Cloud Load Balancer

ベストプラクティス

製品ドキュメント

カタログ：

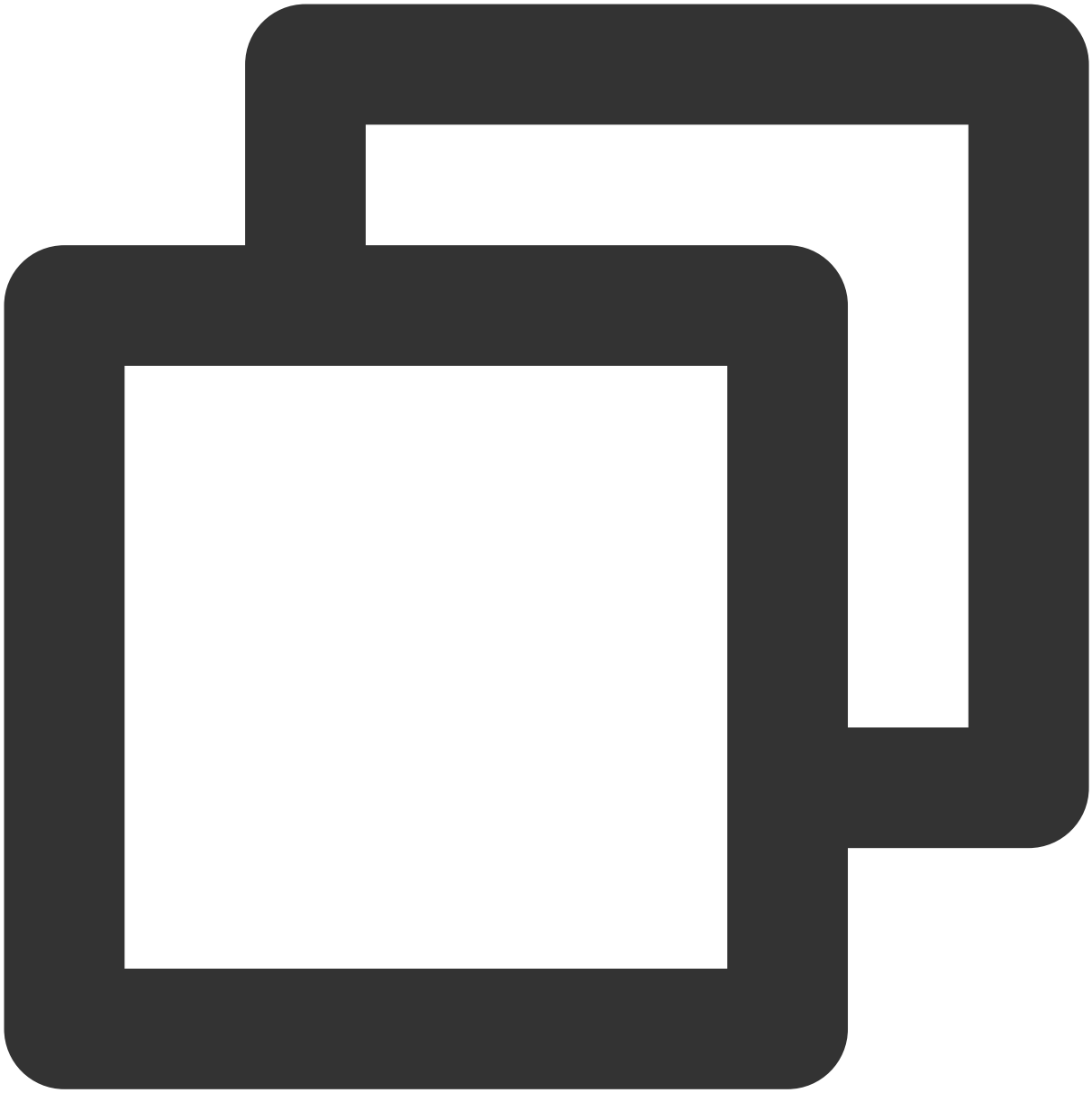
ベストプラクティス
CLBのGzip有効化設定およびチェック方法の説明

最終更新日：：2024-01-04 17:48:23

パブリックネットワークCLB、パブリックネットワーク固定IPタイプのCLBインスタンスでは、**HTTP/HTTPSプロトコル**はユーザーによるGzip圧縮機能の有効化をデフォルトでサポートしています。Gzip機能を有効化すると、ウェブページを圧縮することでネットワーク伝送のデータ量を有効に縮小し、クライアントブラウザのアクセス速度を向上させることができます。ご使用中は次の事項に注意する必要があります。

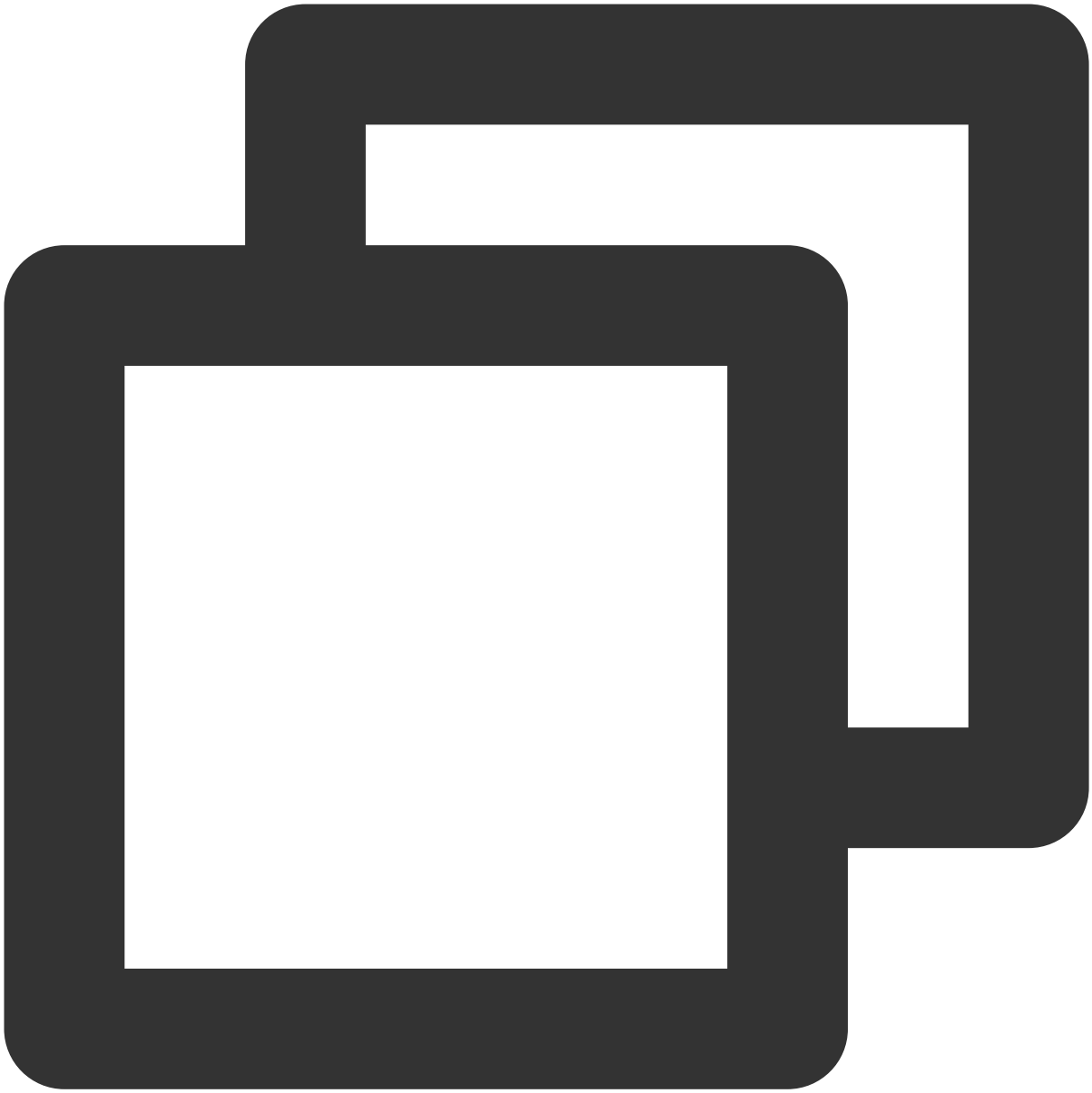
注意事項

バックエンドCVMでもGzipのサポートを同時に有効にしておく必要があります
一般的なNginxサービスコンテナでは、設定ファイル（デフォルトではnginx.conf）でGzipを有効化し、サービスを再起動する必要があります



```
gzip on;
```

