;
        root         /var/www/nextcloud;

        add_header Referrer-Policy "no-referrer" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-Download-Options "noopen" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header X-Permitted-Cross-Domain-Policies "none" always;
        add_header X-Robots-Tag "none" always;
        add_header X-XSS-Protection "1; mode=block" always;

        client_max_body_size 512M;
        fastcgi_buffers 64 4K;

        gzip on;
        gzip_vary on;
        gzip_comp_level 4;
        gzip_min_length 256;
```

```
        gzip_proxied expired no-cache no-store private no_last_modified no_etag aut
        gzip_types application/atom+xml application/javascript application/json app

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
            try_files $uri $uri/ =404;
            index index.php;
        }

        location ~ ^\\/(?:build|tests|config|lib|3rdparty|templates|data)\\/ {
            deny all;
        }
        location ~ ^\\/(?:\\.|autotest|occ|issue|indie|db_|console) {
            deny all;
        }

        location ~ ^\\/(?:index|remote|public|cron|core\\/ajax\\/update|status|ocs\
            fastcgi_split_path_info ^(.+?\\.php)(\\/.*|)$;
            set $path_info $fastcgi_path_info;
            try_files $fastcgi_script_name =404;
            include        fastcgi_params;
            fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $path_info;
            fastcgi_param modHeadersAvailable true;
            fastcgi_pass   127.0.0.1:9000;
            fastcgi_intercept_errors on;
            fastcgi_request_buffering off;
        }

        location ~ ^\\/(?:updater|oc[ms]-provider)(?:$|\\/) {
            try_files $uri/ =404;
            index index.php;
        }

        location ~ \\.(css|js|svg|gif)$ {
            add_header Cache-Control "max-age=15778463";
        }

        location ~ \\.woff2?$ {
            add_header Cache-Control "max-age=604800";
        }
    }

# Settings for a TLS enabled server.
#
```

```
#    server {
#        listen       443 ssl http2 default_server;
#        listen       [::]:443 ssl http2 default_server;
#        server_name  _;
#        root         /usr/share/nginx/html;
#
#        ssl_certificate "/etc/pki/nginx/server.crt";
#        ssl_certificate_key "/etc/pki/nginx/private/server.key";
#        ssl_session_cache shared:SSL:1m;
#        ssl_session_timeout  10m;
#        ssl_ciphers HIGH:!aNULL:!MD5;
#        ssl_prefer_server_ciphers on;
#
#        # Load configuration files for the default server block.
#        include /etc/nginx/default.d/*.conf;
#
#        location / {
#        }
#
#        error_page 404 /404.html;
#            location = /40x.html {
#        }
#
#        error_page 500 502 503 504 /50x.html;
#            location = /50x.html {
#        }
#    }

}
```

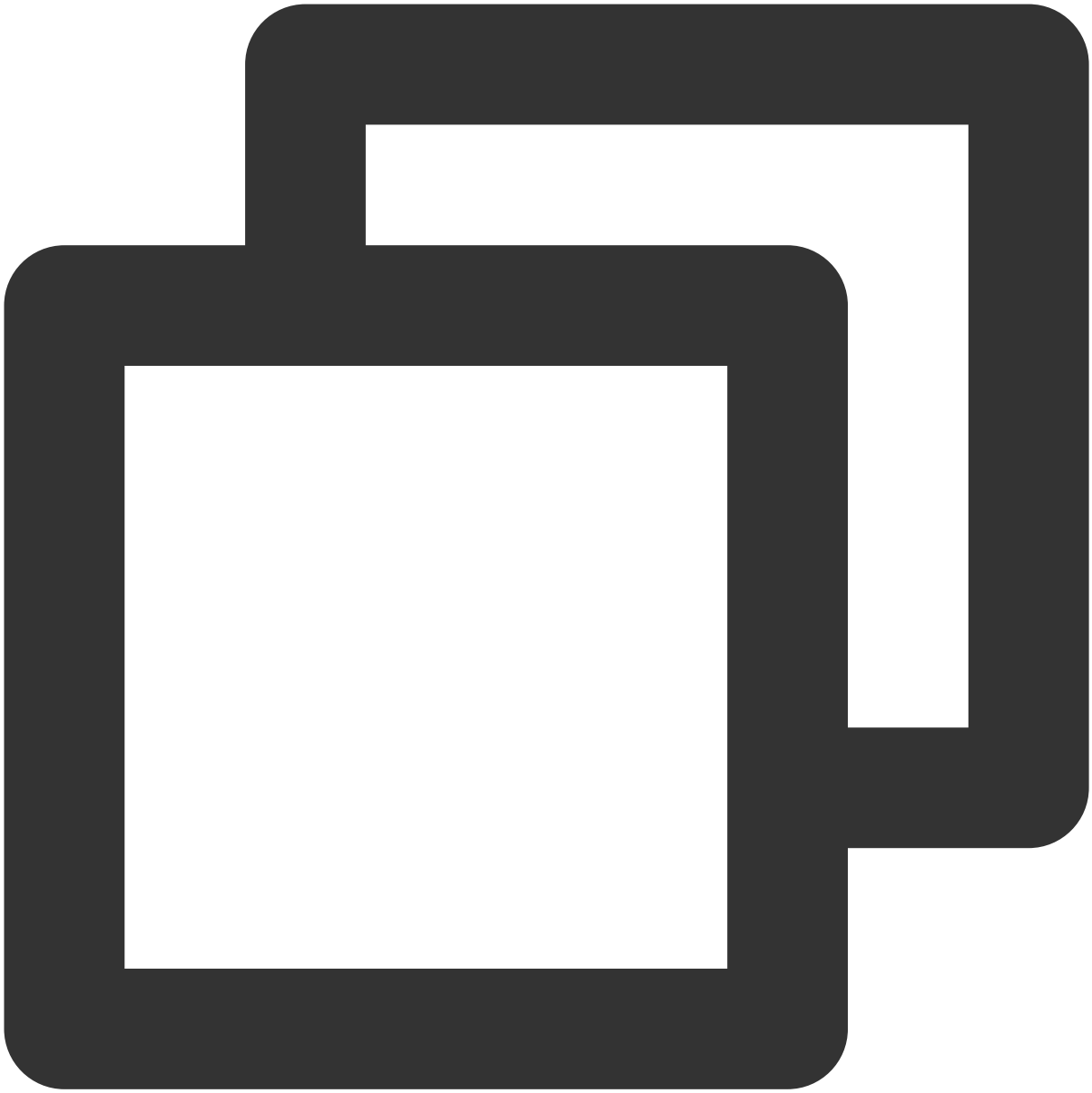
4. 次のコマンドを順に実行し、Nginxサービスを起動します。

コマンド1：



```
systemctl enable nginx
```

コマンド2：

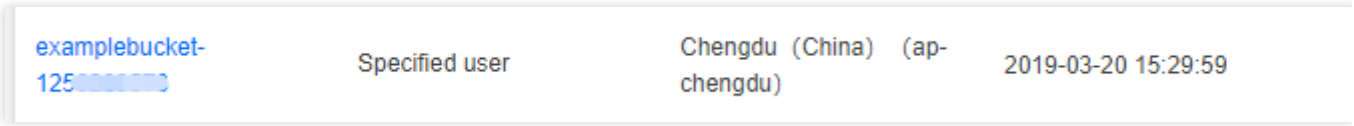


```
systemctl start nginx
```

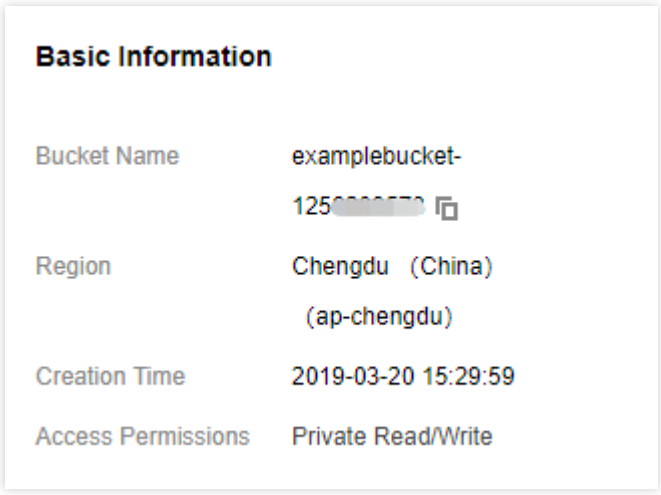
NextCloudサーバーでのCOS使用設定

COS関連情報の取得

1. COSコンソールにログインします。
2. 作成済みのバケットを見つけ、バケット名をクリックします。



3. 左側ナビゲーションバーで**概要**タブを選択し、**基本情報**の**バケット名**と**所属リージョン**の中の英語部分をメモします。

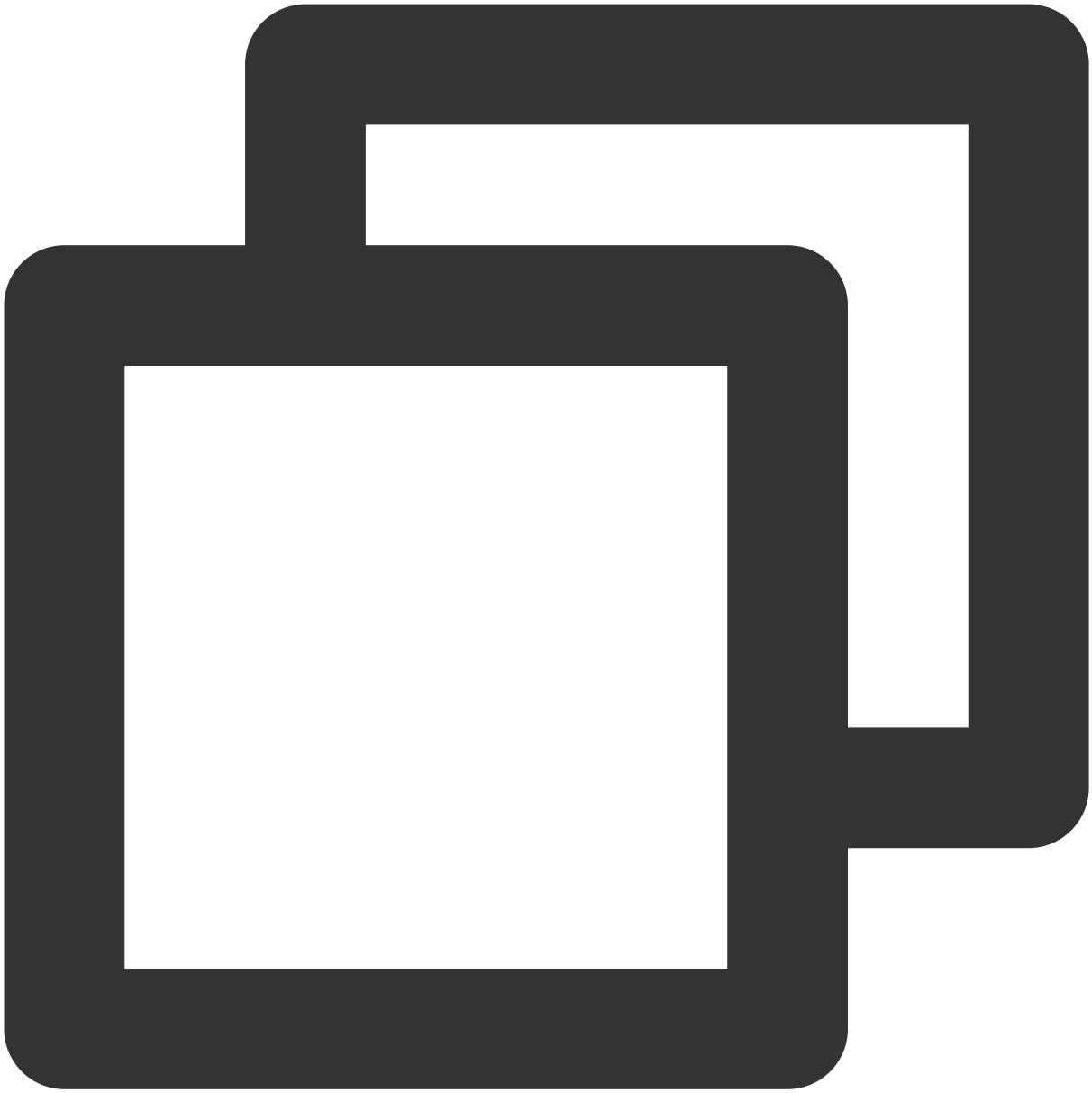


APIキーの取得

サブアカウントキーを使用し、最小権限ガイドに従うことで、使用上のリスクを低減させることをお勧めします。サブアカウントキーの取得については、サブアカウントのアクセスキー管理をご参照ください。

NextCloudサーバー設定ファイルの変更

1. テキスト編集ツールを使用して `config.php` を作成し、次の内容を入力して、メモの内容に基づいて関連の値を変更します。

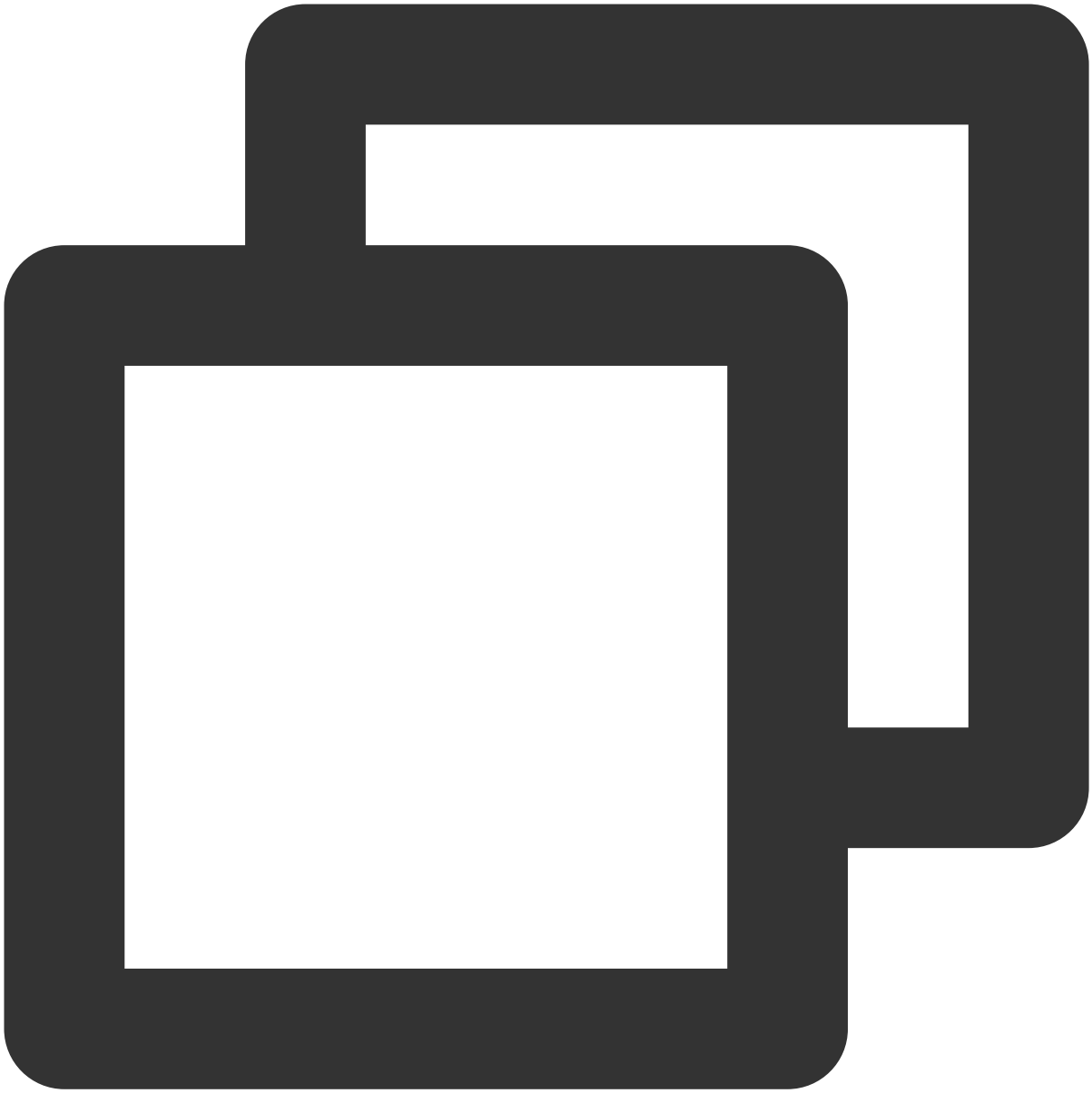