現在CLBがサポートするファイルタイプは次のとおりです。Gzip_types設定項目でファイルタイプを指定して圧縮することができます。



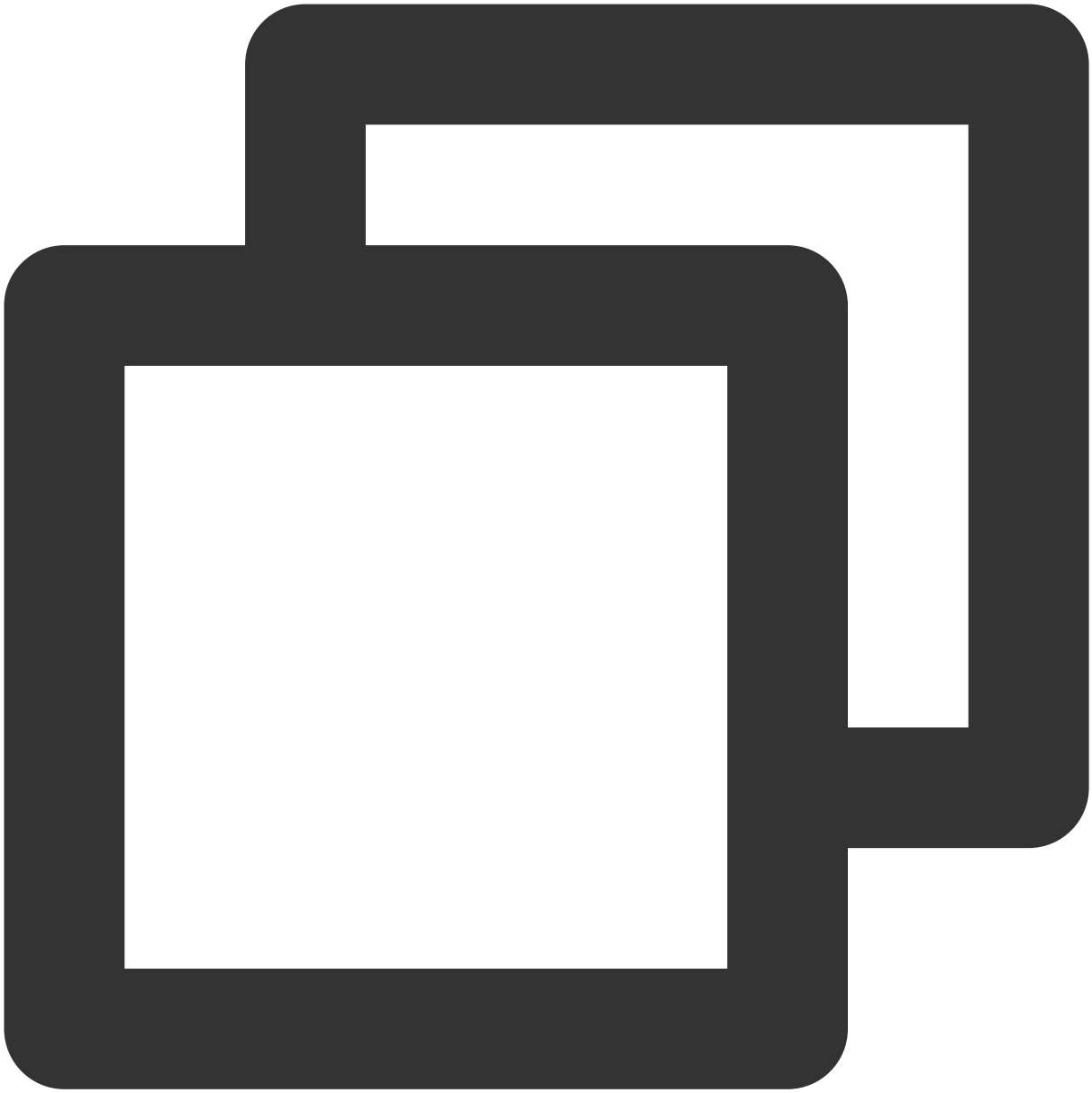
```
application/atom+xml application/javascript application/json application/rss+xml ap
```

ご注意：

CLBのバックエンドCVMの業務ソフトウェアでも、上記のファイルタイプのGzipサポートを同時に有効にしておく必要があります。

クライアントリクエストには圧縮リクエストであることを示す識別子を付ける必要があります

圧縮を有効にするには、さらにクライアントリクエストの際に次の識別子を付ける必要があります。

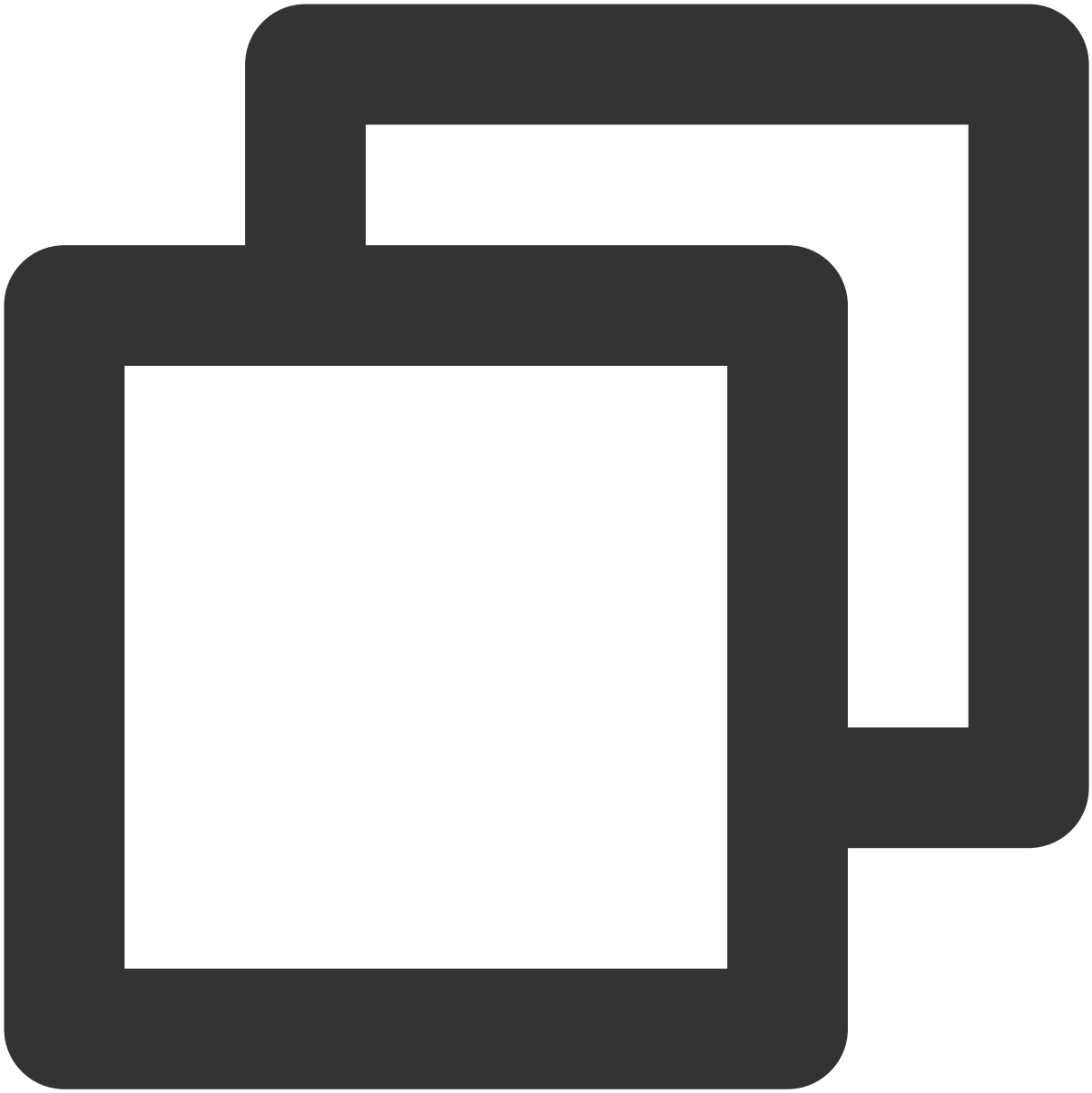


```
Accept-Encoding: gzip,deflate,sdch
```