```
<?php
$CONFIG = array(
  'objectstore' => array(
    'class' => '\\\\OC\\\\Files\\\\ObjectStore\\\\S3',
    'arguments' => array(
      'bucket' => 'nextcloud-1250000000', // バケット名（スペース名）
      'autocreate' => false,
      'key'  => 'AKIDxxxxxxxx', // ユーザーのSecretIdに置き換えます
      'secret' => 'xxxxxxxxxxxx', // ユーザーのSecretKeyに置き換えます
      'hostname' => 'cos.<Region>.myqcloud.com', // <Region>を所属リージョンに変更します
      'use_ssl' => true,
```

```
      ),
    ),
);
```

下図に示すように：

2. このファイルを保存し、`/var/www/nextcloud/config/` ディレクトリ下にアップロードします（ファイル名は `config.php` のままにしておきます）。SFTPまたはSCPソフトウェアでファイルをアップロードするか、または `rz -bye` コマンドによってアップロードすることができます。
3. 次のコマンドを実行し、設定ファイルの所有者を変更します。



```
chown nginx:nginx /var/www/nextcloud/config/config.php
```

ドメイン名の設定

ご自身のNextCloudサーバーにアクセスするのに、IPアドレスではなくご自身のドメイン名の使用を予定している場合は、各ドメイン名登録プロバイダの説明ドキュメントをご参照の上、新しいドメイン名を登録してCVMのIPアドレスに解決し、ICP登録を完了してください。

NextCloudサーバーはインストールの過程で、インストール時に使用したドメイン名またはIPアドレスを記録するため、インストールの開始前にドメイン名の登録、解決、ICP登録を完了しておくとともに、ドメイン名を使用してNextCloudサーバーのセキュリティ画面にアクセスすることをお勧めします。
NextCloudサーバーの完成後にドメイン名またはIPアドレスを変更したい場合は、ご自身で `/var/www/nextcloud/config/config.php` 設定ファイル内の `trusted_domains` を変更することができます。詳細については、NextCloud公式ドキュメントをご参照ください。

NextCloudサーバーのインストール

1. ブラウザを使用してNextCloudサーバーにアクセスし、管理者ユーザー名とパスワードを作成して忘れないように記憶しておきます。
2. **ストレージとデータベース**を開き、下表の説明に従って設定します。

設定項目	値
データディレクトリ	/var/www/nextcloud/data（デフォルトを維持）
データベース設定	MySQL/MariaDB
データベースユーザー	root
データベースパスワード	TencentDB for MySQLを初期化した際に入力したrootパスワード
データベース名	nextcloud（またはその他の使用されていないデータベース名）
データベースホスト（デフォルトではlocalhostと表示）	TencentDB for MySQLのプライベートネットワークアドレス

3. **インストール完了**をクリックし、NextCloudサーバーのインストールが完了するまで待ちます。4. インストール中に504 Gateway Timeoutなどのエラーメッセージが表示された場合は、そのまま更新してリトライします。5. インストール完了後、管理者アカウントを使用してNextCloudサーバーにログインすると、Web版NextCloudの使用を開始できます。

NextCloudサーバーのチューニング

バックエンドタスク

NextCloudサーバーは、ユーザーとのインタラクションが必要ないときに、一部のバックエンドタスク（例えばデータベースのクリーンアップ操作など）を実行しなければならない場合があります。PHPには、PHPベースの

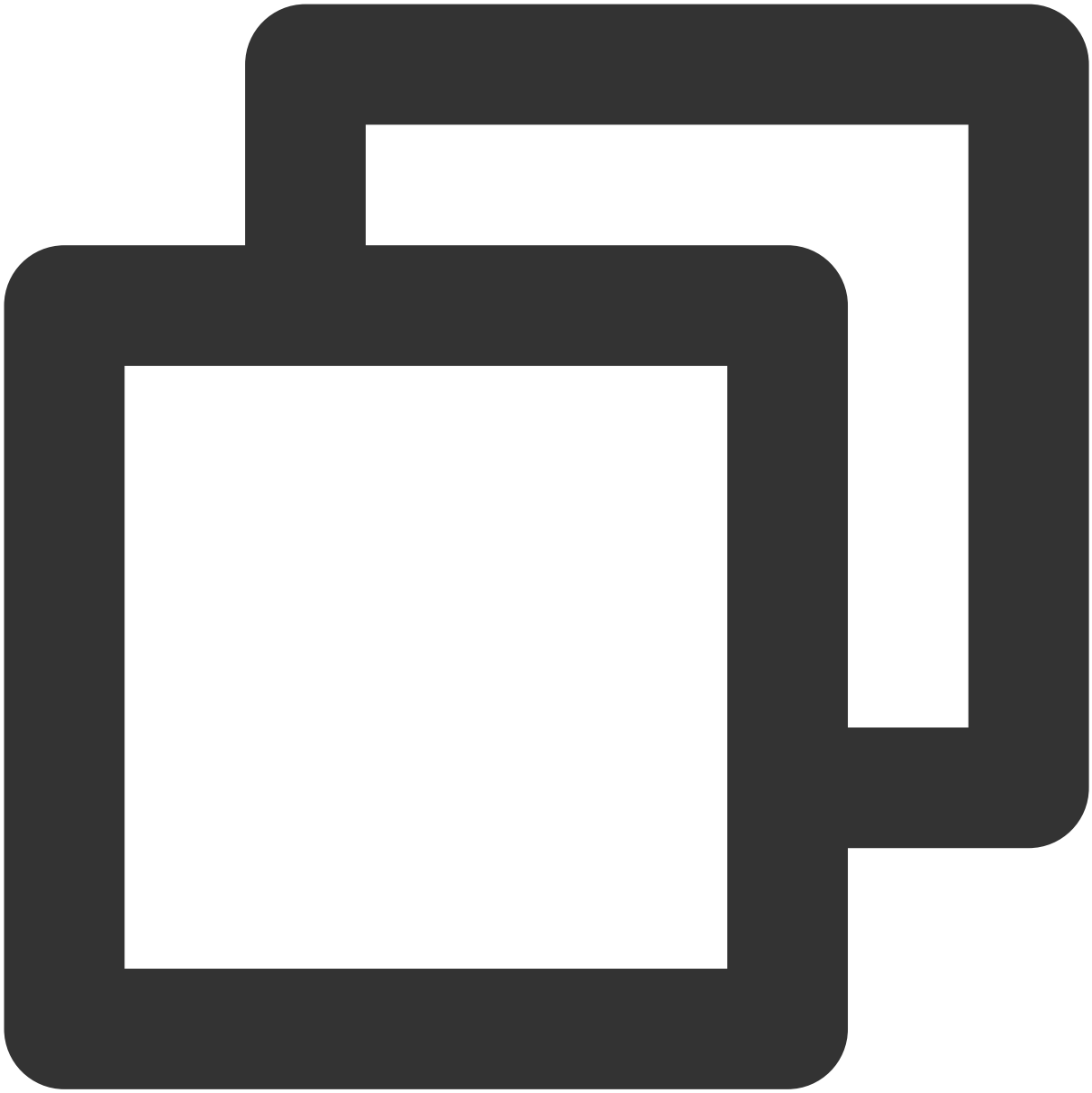
プログラムが内部で独立した作業プロセスまたはスレッドを維持できないように制限する実行特性があるため、バックエンドタスクのようなケースでは、対応するPHPプログラムを外部から能動的に呼び出して実行する必要があります。

NextCloudサーバーは3種類のバックエンドタスクの呼び出しメソッドを提供しています。デフォルトは、ウェブページ側のログインユーザーに、ブラウザからAJAXリクエストを自動送信してサーバーのバックエンドタスクを実行するようリマインドする方法です。このメソッドはユーザーのログイン状態に強く依存し、ユーザーがログインしていないとこれらのバックエンドタスクが実行できないため、信頼性が最も低くなっています。

AJAXをベースにしたバックエンドタスクの低信頼性の問題を回避するためには、Linuxのcronを使用してバックエンドタスクを設定する方法を推奨します。Linuxのcronはタスクがリマインドされる時間を正確に制御でき、例えば5分ごと（分数は5の整数倍）や毎時10分などとすることができます。定義できる時間粒度には分、時間、1か月のうちの特定の日、月、特定の曜日などがあり、一部のOSでは秒と年もサポートされているため、高い柔軟性を有します。cronと関連説明および設定については、関連の資料をご参照ください。

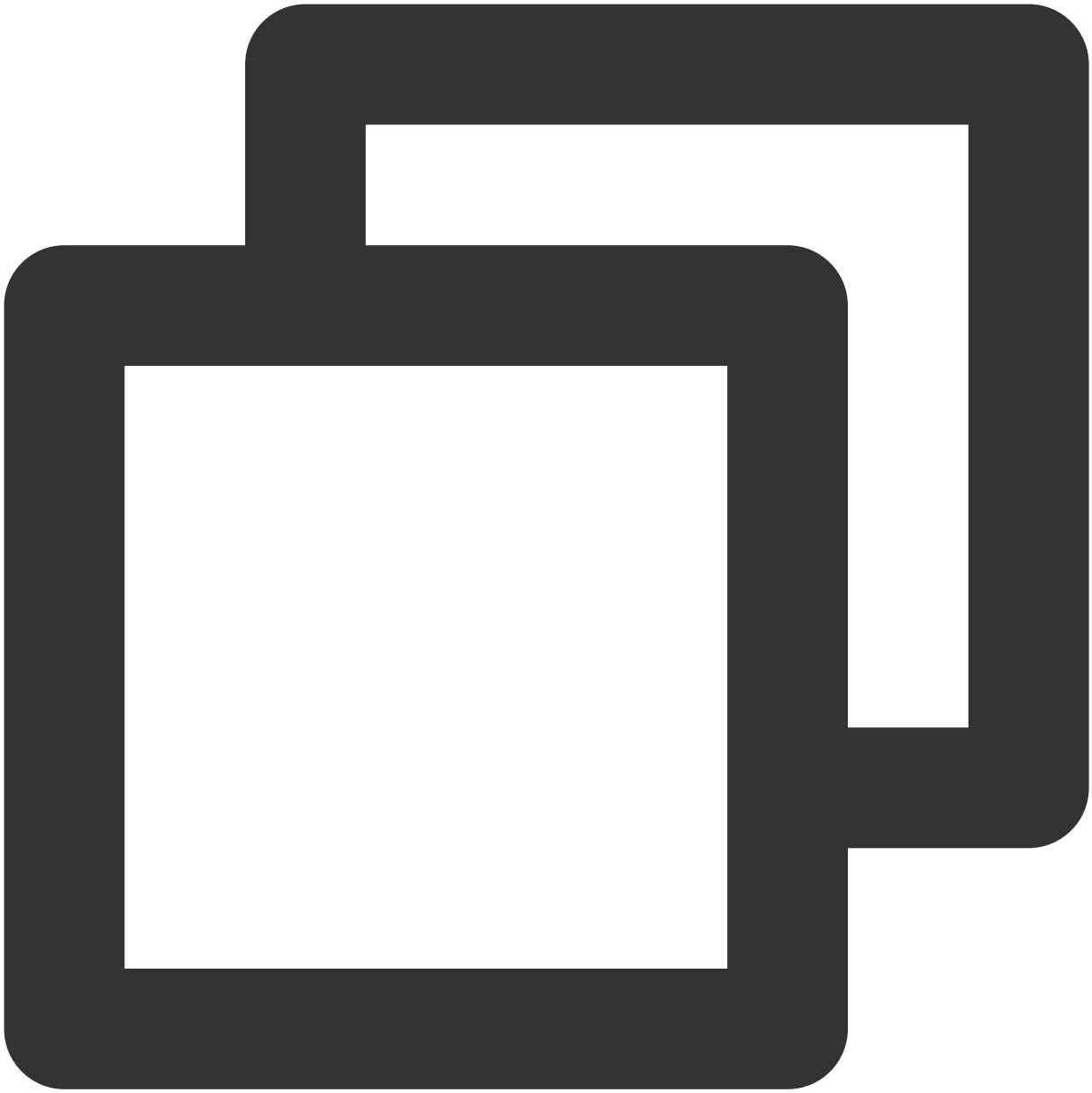
cronを設定してNextCloudサーバーのバックエンドタスクを行えるようにする方法について、以下でご説明します。

1. SSHツールを使用して新規購入サーバーにログインします。
2. 次のコマンドを実行し、PHPモジュールをインストールします。



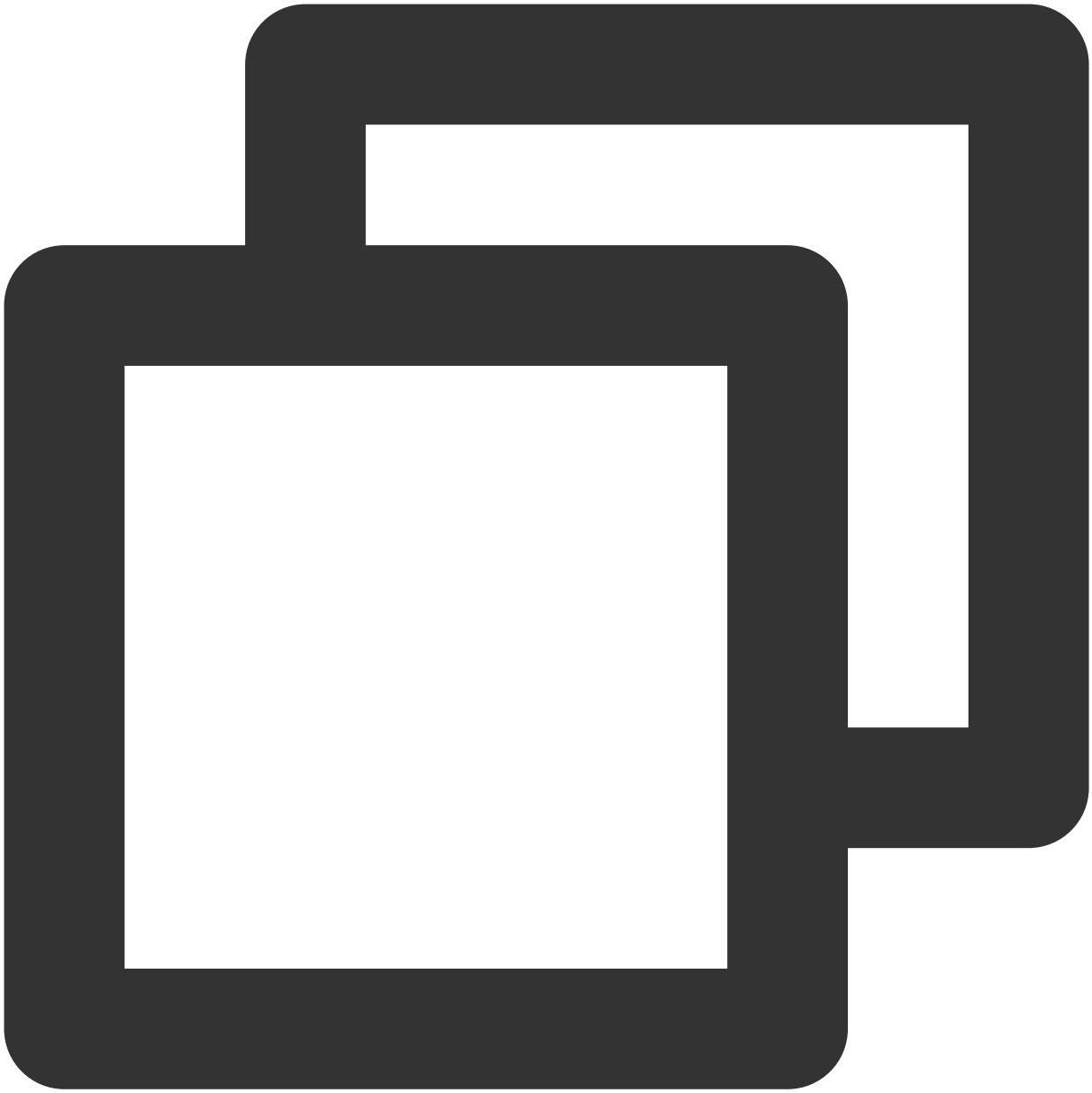
```
yum install php-posix
```

3. 次のコマンドを実行し、Nginxアカウントで使用するcronの設定を開くか、または作成します。