バックエンドCVMでのGzip有効化フローの例

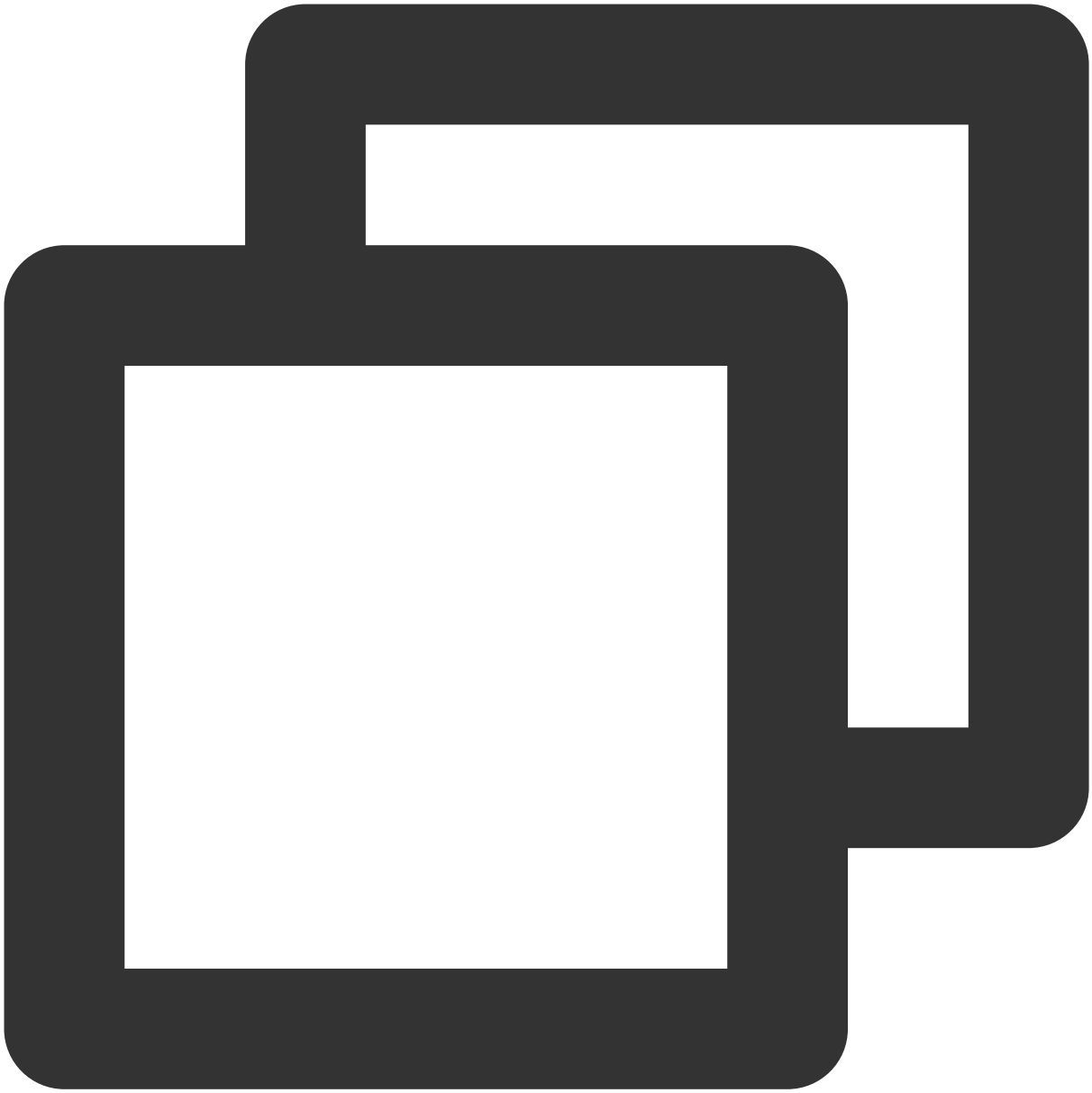
サンプルCVMの実行環境：Debian 6

1. vimを使用して、ユーザーパスを基にNginx設定ファイルを開きます。



```
vim /etc/nginx/nginx.conf
```

2. 次のコードを見つけます。



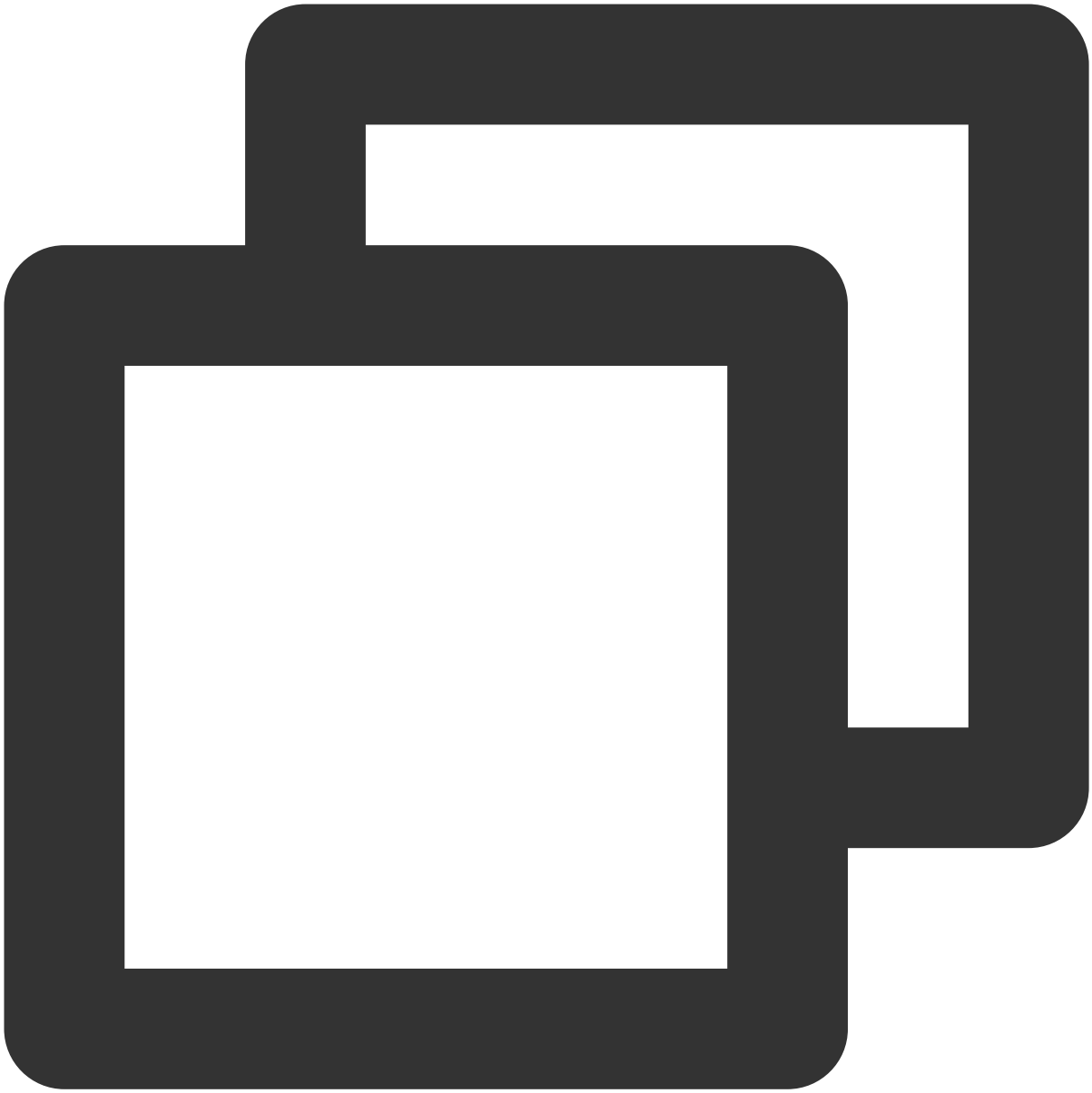
```
gzip on;
gzip_min_length 1k;
gzip_buffers 4 16k;
gzip_http_version 1.1;
gzip_comp_level 2;
gzip_types text/html application/json;
```