```
crontab -u nginx -e
```

4. 続いての編集画面はvi/vimです。 `i` キーを押して編集モードに入り、次の内容の1行を挿入します。



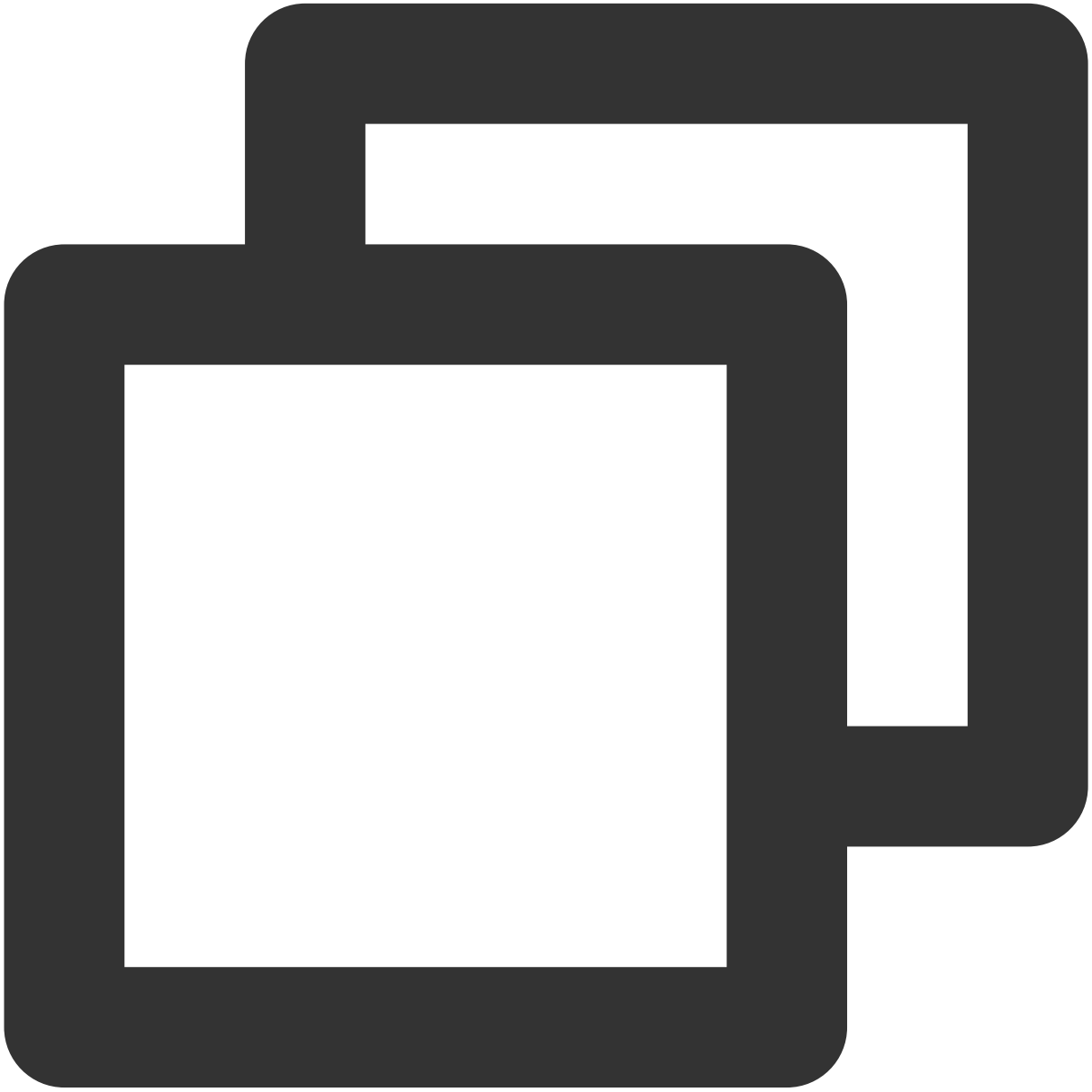
```
*/5 * * * * php -f /var/www/nextcloud/cron.php
```

その後、 `ESC` を押して編集モードを終了し、 `:wq` を入力して保存し、終了します（vi/vimの詳細な操作ガイドについては関連ドキュメントをご参照ください）。
上記の設定ではNextCloud公式が推奨する5分に1回の実行を使用しています（分数は5の整数倍）。5分後にバックエンドタスクの実行が完了すると、ブラウザを開いてNextCloudサーバーにログインできるようになります。右上隅にあるユーザー名のイニシャルのアイコンをクリックし、**設定**に進み、左側メニューから**基本設定**に進むと、**バックエンドタスク**のところにデフォルトでCronが選択されていることを確認できます。

メモリキャッシュ

PHPはOPcacheによってパフォーマンスを向上させることができ、NextCloudサーバーもAPCuメモリキャッシュの使用によるパフォーマンス向上をサポートしています。関連の操作フローを以下でご説明します。

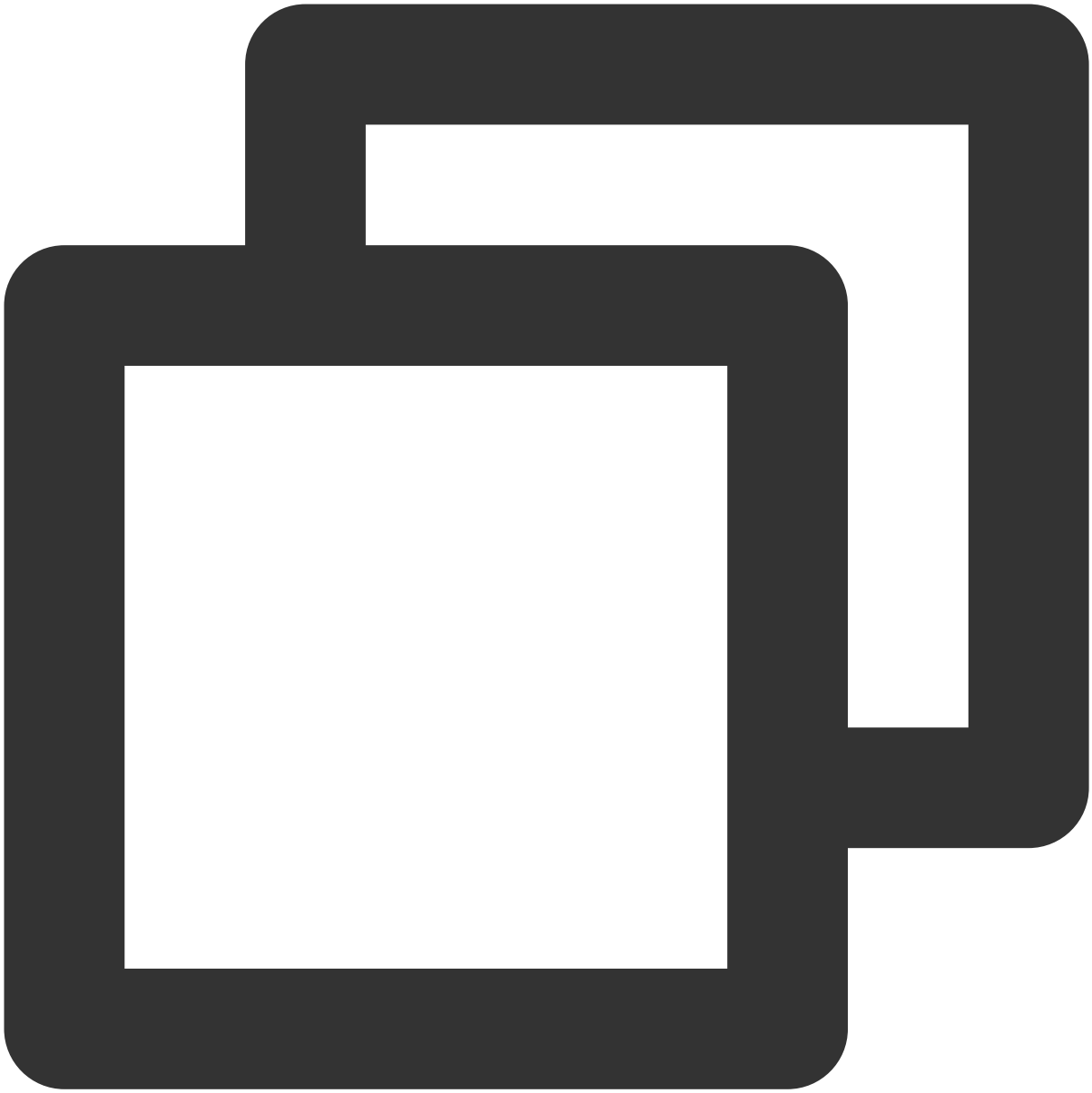
1. SSHツールを使用して新規購入サーバーにログインします。
2. 次のコマンドを実行し、PHPモジュールをインストールします。



```
yum install php-pecl-apcu
```

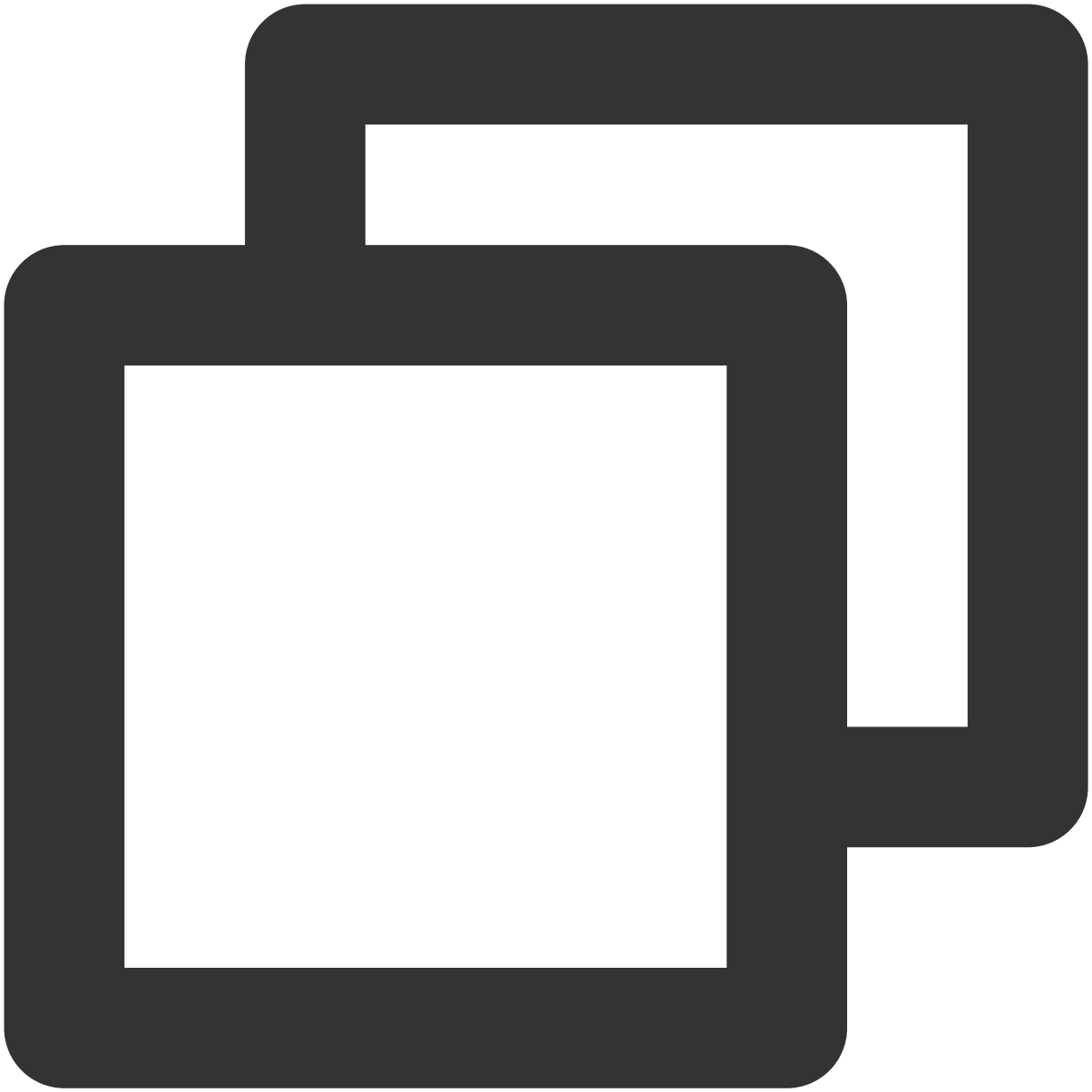
3. 次のコマンドを順に実行し、Nginx とPHP-FPMを再起動します。

コマンド1：