上記のコードの構文の説明は次のとおりです。

Gzip：Gzipモジュールを有効化または無効化します。

構文： `gzip on/off`

スコープ：http、server、location
gzip_min_length：圧縮を許可するページの最小バイト数を設定します。ページのバイト数はheaderのContent-Lengthから取得できます。デフォルト値は1kです。
構文： `gzip_min_length length`
スコープ：http、server、location
gzip_buffers：システムがGzipの圧縮結果のデータストリームを保存するために何単位のキャッシュを取得するかを設定します。16kは16kを単位とすることを表し、16kを単位としてオリジナルデータサイズの4倍のメモリを申請します。
構文： `gzip_buffers number size`
スコープ：http、server、location
gzip_http_version ：Gzipの機能を使用できるHTTPの最も低いバージョンを表します。HTTP/1.0を設定した場合、それがGzipの機能を使用できるHTTPの最も低いバージョンとなるため、GzipはHTTP/1.1への前方互換性を有します。Tencent Cloudは全ネットワークでHTTP/1.1をサポートしているため、変更の必要はありません。
構文: `gzip_http_version 1.0 | 1.1;`
スコープ: http、server、location
gzip_comp_level：Gzipの圧縮比であり、範囲は1～9です。1は圧縮比が最低で処理速度が最も速く、9は圧縮比が最高で処理が最も遅くなります（伝送速度は速くなりますがcpuを消費します）。
構文: `gzip_comp_level 1..9`
スコープ: http、server、location
gzip_types：マッチするMIMEタイプの圧縮を行います。デフォルトでは"text/html"タイプの圧縮が行われます。また、NginxのGzipはデフォルトではjavascript、画像などの静的リソースファイルの圧縮を行いません。gzip_typesで圧縮するMIMEタイプを指定することができ、設定した値以外のものは圧縮されません。**例えばjson形式のデータを圧縮したい場合は、このステートメントにapplication/jsonタイプのデータを追加する必要があります。**
サポートするタイプは次のとおりです。

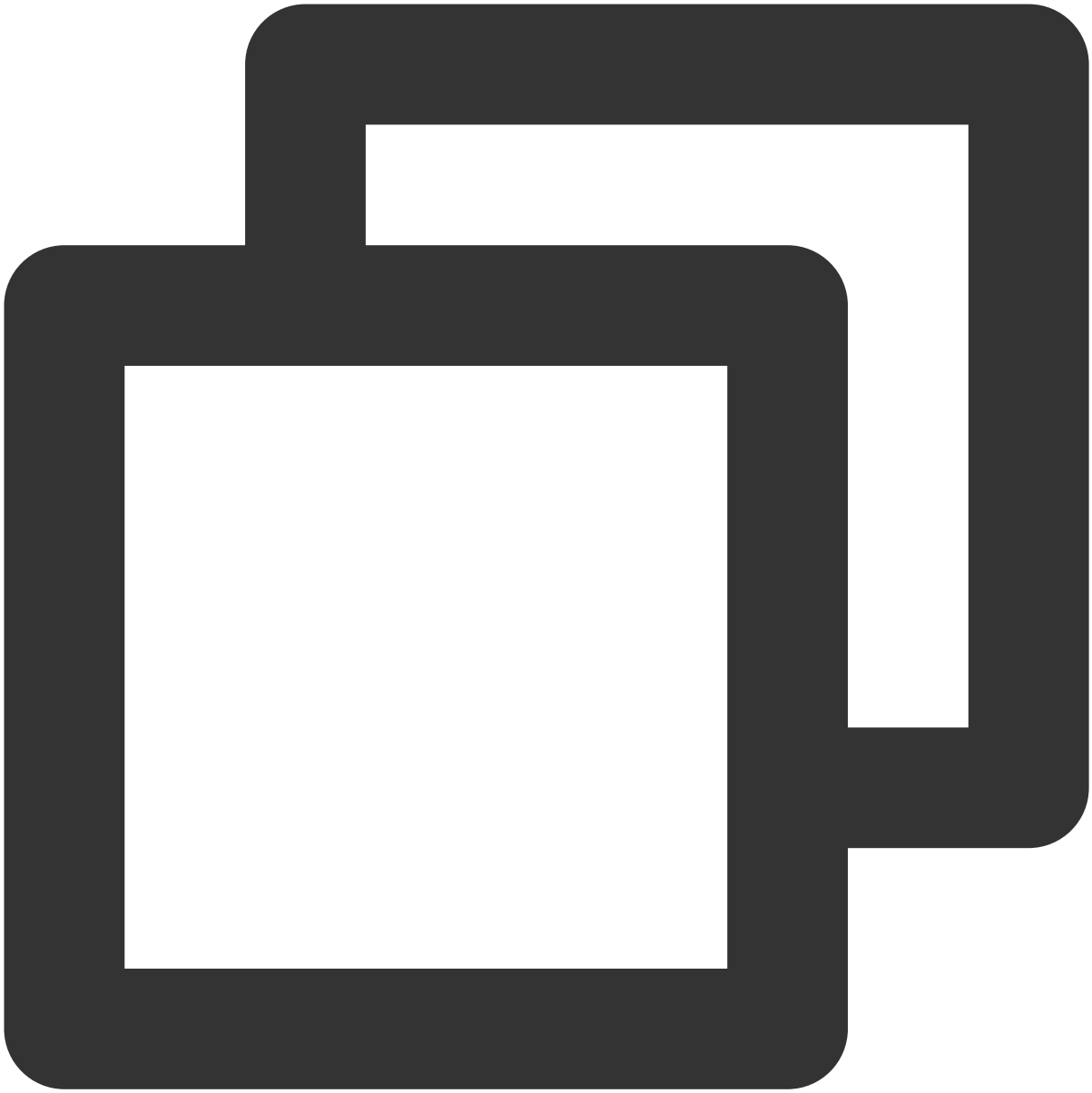


```
text/html text/plain text/css application/x-javascript text/javascript application/
```