```
systemctl restart nginx
```

コマンド2：

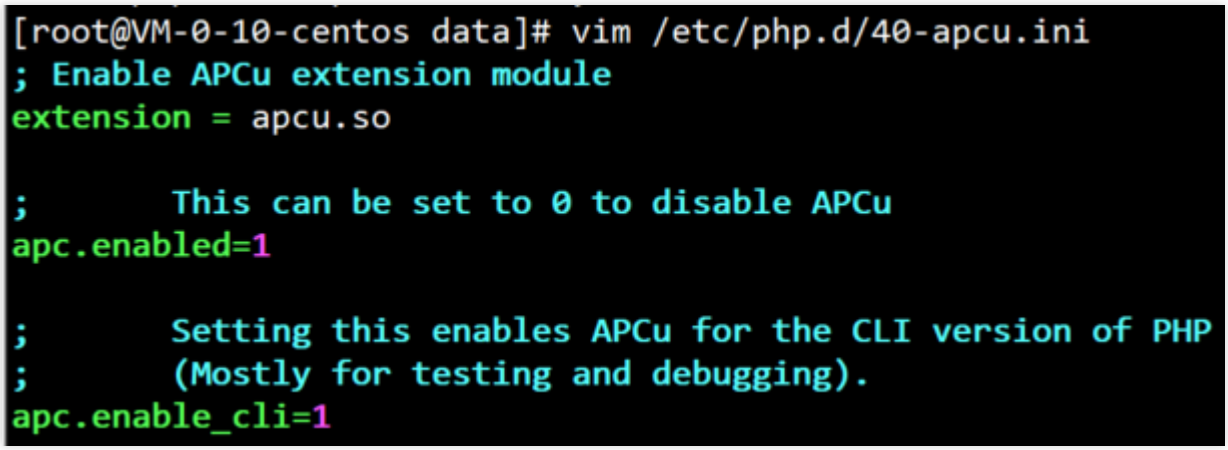


```
systemctl restart php-fpm
```

4. `vim /var/www/nextcloud/config/config.php` を実行し、NextCloudサーバーの設定ファイルを開き、 `$CONFIG = array (` の中に `'memcache.local' => '\\OC\\Memcache\\APCu',` という1行を追加します。その後ファイルを保存して終了します。

5. cronを設定してバックエンドタスクを最適化した場合は、PHPのapc設定も変更する必要があります。`vim /etc/php.d/40-apcu.ini` を実行し、PHP APCuの設定ファイルを開き、`;apc.enable_cli=0` を `apc.enable_cli=1` に変更し（同時に前のセミコロンを削除する必要があることにご注意ください）、保存して終了します。パス `/etc/php.d/40-apcu.ini` が存在しない場合は、apc

またはapcuの文字がある `.ini` 設定ファイルをご自身で `/etc/php.d/` ディレクトリから見つけて編集してください。



クライアントアクセス設定

NextCloud公式はデスクトップ同期クライアントとモバイルクライアントを提供しており、NextCloud公式サイトまたは各大手アプリストアからダウンロードできます。NextCloudを設定する際には、NextCloudのサーバーアドレス（ドメイン名またはIP）を入力し、その後ご自身のユーザー名とパスワードを入力してログインすると、クライアントの使用を開始できます。

WindowsサーバーにCOSをローカルディスクとしてマウント

最終更新日：：2024-06-26 10:47:37

ユースケース

現在、WindowsシステムTencentのCloud Object Storage(COS)を操作するための主なメソッドとしては、API、COSBrowser、COSCMDツールがあります。
Windowsサーバーを好んで使用するユーザーにとっては、COSBrowserツールはほとんどの場合、クラウドストレージとしてしか使えないため、サーバー上でプログラムを直接使用したり、操作したりするには適していません。ここでは、ストレージ料金が安価なCOSをWindowsサーバーにマウントすることで、ローカルディスクにマッピングする方法についてご説明します。
説明：
このプラクティス事例はWindows 7 / Windows Server 2012 / 2016 / 2019 / 2022システムに適合します。

操作手順

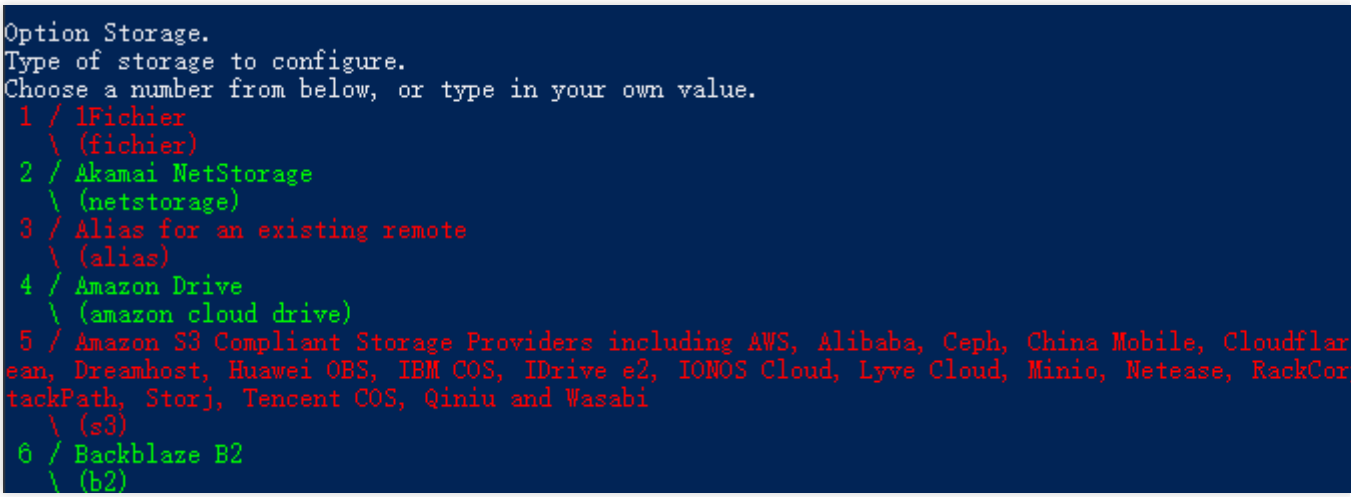
ダウンロードとインストール

このプラクティス事例では次の3種類のソフトウェアを使用します。お使いのシステムに適したソフトウェアバージョンを選択してインストールすることができます。
1. Githubに移動してWinfspをダウンロードします。
このプラクティス事例でダウンロードするバージョンはwinfsp-1.12.22301です。ダウンロードが完了すると、手順に沿ってデフォルトでインストールされます。
説明：
Windows Server 2012 R2はWinfsp 1.12.22242バージョンには適合せず、Winfsp 1.11.22176バージョンに適合します。
2. Git公式サイトまたはGithubに移動してGitツールをダウンロードします。
このプラクティス事例でダウンロードするバージョンはGit-2.38.1-64-bitです。ダウンロードが完了すると、手順に沿ってデフォルトでインストールされます。
3. Rclone公式サイトまたはGithubに移動してRcloneツールをダウンロードします。
このプラクティス事例でダウンロードするバージョンはrclone-v1.60.1-windows-amd64です。このソフトウェアはインストールの必要がなく、ダウンロード後、任意の英語ディレクトリに解凍するだけで完了します（解凍したパスに中国語が含まれているとエラーとなる場合があります）。このプラクティス事例のパスの例は、E:\\AutoRcloneです。

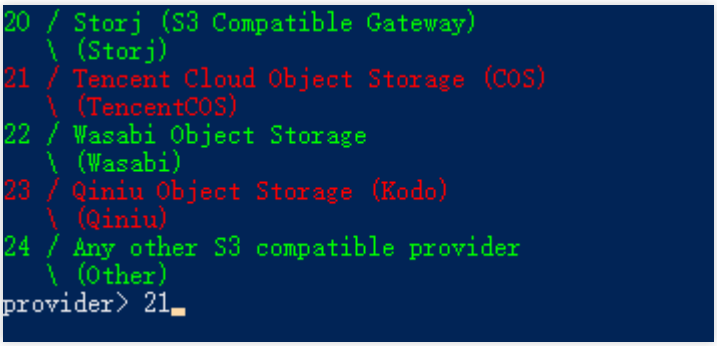
説明：
Githubのダウンロード速度が遅かったり、開かなかったりすることがありますので、他の公式チャネルからご自分でダウンロードすることもできます。

Rcloneの設定

注意：
以下の設定手順はrclone-v1.60.1-windows-amd64バージョンを例としています。その他のバージョンでは設定手順に若干の違いがあるため、適宜調整してください。
1. 任意のフォルダを開き、左側のナビゲーションディレクトリから**このPC**を見つけ、右クリックして**プロパティ > システムの詳細設定 > 環境変数 > システム変数 > Path**を選択し、**新規作成**をクリックします。
2. ポップアップウィンドウに、Rcloneが解凍された後のパス(E:\\AutoRclone)を入力し、**OK**をクリックします。
3. Windows Powershellを開き、`rclone --version` コマンドを入力して**Enter**を押し、Rcloneが正しくインストールされているか確認します。
4. Rcloneが正しくインストールされたことを確認したら、Windows Powershellでコマンド `rclone config` を入力して**Enter**を押します。
5. Windows Powershellに**n**と入力して**Enter**を押し、New remoteを新規作成します。
6. Windows Powershellにディスクの名前（例：myCOS）を入力し、**Enter**を押します。
7. 表示されたオプションの中から、“Tencent COS”を含むオプションを選択、すなわち**5**を入力して**Enter**を押します。



8. 表示されたオプションの中から、“TencentCOS”を含むオプションを選択し、**21**を入力して**Enter**を押します。



9. `env_auth>` まで実行したら、Enterを押します。

10. `access_key_id>` まで実行したら、Tencent Cloud COSのアクセスキーSecretIdを入力し、**Enter**を押します。

説明：

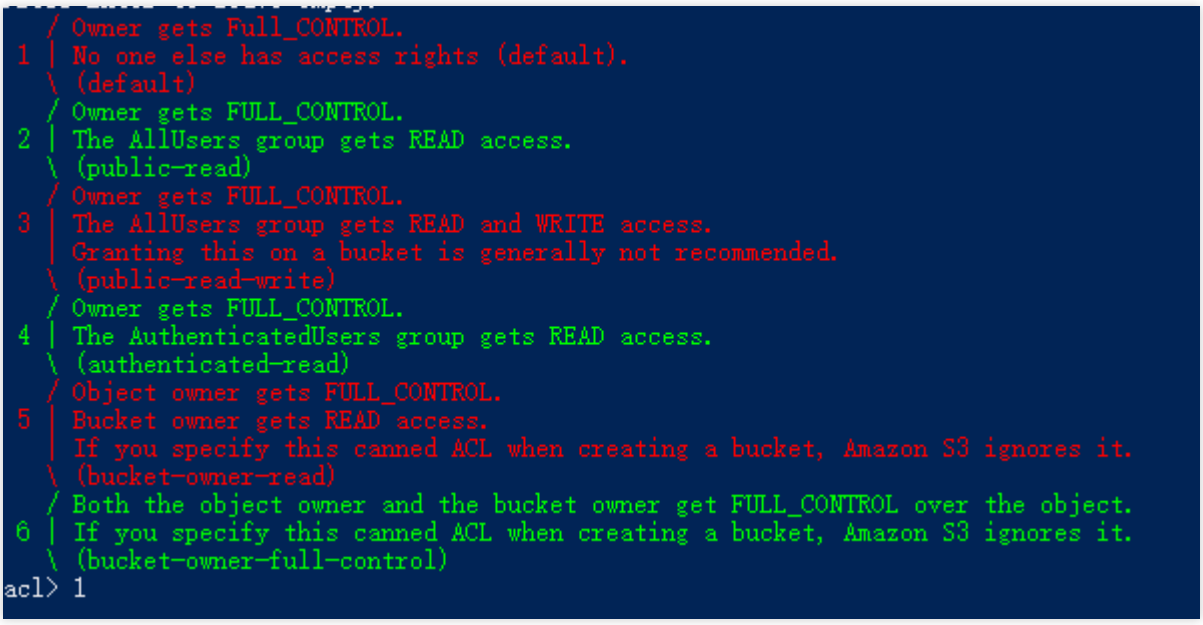
ここではサブアカウント権限を使用することをお勧めします。APIキー管理に移動すると、ご自分のSecretIdとSecretKeyを確認できます。

11. `secret_access_key>` まで実行したら、Tencent Cloud COSのアクセスキーSecretKeyを入力し、**Enter**を押します。

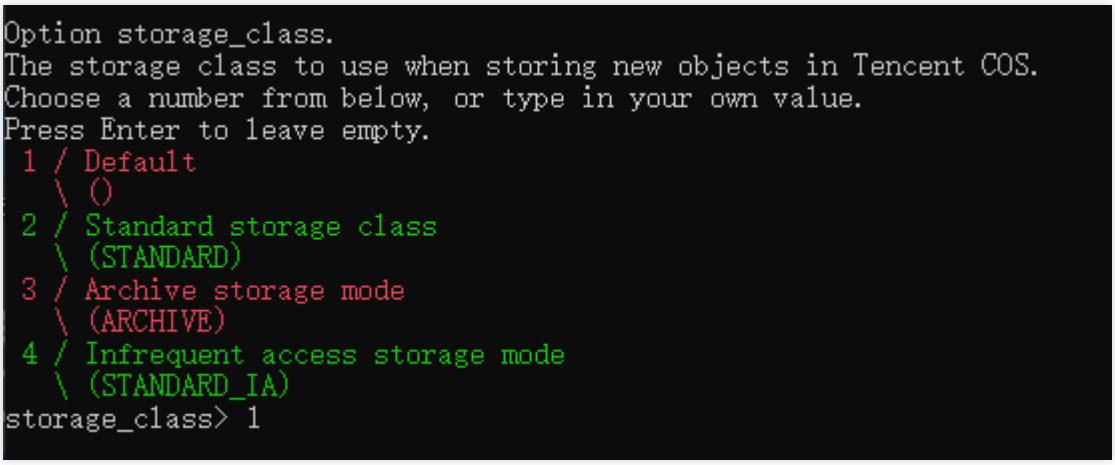
12. 表示されたTencent Cloudの各リージョンのゲートウェイアドレスをもとに、バケットが属するリージョンを確認し、対応するリージョンを選択します。

このプラクティスでは広州を例として、 `cos.ap-guangzhou.myqcloud.com` を選択し、**4**と入力して**Enter**を押します。

13. 表示されたTencent Cloud COSの権限タイプから、実際のニーズに応じてdefault、public-readなどを選択します。ここで選択した権限タイプはオブジェクト権限タイプで、新しくアップロードされたファイルに対してのみ有効です。このプラクティスではdefaultを例として、**1**を入力し、**Enter**を押します。



14. 表示されたTencent Cloud COSのストレージタイプから、実際のニーズに応じてCOSにファイルをアップロードするストレージタイプを選択できます。このプラクティスではDefaultを例として、**1**と入力して**Enter**を押します。



Defaultとはデフォルトを意味します

Standard storage classとは標準ストレージ(STANDARD)を意味します

Archive storage modeとはアーカイブストレージ(ARCHIVE)を意味します

Infrequent access storage modeとは低頻度ストレージ（STANDARD_IA）を意味します

説明：

INTELLIGENT_TIERINGストレージまたはディープアーカイブストレージを設定したい場合は**設定ファイルを変更する**方法を使用し、設定ファイルのstorage_classの値をINTELLIGENT_TIERINGまたはDEEP_ARCHIVEに設定します。ストレージタイプに関するその他の説明については、ストレージタイプの概要をご参照ください。

15. `Edit advanced config? (y/n)` まで実行したら、**Enter**を押します。

16. 情報が正しいことを確認したら、**Enter**を押します。

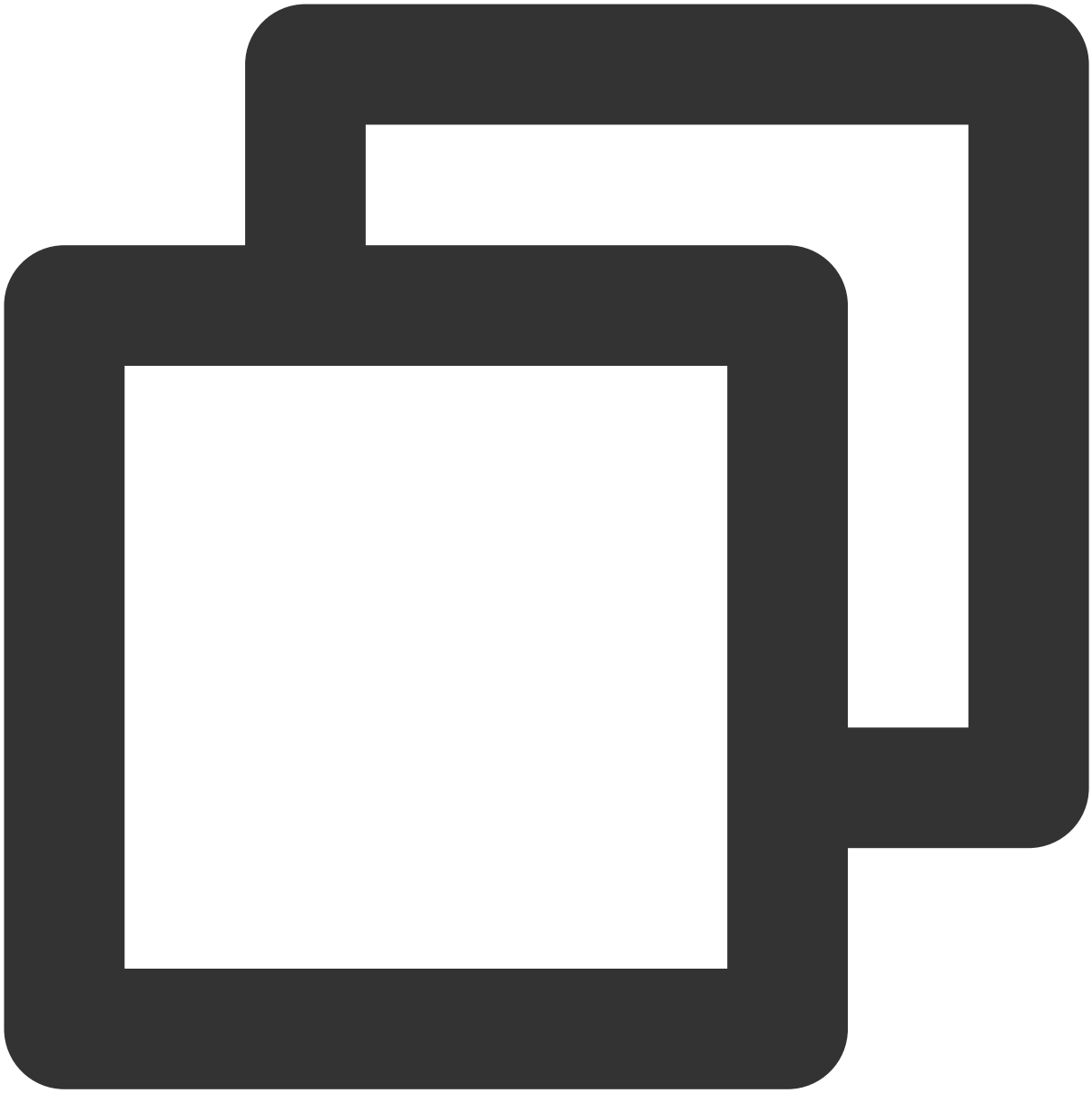
17. **q**と入力し、設定を完了します。

構成ファイルの変更

上記の手順が完了すると、rclone.confという名前の設定ファイルが生成されます。通常は `C:\\Users\\ユーザー名\\AppData\\Roaming\\rclone` フォルダ内にあります。rcloneの設定を変更したい場合はこれを直接変更できます。この設定ファイルが見つからない場合は、コマンドウィンドウで `rclone config file` コマンドを実行し、この設定ファイルを照会することができます。

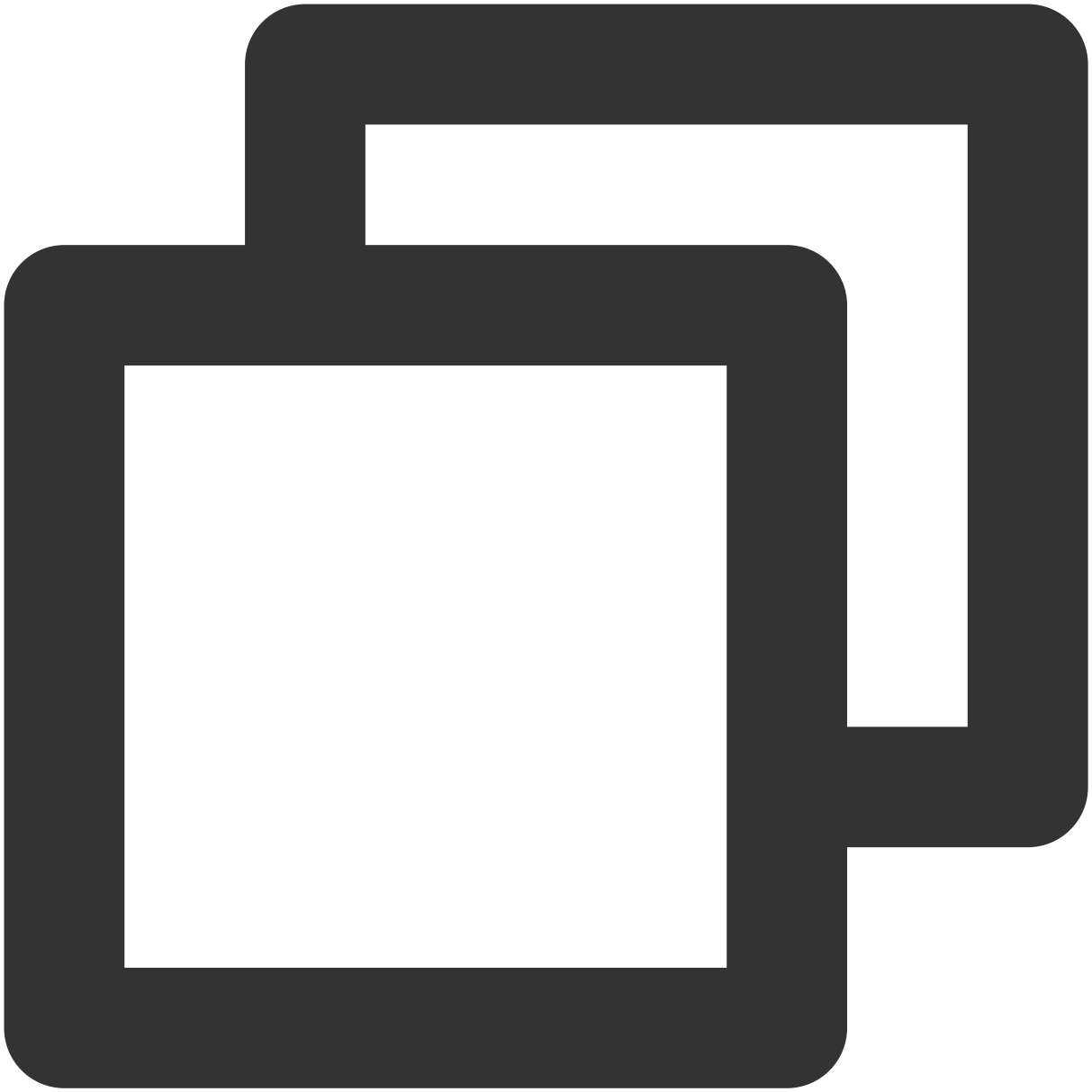
COSをローカルディスクとしてマウント

1. インストールしたGit Bashを開き、実行コマンドを入力します。ここでは2つのユースケース（二択）が提供されていますので、実際のニーズに応じていずれかを選択できます。
LAN共有ドライブとしてマッピングされている場合（推奨）、以下のようにコマンドを実行します。