構文: `gzip_types mime-type [mime-type ...]`

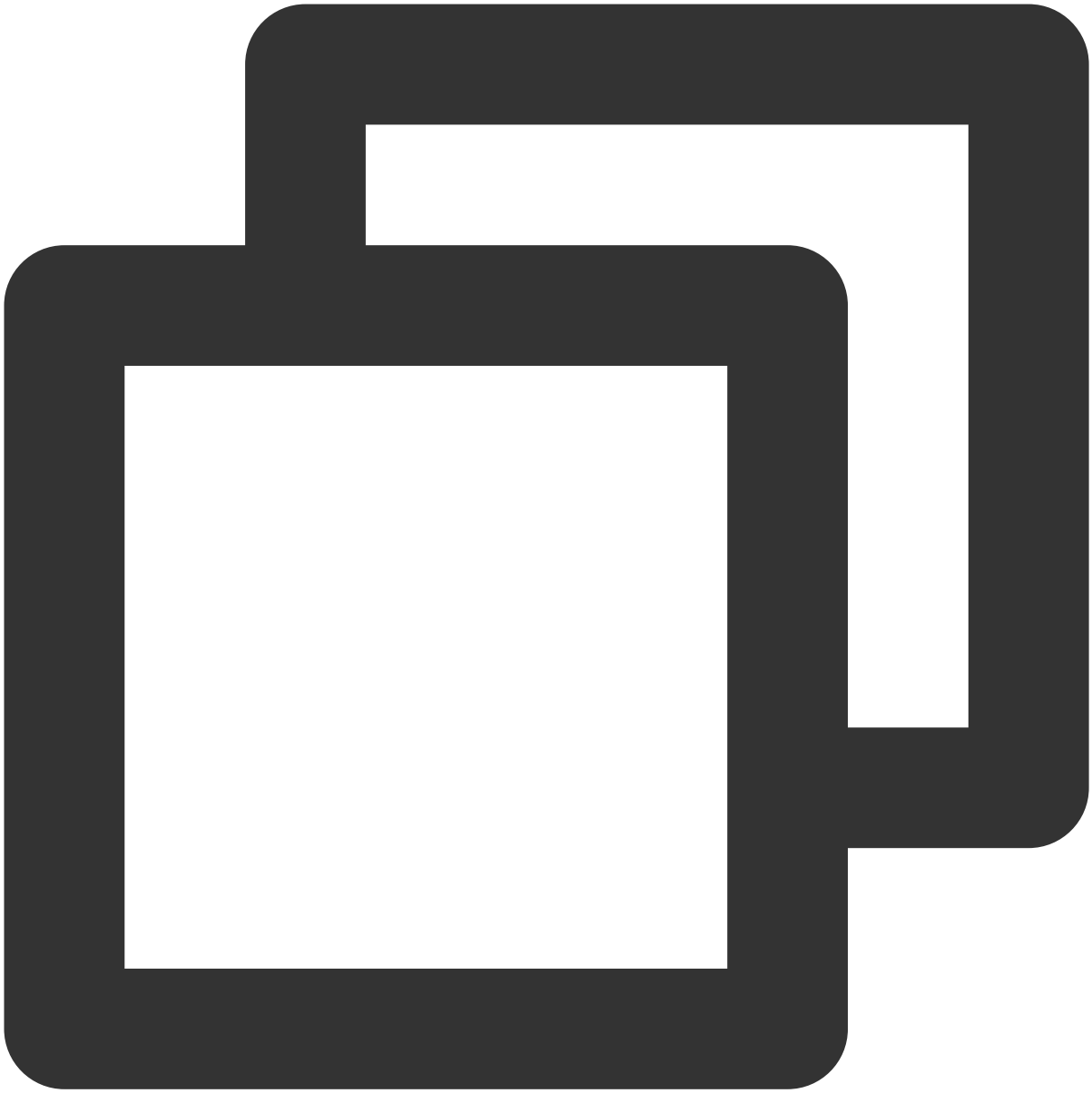
スコープ: http、server、location

3. 設定を変更する場合は、先にファイルを保存して終了し、Nginx binファイルディレクトリに進み、次のコマンドを実行してNginxをリロードします。



```
./nginx -s reload
```

4. 以下のcurlコマンドを実行してGzipの有効化に成功したかどうかをテストします。



```
curl -I -H "Accept-Encoding: gzip, deflate" "http://cloud.tencent.com/example/"
```

コマンド実行後に結果が戻る場合は、有効化に成功したことを表します。
コマンド実行後に結果が戻らない場合は、有効化に失敗したことを表します。

HTTPS転送設定スタートガイド

最終更新日：：2024-01-04 17:48:23

1. CLB機能の説明

Tencent Cloud CLBロードバランサは、プロトコルスタックおよびサーバーのハイレベルな最適化により、HTTPSパフォーマンスの大幅な向上を実現しました。またTencentは海外事業者との提携によって証明書にかかるコストを大きく削減しました。Tencent Cloud CLBは次のような点でお客様のビジネスに大きなメリットをもたらします。

1. HTTPSを使用してもClientのアクセス速度を低下させません。
2. クラスター内の単一サーバーのSSL暗号化復号パフォーマンスは6.5W cpsのフルハンドシェイクに達します。これはハイパフォーマンスCPUの少なくとも3.5倍アップに相当します。サーバーのコストを削減することで、業務運営およびトラフィック急増時のサービス機能を大幅に引き上げ、コンピューティング型の攻撃防御機能を増強します。
3. マルチプロトコルのオフロードと変換をサポートし、業務上でクライアントのさまざまなプロトコルに対応するための負荷を減らすことができます。業務バックエンドがHTTP1.1をサポートしていれば、 HTTP2、SPDY、SSL3.0、TLS1.2などの各バージョンのプロトコルを使用できます。
4. SSL証明書の申請、モニタリング、更新をワンストップで行えます。国際的な証明書認証局であるComodo、SecureSiteとの間で協議と提携を行い、証明書申請フローの大幅な短縮とコスト削減を実現しました。
5. CC攻撃防止およびWAF機能を有します。スロー攻撃、高頻度ターゲット攻撃、SQLインジェクション、トロイの木馬などのアプリケーション層攻撃を有効に排除します。

2. HTTP、HTTPSヘッダー識別子

CLBはHTTPSのプロキシとなります。クライアントからのHTTPまたはHTTPSリクエストが、CLBに到達してバックエンドサーバーに転送される時点で、CLBとバックエンドサービスの間のプロトコルにHTTP、HTTPSまたはgRPCを選択することをサポートしています。詳細については、HTTPSリスナーの設定をご参照ください。
CLBとバックエンドサービスの間のプロトコルにHTTPを選択した場合、開発者はフロントエンドのリクエストがHTTPかHTTPSかを識別できない可能性があります。
Tencent Cloud CLBはリクエストをバックエンドサーバーに転送する際、headerにX-Client-Protoを埋め込みます。
X-Client-Proto: http（フロントエンドはHTTPリクエスト）
X-Client-Proto: https（フロントエンドはHTTPSリクエスト）

3. クイック設定

ユーザーがウェブサイト `https://example.com` を設定したいとします。開発者は、ユーザーがブラウザにURLを入力する際、 `www.example.com` を直接入力するだけで、HTTPSプロトコルによってセキュアにアクセスできるようにしたいと考えています。

Cloud Load Balancer（CLB）の操作フローは以下のドキュメントをご参照ください。

CLBインスタンスの作成の例

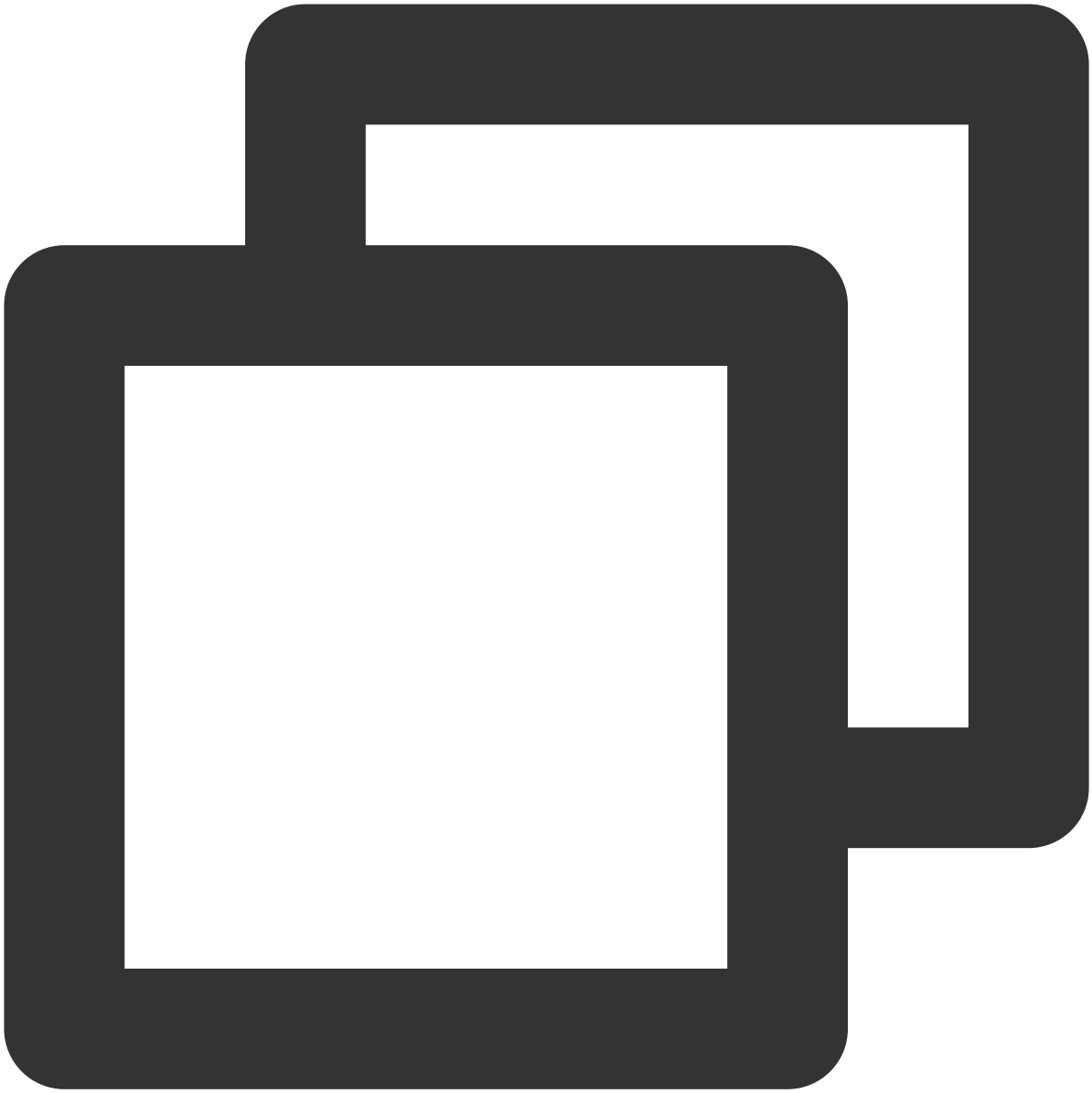
HTTPリスナーの設定

バックエンドサーバーの管理

ユーザーが入力した `www.example.com` リクエストの転送フローは次のようになります。

1. このリクエストはHTTPプロトコルで伝送され、VIPを介してCLBリスナーの80番ポートにアクセスし、バックエンドサーバーの8080番ポートに転送されます。
2. Tencent Cloudバックエンドサーバーの Nginx上でrewrite操作を設定することで、このリクエストは8080番ポートを経由し、 `https://example.com` ページにリライトされます。
3. このとき、ブラウザは `https://example.com` リクエストを対応するHTTPSサイトに再度送信します。このリクエストはVIPを介してCLBリスナーの443番ポートにアクセスし、バックエンドサーバーの80番ポートに転送されます。

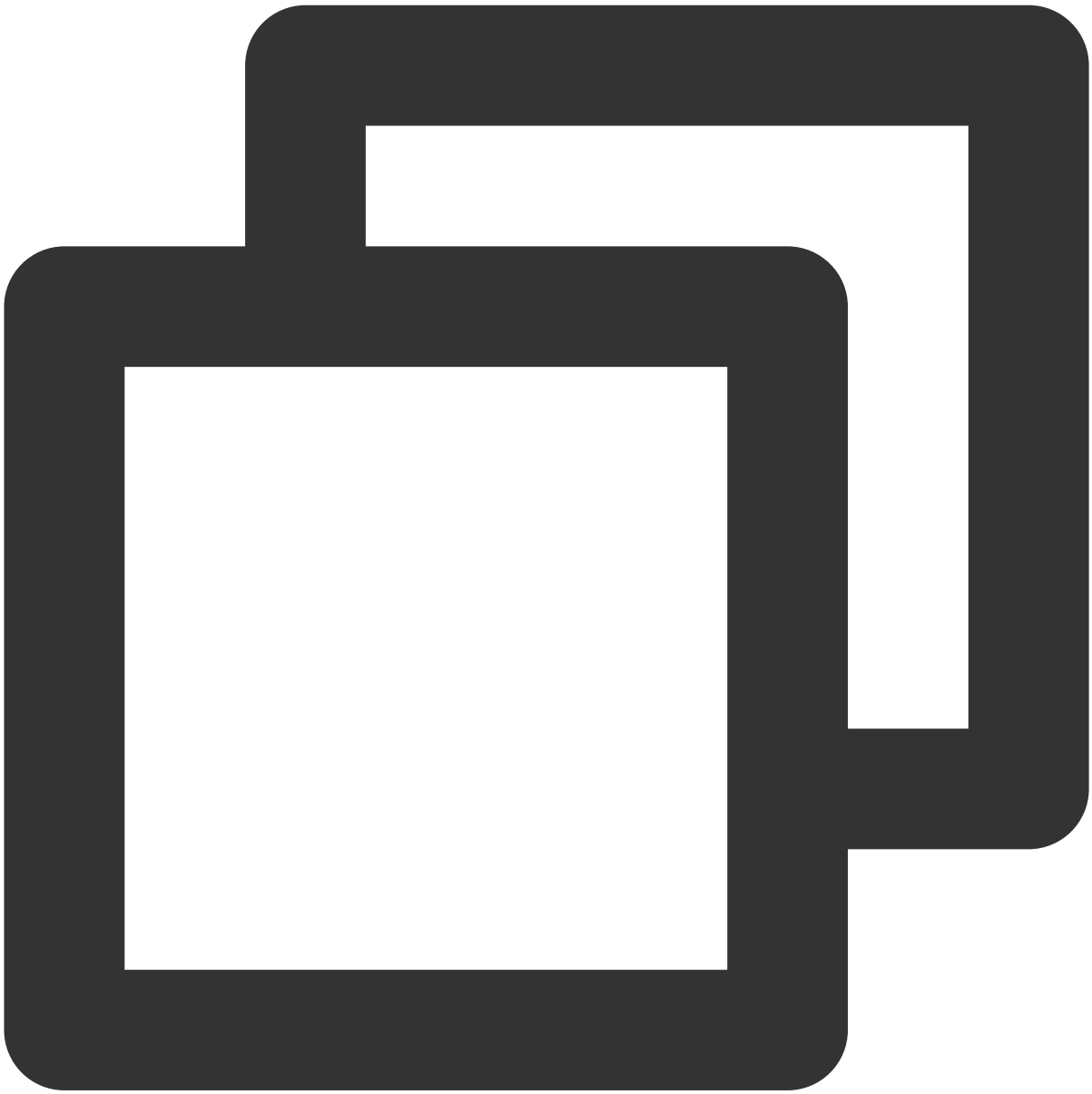
この操作はブラウザのユーザーに感知されずに、ユーザーのHTTPリクエストをより安全なHTTPリクエストに書き換えるものです。上記のリクエスト転送操作を実現するために、ユーザーはバックエンドサーバーを次のように設定することができます。



```
server {

    listen 8080;
    server_name example.qcloud.com;

    location / {

        #! customized_conf_begin;
        client_max_body_size 200m;
        rewrite ^/.(.*) https://$host/$1 redirect;
```

```
}
}
```