```
rclone mount myCOS:/ Y: --fuse-flag --VolumePrefix=\\server\\share --cache-dir E:\\
```

ローカルディスクにマッピングされている場合、以下のようにコマンドを実行します。



```
rclone mount myCOS:/ Y: --cache-dir E:\\temp --vfs-cache-mode writes &
```

myCOS：ユーザー定義のディスク名に置き換えます。

Y：マウントしたいハードディスクのボリュームラベル名に置き換えてください。ローカルのC、D、Eドライブなどと重複しないようにしてください。

E:\\tempはローカルキャッシュディレクトリで、ご自分で設定できます。注意：ユーザーがディレクトリの権限を持つことを確実にする必要があります。

「The service rclone has been started」と表示されたらマウント成功です。

2. **exit**と入力し、ターミナルからログアウトします。

3. ローカルコンピュータの**マイコンピュータ**にmyCOS(Y:)という名前のディスクがあります。
ディスクを開くと、広州の全リージョンを含むすべてのバケット名が表示されます。この時点で、アップロード、ダウンロード、作成、削除など、ローカルディスクの通常操作を行うことができます。

注意：

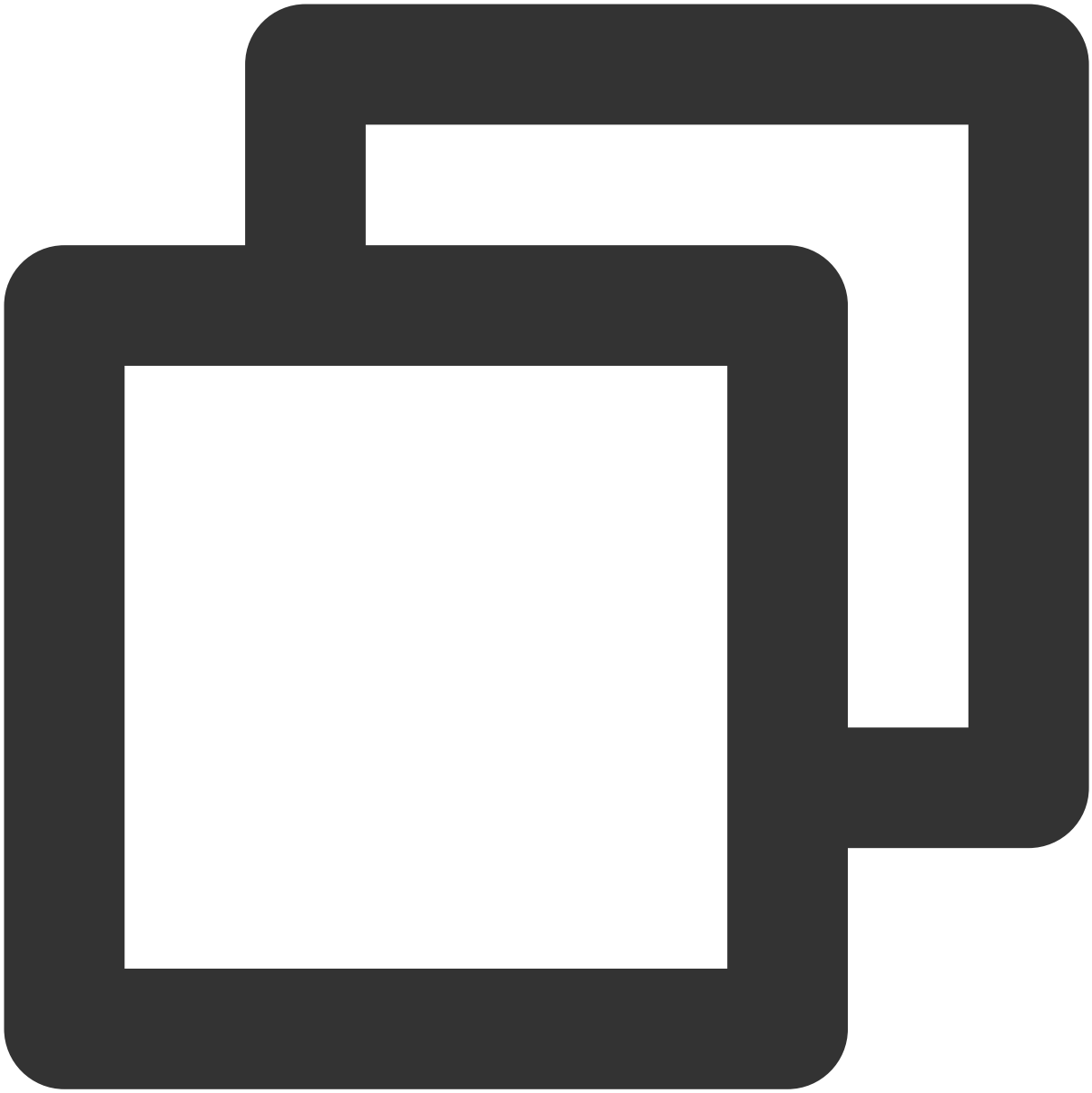
操作中にエラーが発生した場合は、git bashソフトウェアでエラーメッセージの詳細情報を確認してください。
マウントされているディスクでバケットに対して削除操作を行うと、バケット内にファイルが存在するかどうかに関わらず削除されますので、慎重に操作してください。
マウントされているディスクのバケット名を変更すると、COSバケット名も変更されますので、慎重に操作してください。

起動したらハードディスクを自動的にマウントするように設定

上記のようにコンピュータを再起動するとマッピングされたディスクは消失してしまうため、再度手動で操作する必要があります。そこで、自動起動装置を設定して、サーバーを再起動するたびに自動的にディスクがマウントされるようにします。
1. RcloneインストールディレクトリE:\\AutoRcloneに、それぞれstartup_rclone.vbsとstartup_rclone.batファイルを作成します。

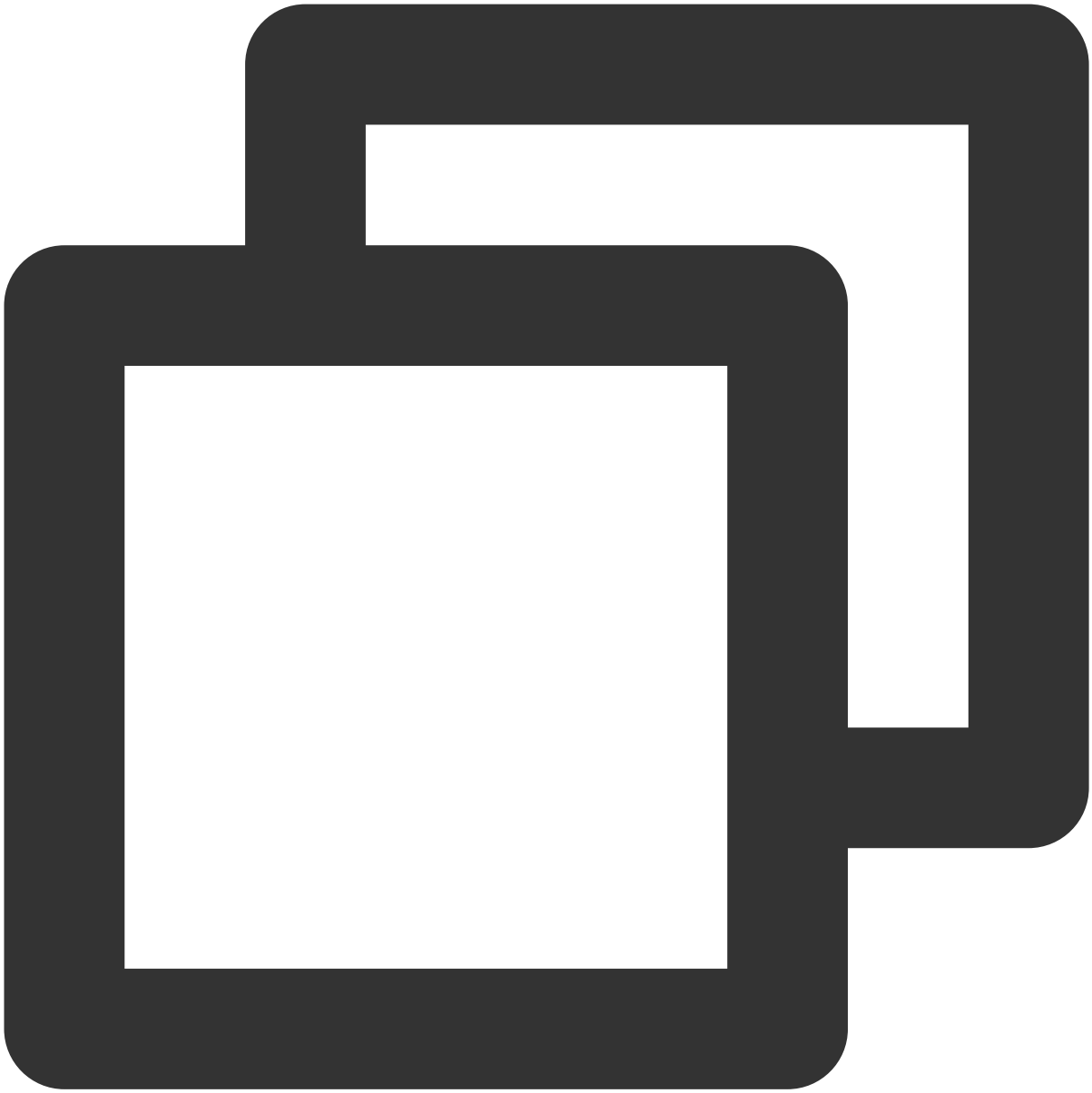
説明：

Powershellでテキストファイルを作成する際はエンコードに注意する必要があります。そうしなければ、生成した.bat、 .vbsなどのテキストファイルが実行できなくなります。
2. startup_rclone.batに、以下のマウントコマンドを記述します。
LAN共有ドライブとしてマッピングされている場合、以下のコマンドを入力します。



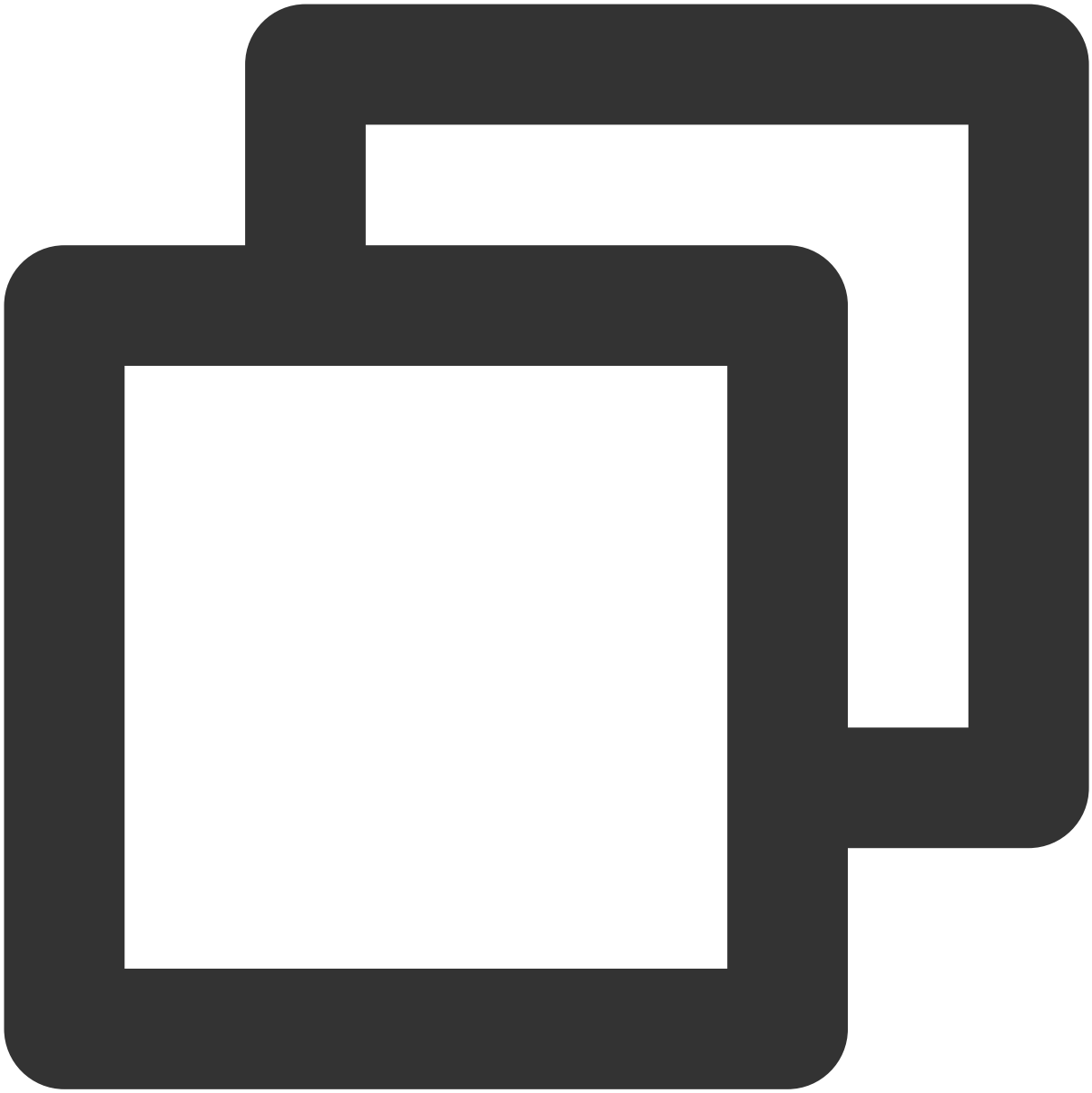
```
rclone mount myCOS:/ Y: --fuse-flag --VolumePrefix=\\server\\share --cache-dir E:\\
```

ローカルディスクにマッピングされている場合、以下のコマンドを入力します。



```
rclone mount myCOS:/ Y: --cache-dir E:\\temp --vfs-cache-mode writes &
```

3. startup_rclone.vbsに、以下のコードを記述します。