もしくはNginxの新バージョンで、推奨される301リダイレクト設定メソッドを用いて、NginxのHTTPページをHTTPSページにリダイレクトします。



```
server {
      listen       80;
      server_name    example.qcloud.com;
      return       301 https://$server_name$request_uri;
}
```

```
server {
      listen      443 ssl;
     server_name    example.qcloud.com;
    [....]
}
```

クライアントリアルIPの取得方法
IPv4 CLBのシーンでのクライアントリアルIPの取得

最終更新日：：2024-01-04 17:48:23

CLBでのクライアントリアルIPの取得に関する説明

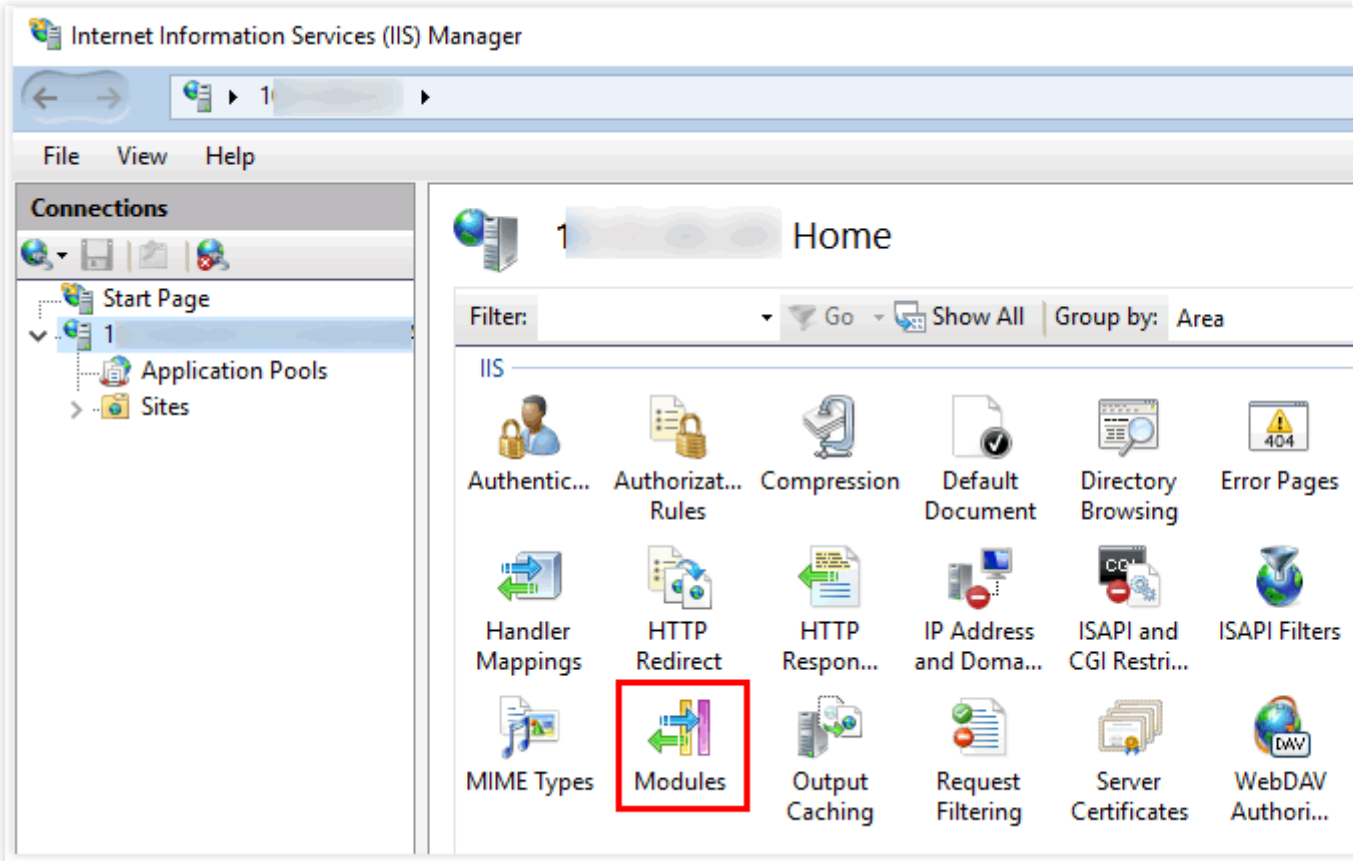
CLBのレイヤー4（TCP/UDP/TCP SSL）およびレイヤー7（HTTP/HTTPS）サービスでは、どちらもバックエンドCVM上でクライアントのリアルIPを直接取得することができ、追加設定を行う必要はありません。
レイヤー4CLBでは、バックエンドCVM上で取得するソースIPがクライアントIPです。
レイヤー7CLBでは、CLBとバックエンドサービス間の短い接続を使用する場合、バックエンドCVMで取得したソースIPがクライアントIPとなります。CLBとバックエンドサービス間の長い接続を使用する場合、CLBはソースIPをパススルーしなくなりますので、X-Forwarded-Forフィールドまたはremote_addrフィールドでクライアントIPを直接取得できます。レイヤー7CLBのアクセスログについては、アクセスログのCLSへの保存設定をご参照ください。
説明：
レイヤー4CLBでは、バックエンドCVM上で追加設定を行うことなくクライアントIPを取得できます。
SNATを行ったその他のレイヤー7ロードバランシングサービスでは、バックエンドCVM上での設定を行ってからX-Forwarded-Forメソッドを使用してクライアントのリアルIPを取得する必要があります。
次に、一般的なアプリケーションサーバー設定スキームについてご説明します。

IIS 6設定スキーム

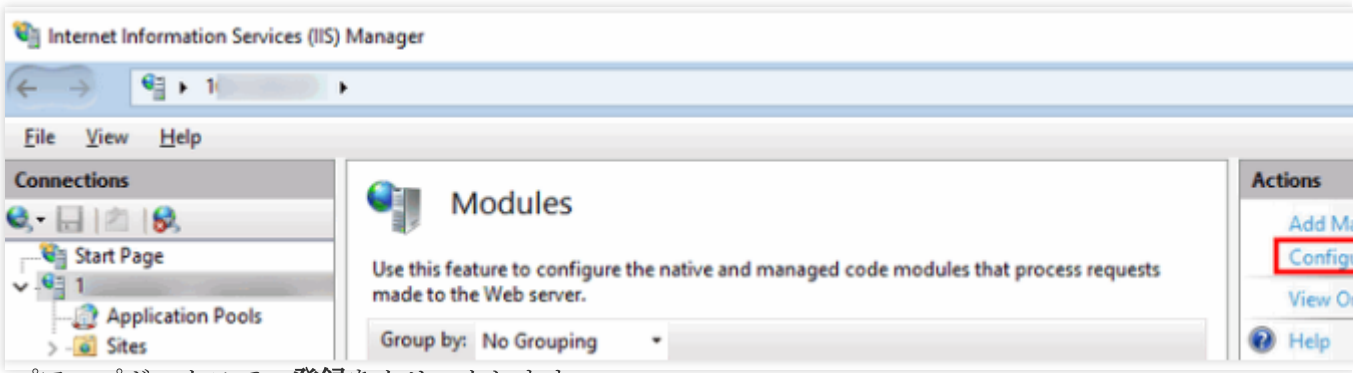
1. F5XForwardedForプラグインモジュールをダウンロードしてインストールし、ご自身のサーバーのOSバージョンに応じて `x86\\Release` または `x64\\Release` ディレクトリ下の `F5XForwardedFor.dll` をあるディレクトリにコピーします。ここでは仮に `C:\\ISAPIFilters` とします。同時にIISプロセスのこのディレクトリに対する読み取り権限を確保します。
2. IISマネージャーを開き、現在開いているウェブサイトを見つけ、このウェブサイト上で右クリックして**プロパティ**を選択し、プロパティページを開きます。
3. プロパティページで**ISAPIフィルター**に切り替え、**追加**をクリックすると、追加ウィンドウがポップアップします。
4. 追加ウィンドウの「フィルター名」に「F5XForwardedFor」、「実行可能ファイル」に `F5XForwardedFor.dll` の完全なパスをそれぞれ入力し、**OK**をクリックします。
5. IISサーバーを再起動し、設定が有効になるのを待ちます。