```
CreateObject("WScript.Shell").Run "cmd /c E:\\AutoRclone\\startup_rclone.bat",0
```

注意：

コード内のパスを実際のパスに変更してください。

4. startup_rclone.vbsファイルを %USERPROFILE%\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startupフォルダにカットします。

5. サーバーを再起動します。

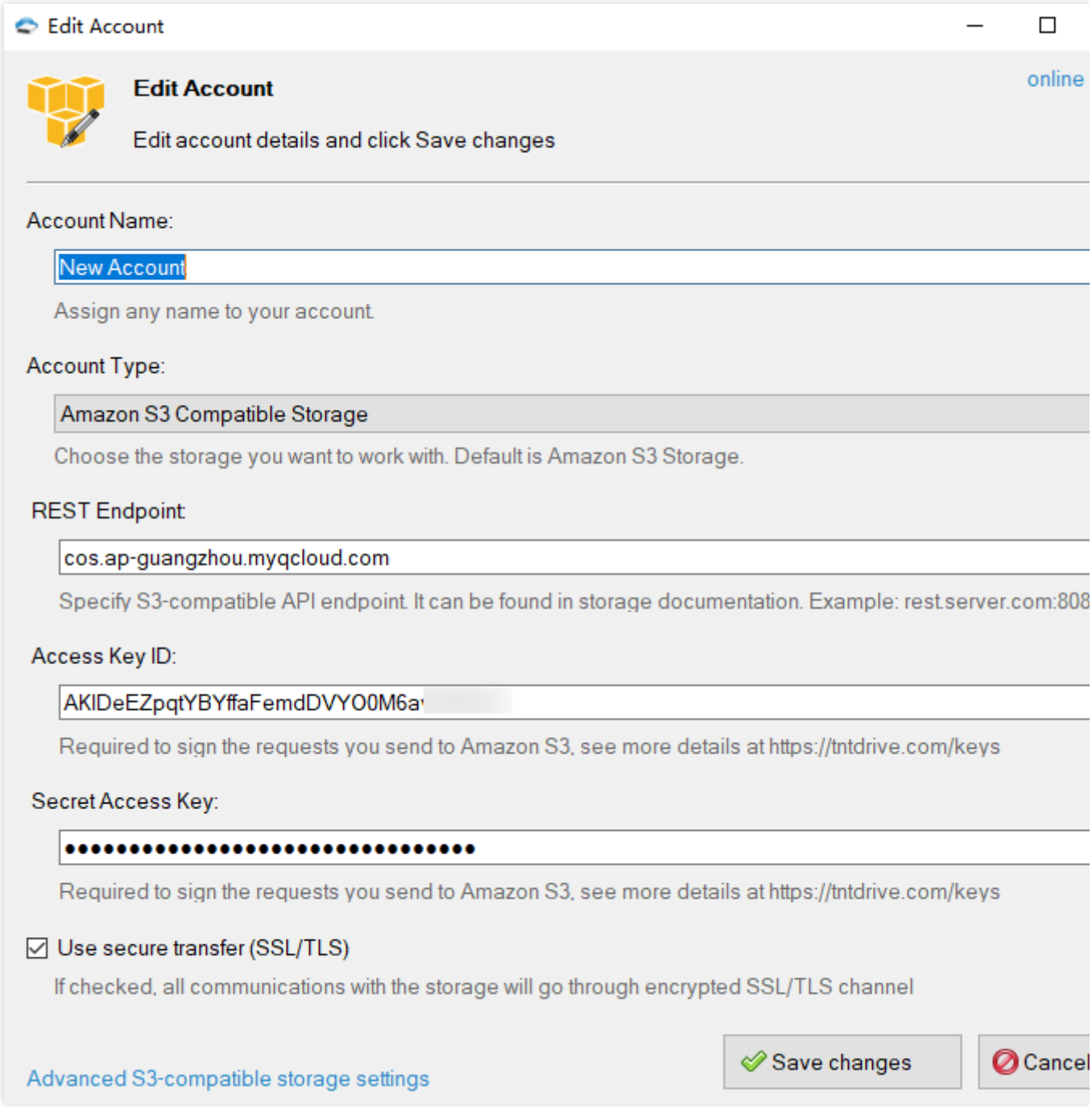
説明：

自動マウント設定後にサーバーを再起動します。通常はマウントが成功したことを確認できるまでに十数秒かかります。

関連する操作

また、サードパーティの商用有償ツールを使用することで、COSをWindowsサーバーにマウントし、ローカルディスクとしてマッピングすることもできます。次の操作では、TntDriveツールを例として取り上げます。

1. TntDriveをダウンロードし、インストールします。
2. TntDriveを開き、**Account > Add New Account**をクリックしてユーザーアカウントを作成します。



主なパラメータ情報は下記の通りです：

Account Name：カスタムアカウント名です。

Account Type：COSはS3互換であるため、ここで**Amazon S3 Compatible Storage**を選択できます。

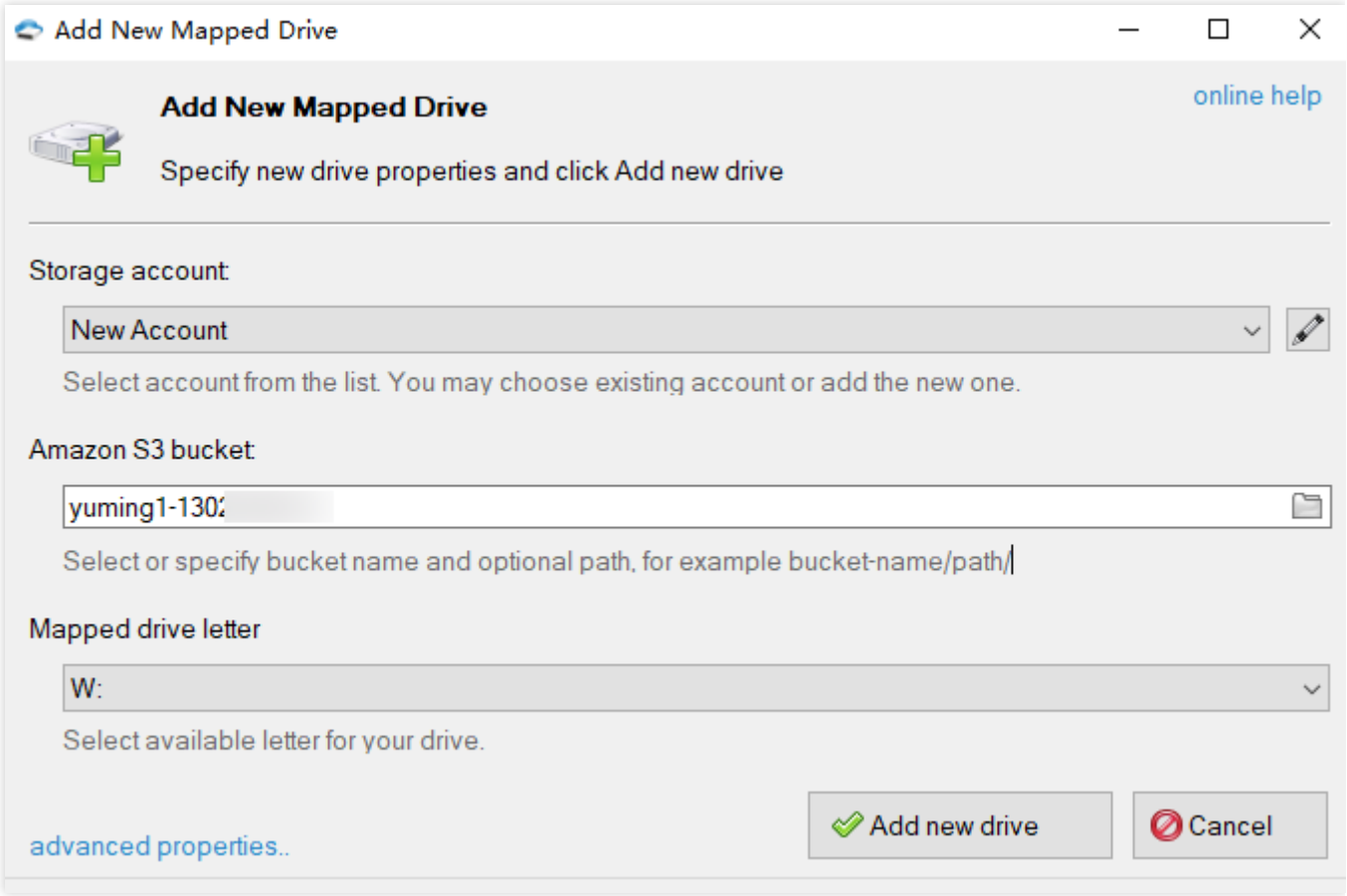
REST Endpoint：バケットのあるリージョンを入力します。例えば、バケットが広州にある場合は、cos.ap-guangzhou.myqcloud.comと入力します。

Access Key ID：SecretIdを入力します。APIキー管理ページで作成し、取得することができます。

Secret Access Key：SecretKeyを入力します。

3. **Add new account**をクリックします。

4. TntDriveインターフェースで、**Add New Mapped Drives**をクリックし、Mapped Drivesを作成します。



主なパラメータ情報は下記の通りです：

Amazon S3 Bucket：バケットのパスを入力するか、またはバケット名を選択します。右側のボタンをクリックすると、バケットを選択できます。これは、ステップ2で設定した広州リージョンにあるバケットを示しています（バケットはディスクに独立してマッピングされます）。

Mapped drives letter：ディスクのボリュームラベル名を設定します。ローカルのC、D、Eドライブなどと重複しないようにしてください。

5. 以上の情報を確認し、**Add new drive**をクリックします。

6. このディスクは、ローカルコンピュータの**マイコンピュータ**にあります。すべてのバケットをWindowsサーバーにマッピングしたい場合は、以上の手順を繰り返してください。

PicGo+Typora+COSでの画像ホスティングサービスの構築

最終更新日：：2024-06-26 10:47:37

概要

画像ホスティングサービスは、画像ストレージ、画像加工処理、画像のネットワーク全体への配信などの機能を提供します。世界中に無数にあるブログウェブサイトおよびコミュニティフォーラムに対し、バックエンドでの画像サービスのサポートを提供します。開発者は、Tencent Cloudの**Cloud Object Storage（COS）**を使用し、画像ホスティングサービスを構築することができます。COSは、Tencent Cloudが提供する大量のファイルをストレージする分散型ストレージサービスです。これまでよりもさらに豊富な機能、優れた性能、信頼性の高い保障を提供します。
COSを画像ホスティングシーンに用いるメリットは、以下のとおりです。
低コスト：ストレージ単価が低く、従量課金のため、使用した分のみのご請求となります。
速度制限無し：アップロード、ダウンロードに速度制限がないため、長い時間をかけてloadingを待つ必要がありません。アクセス品質もさらに良くなりました。
高可用性：高レベルのSLA可用性保障を有しています。ストレージのデータは99.9999999999%もの耐久性を保障します。
容量無制限：ファイル分散型ストレージで、大量のファイルをサポートします。必要に応じた容量を使用します。

実践シーン

シーン1：画像追加の際にCOSを使用し、画像ホスティングサービスを構築する

このシーンで使用するのは、以下のツールです。
PicGo：複数のクラウドストレージ設定、クイック画像リンク作成をサポートするツールです。
Typora：軽量級のMarkdownエディタです。複数の出力形式をサポートし、ローカルの画像をワンクリックで画像ホスティングへアップロードします。

操作手順

1. PicGoをインストールし、Tencent Cloud COSサービス関連のパラメータを設定します。
説明：
今回の実践で使用するのはPicGo 2.3.1バージョンです。その他のバージョンでは設定手順に若干の違いがあるため、適宜調整してください。

PicGo公式サイト でPicGoをダウンロードしてからインストールし、画像ホスティングで**Tencent Cloud COS**を見つけ、以下の関連パラメータ項目を入力します。
COSバージョン：COS v5を選択します。
SecretIdの設定：開発者は所有するプロジェクト身分識別IDを、身分認証に用います。APIキー管理ページで作成し、取得することができます。
SecretKeyの設定：開発者は所有するプロジェクト身分キーを、APIキー管理ページで取得することができます。
Bucketの設定：バケットは、COSでデータをストレージするコンテナとして使用されます。バケットの詳細については、バケット概要ドキュメントをご参照ください。
AppIdの設定：開発者はCOSにアクセスする際に所有するユーザーの次元における唯一のリソース識別として、識別リソースを用います。APIキー管理ページで取得することができます。
ストレージリージョンの設定：バケットの所属リージョンの情報です。列挙値については、アベイラビリティリージョンドキュメントをご参照ください。例えばap-beijing、ap-hongkong、eu-frankfurtなどです。
ストレージパスの設定：画像をCOSバケットに格納する際のパスです。
カスタムドメイン名の設定：選択可能です。上方のストレージ容量にカスタムオリジンサーバードメイン名を設定する場合、入力が可能です。説明についてはカスタムオリジンサーバードメイン名の有効化をご参照ください。
URLサフィックスの設定：URLのサフィックスにCOSデータ処理パラメータを追加することで、画像の圧縮、トリミング、形式変換などの操作を行うことができます。関連の説明については画像処理をご参照ください。
2. typoraを設定します（オプション）。
説明：
編集ニーズがMarkdownシーンではない場合は、このステップを無視してかまいません。前のステップでインストールしたPicGoのみを画像ホスティングツールとして使用します。
設定ガイドは以下のとおりです。
1. typora上の好みの設定の**画像**について、以下の設定を行います。
画像挿入時にて、**画像アップロード**を選択します。
アップロードサービスの設定にて、**PicGo(app)**を選択し、先ほどインストールしたPicGo.exeの場所を設定します。
2. typoraを再起動し、設定を有効化します。
3. typoraエディタエリアに移動し、直接画像をドラッグあるいはペーストすれば、画像がアップロードされ、COSファイルのリンクに自動で置き換えることができます。（ペースト後に、自動でCOSのリンクに置き換わらない場合は、PicGoのserver設定が開いているかどうかをチェックしてください）。

シーン2：元の画像ホスティングリポジトリの画像を迅速にTencent Cloud COSに移行する

ある画像ホスティングサービスを例にすると、ローカル画像ホスティングフォルダを探すか、オンライン上から完全なフォルダをダウンロードして、フォルダ内のすべての画像をCOSバケットに保存することができます。最後にリンクドメイン名を統一して変換すれば、サイトを元に戻すことができます。

操作手順

ステップ1：画像ホスティングサービスの画像をダウンロードする

元の画像ホスティングウェブサイトにログインし、これまでのアップロード済み画像フォルダをダウンロードします。

ステップ2：COSバケットの作成およびリンク不正アクセス防止の設定

1.Tencent Cloudアカウントの登録をし、アクセス権限が**パブリック読み取り・プライベート書き込み**のバケットを作成します。操作ガイドはバケットの作成をご参照ください。
2.バケット作成後、バケットのリンク不正アクセス防止設定を開き、画像が不正使用されるのを防ぎます。操作ガイドはリンク不正アクセス防止の設定をご参照ください。

ステップ3：アップロードフォルダをバケットに保存

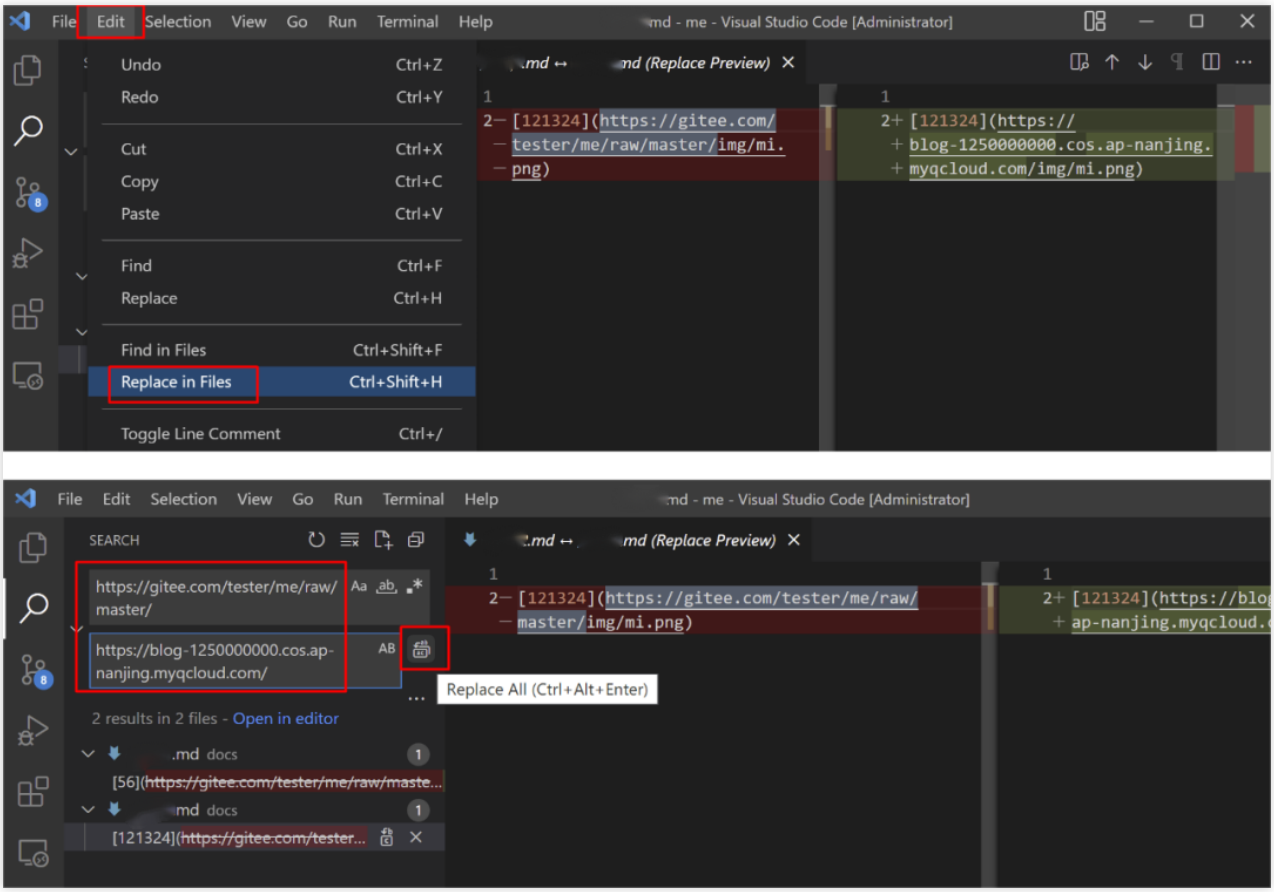
先ほど作成したバケットからアップロードフォルダをクリックし、先ほど準備した画像フォルダをCOSバケットにアップロードします。

説明：
画像数が多い場合は、COSBrowserクライアント を使用すれば、画像を迅速にアップロードできます。

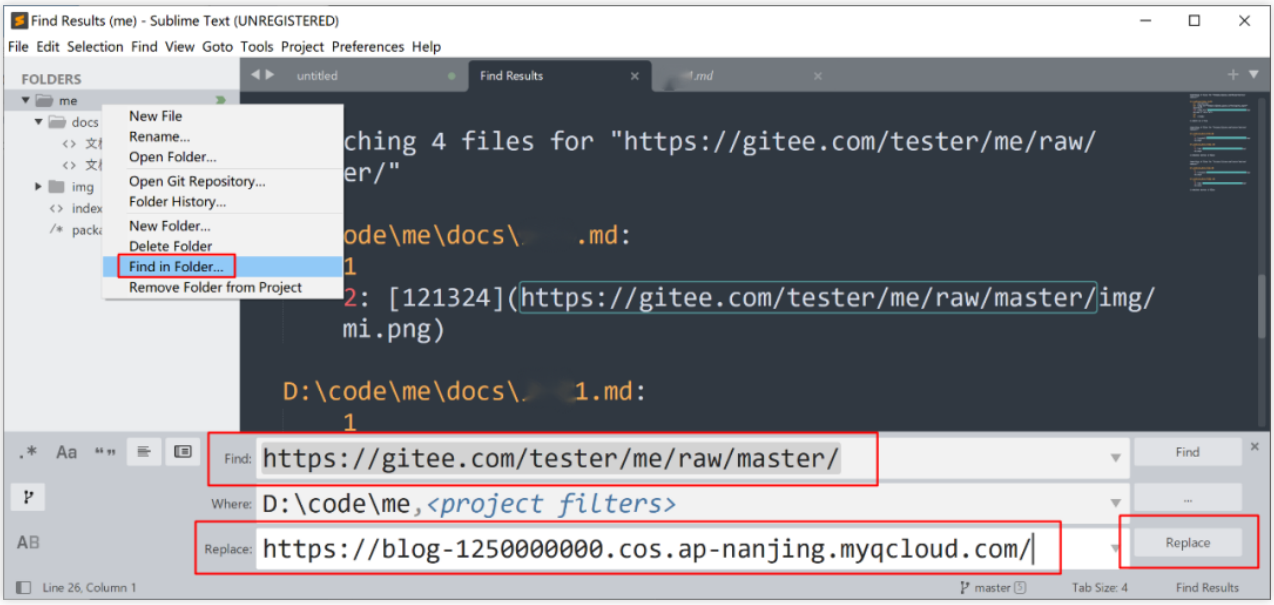
ステップ4：リンクドメイン名のグローバルな自動置き換え

COSコンソールバケットの概要ページに、バケットデフォルトドメン名をコピーします（カスタムCDNアクセラレーションドメイン名のバインドでも可）。共通コードエディタを使用し、プロジェクトグローバル検索置き換えに対して失効したリンクプレフィックスをCOSバケットのデフォルトドメイン名にします。

説明：
デフォルトドメイン名については、リージョンとアクセスドメイン名をご参照ください。
vscode検索置き換え事例：



sublime text検索置き換え事例：



CloudBerry ExplorerによってCOSリソースを管理

最終更新日：：2024-06-26 10:47:37

概要

CloudBerry Explorer はCloud Object Storage（COS）を管理するために使用されるクライアントツールです。CloudBerry ExplorerによってCOSをWindowsなどのOSにマウントすることができ、ユーザーのCOSファイルへのアクセス、移動および管理を容易にします。

サポートしているシステム

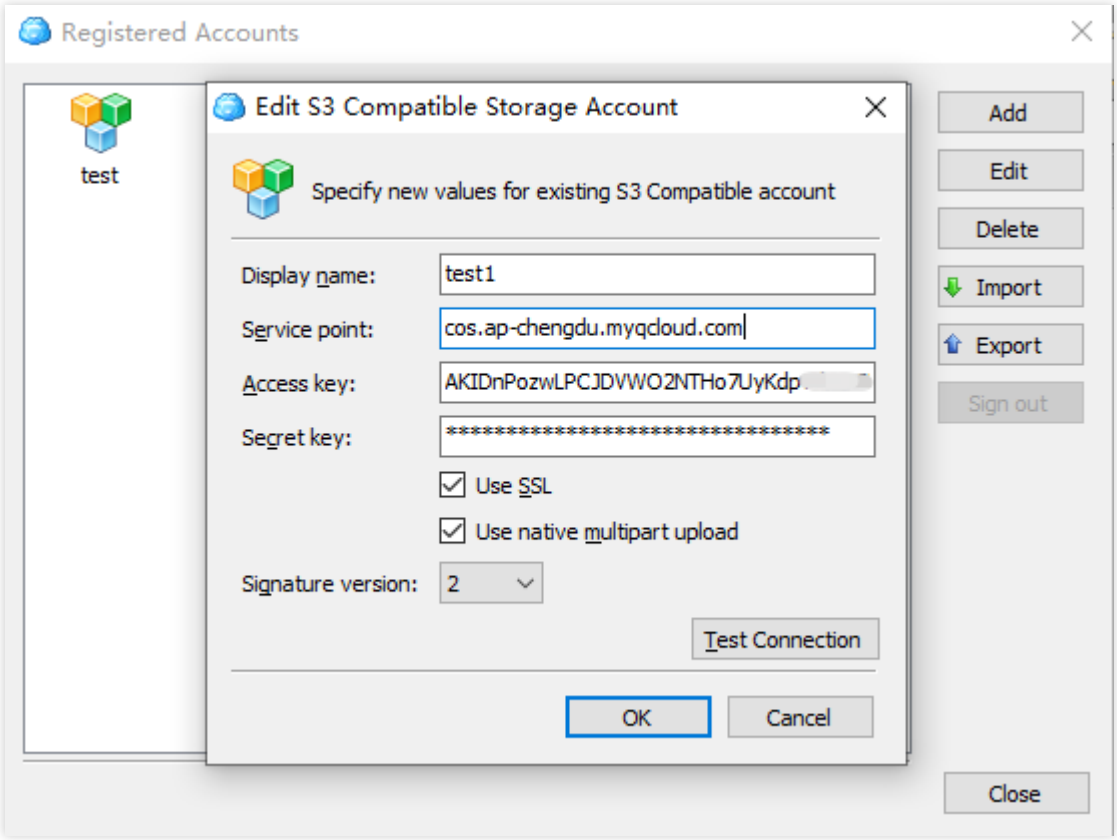
Windows、macOSシステムをサポートしています。

ダウンロードアドレス

CloudBerry 公式ダウンロードに進みます。

インストールと設定

注意：
以下の設定手順はCloudBerry Explorer Windows v6.3バージョンを例としています。その他のバージョンでは設定手順に若干の違いがあるため、適宜調整してください。
1. インストールパッケージをダブルクリックし、表示に従ってインストールを完了させます。
2. ツールを開き、S3 Compatibleを選択してダブルクリックします。
3. ポップアップウィンドウ内で以下の情報を設定し、Test Connectionをクリックして、接続に成功したことが表示されれば完了です。



設定項目の説明は次のとおりです。

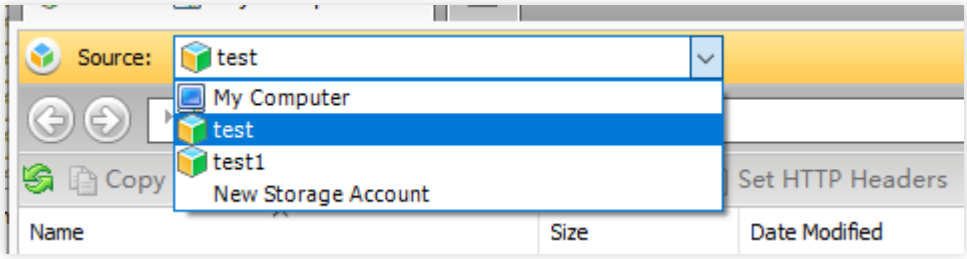
Display name：カスタムユーザー名を入力します。

Service point：形式が `cos.<Region>.myqcloud.com` で、例えば成都リージョンのバケットにアクセスする場合は、 `cos.ap-chengdu.myqcloud.com` と入力します。適用するリージョンの略称（region）については、リージョンとアクセスドメイン名をご参照ください。

Access key：アクセスキー情報SecretIdを入力し、Tencent Cloud APIキー に進んで作成と取得を行うことができます。

Secret key：アクセスキー情報Secretkeyを入力し、 Tencent Cloud APIキー に進んで作成と取得を行うことができます。

4. アカウント情報の追加が完了したら、Source内で以前設定したユーザー名を選択すると、そのユーザー名の下のバケットリストを確認することができ、これは設定が完了したことを意味します。



COSファイルの管理

バケットリストの照会

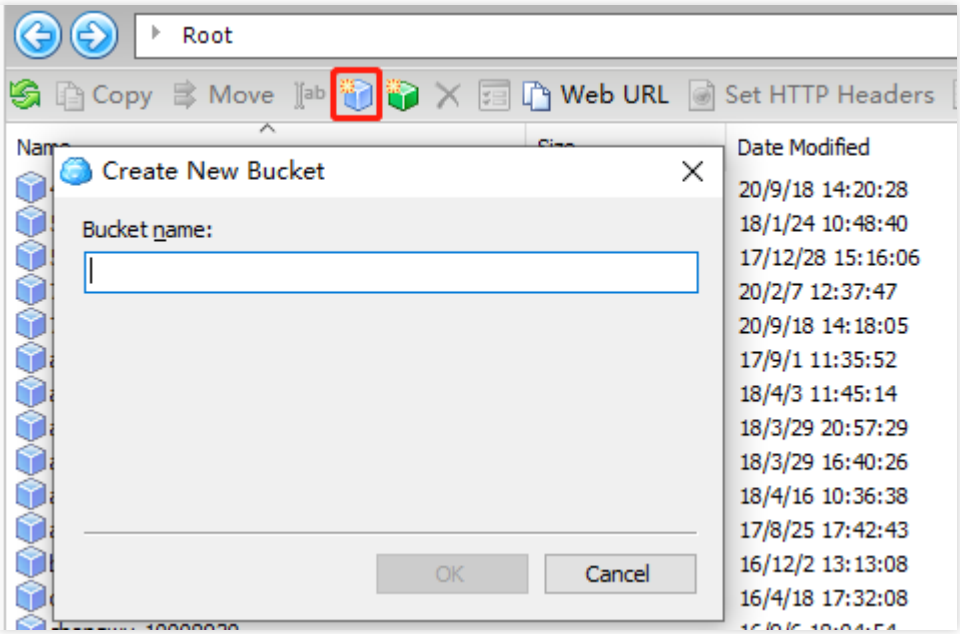
Source内で以前設定したユーザー名を選択すると、そのユーザー名下のバケットリストを確認することができます。

注意：

Service pointに設定されたリージョンに対応するバケットのみ確認できます。その他のリージョンのバケットを確認したい場合は、**File > Edit Accounts**をクリックして、ユーザー名を選択し、Service pointパラメータをその他のリージョンに修正すれば完了です。

バケットの作成

画像内のアイコンをクリックし、ポップアップウィンドウ内に完全なバケット名を入力します。例えばexamplebucket-1250000000です。入力情報に誤りがなければ、**OK**をクリックすると作成が完了します。
バケットの命名ルールについては、 バケット命名ルールをご参照ください。

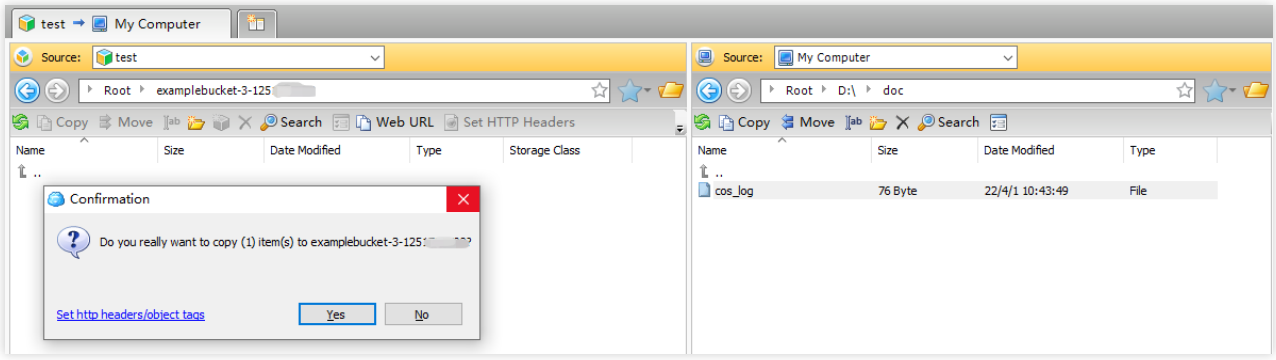


バケットの削除

バケットリスト内で削除する必要があるバケットを選択し、 **Delete**を右クリックすると削除することができます。

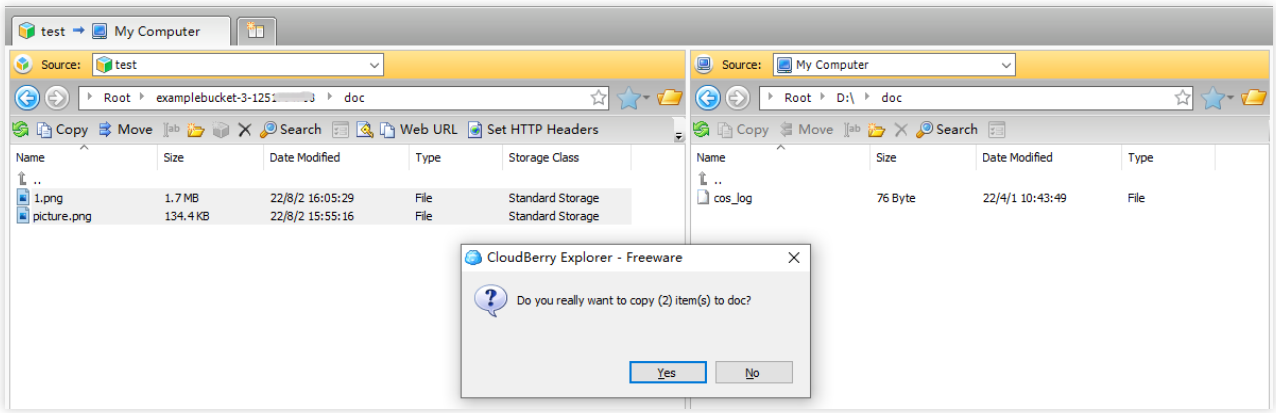
オブジェクトのアップロード

バケットリスト内でオブジェクトのアップロードを行う必要があるバケットまたはパスを選択し、その後ローカルコンピュータでアップロードする必要があるオブジェクトを選択し，其それを左側のウィンドウ内にドラッグすれば、アップロード操作が完了します。



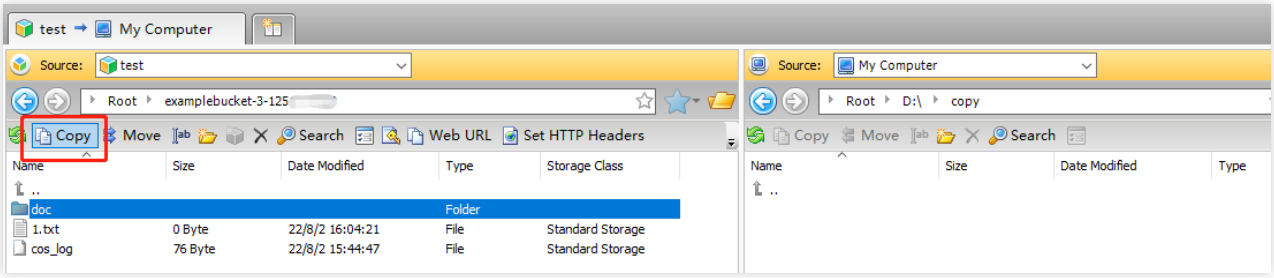
オブジェクトのダウンロード

左側のウィンドウ内でダウンロードする必要があるオブジェクトを選択し、次にオブジェクトを右側のローカルコンピュータのフォルダ内にドラッグすれば、ダウンロード操作が完了します。



オブジェクトのコピー

ツールの右側のウィンドウで、オブジェクトがコピーされた後のターゲットパスを選択し、その後左側のウィンドウ内でコピーする必要があるオブジェクトを選択し、**Copy**を右クリックし、ポップアップウィンドウの情報を確認すると、オブジェクトのコピー操作が完了します。



オブジェクトのリネーム

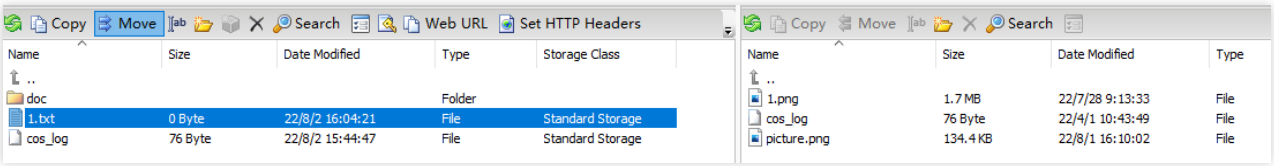
バケット内でリネームが必要なオブジェクトを探し、 **Rename**を右クリックして、新しい名称を入力すれば完了です。

オブジェクトの削除

バケット内で削除が必要なオブジェクトを選択し、 **Delete**を右クリックすれば、オブジェクトの削除が完了します。

オブジェクトの移動

ツールの右側のウィンドウで，オブジェクトが移動された後のターゲットパスを選択し、その後左側のウィンドウ内で移動する必要があるオブジェクトを選択し、**Move**を右クリックし、ポップアップウィンドウの情報を確認すれば、オブジェクトの移動操作が完了します。



その他の機能

以上の機能以外に、CloudBerry Explorerはさらに他の機能をサポートしています。例えばオブジェクトのACLの設定、オブジェクトのメタデータの確認、Headersのカスタマイズ、オブジェクトのURLの取得などです。ユーザーは実際のニーズに応じて操作を行うことができます。