IIS 7設定スキーム

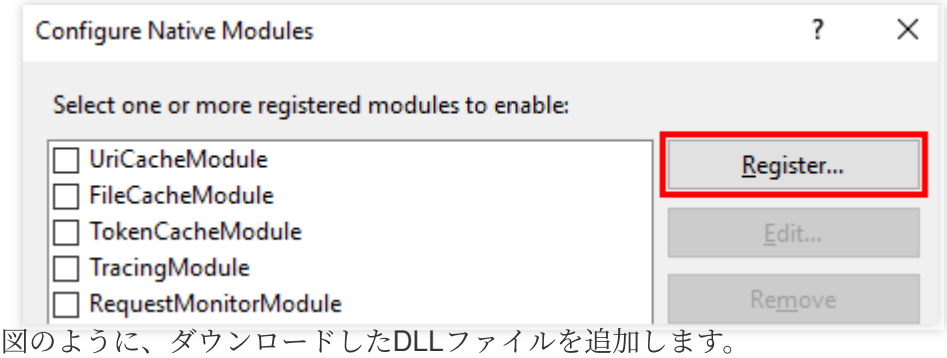
1. F5XForwardedForプラグインモジュールをダウンロードしてインストールし、ご自身のサーバーのOSバージョンに応じて `x86\\Release` または `x64\\Release` ディレクトリ下の `F5XFFHttpModule.dll` と `F5XFFHttpModule.ini` をあるディレクトリにコピーします。ここでは仮に `C:\\x_forwarded_for` とし、IISプロセスのこのディレクトリに対する読み取り権限を確保します。
2. **IISサーバー**を選択し、**モジュール**機能をダブルクリックします。



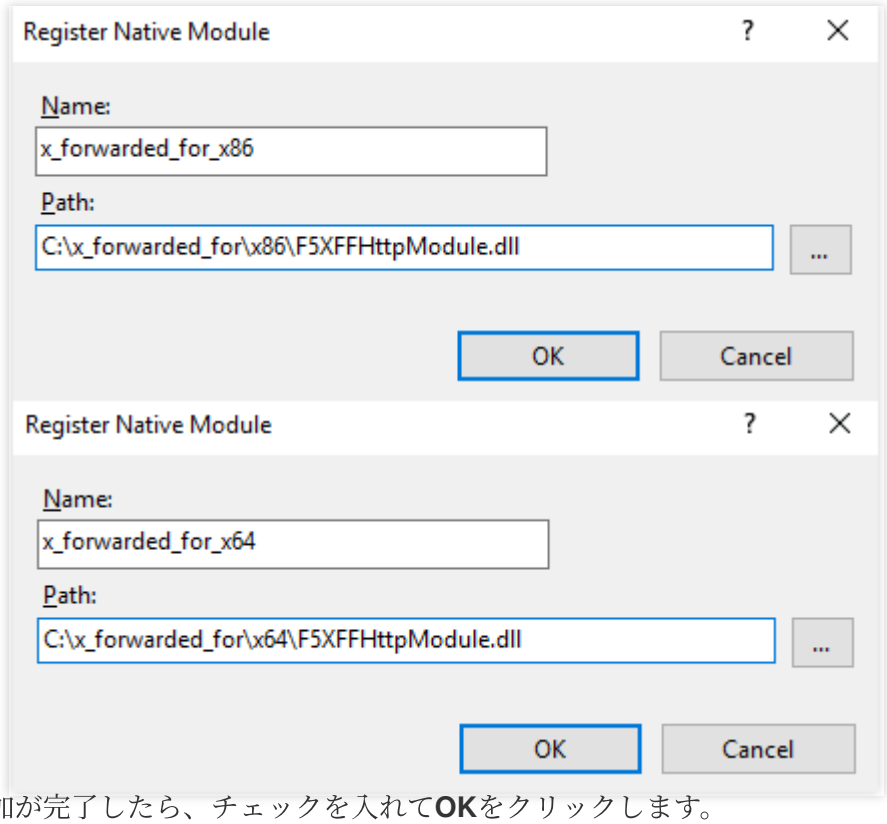
3. **ネイティブモジュールの設定**をクリックします。



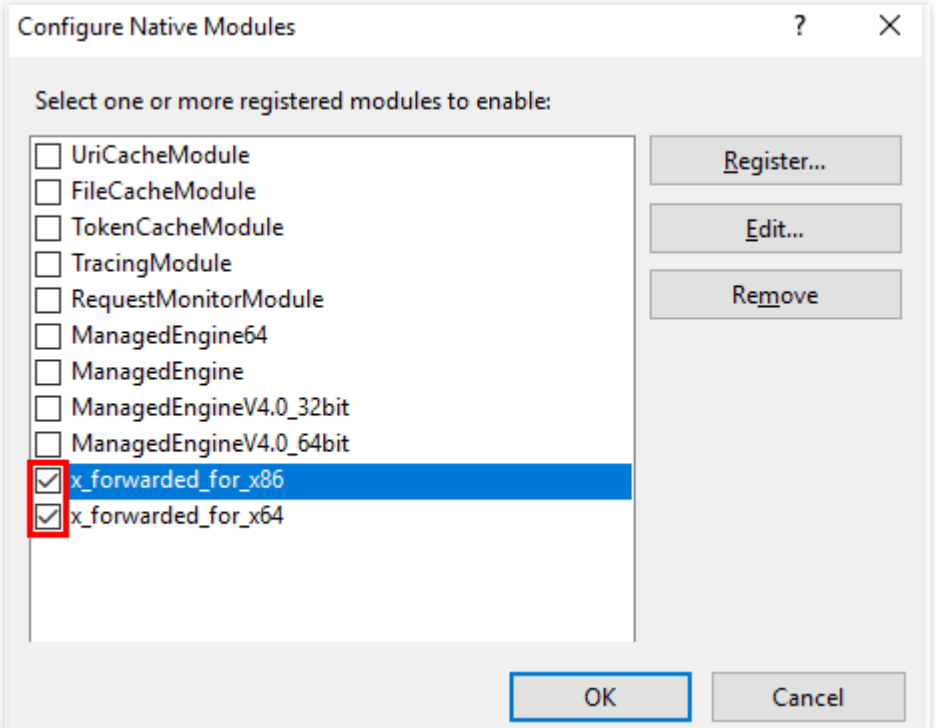
4. ポップアップボックスで、**登録**をクリックします。



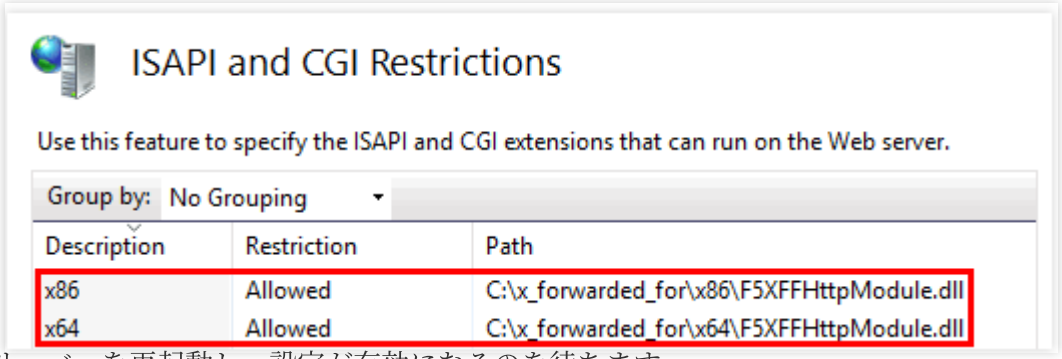
5. 下図のように、ダウンロードしたDLLファイルを追加します。



6. 追加が完了したら、チェックを入れて**OK**をクリックします。



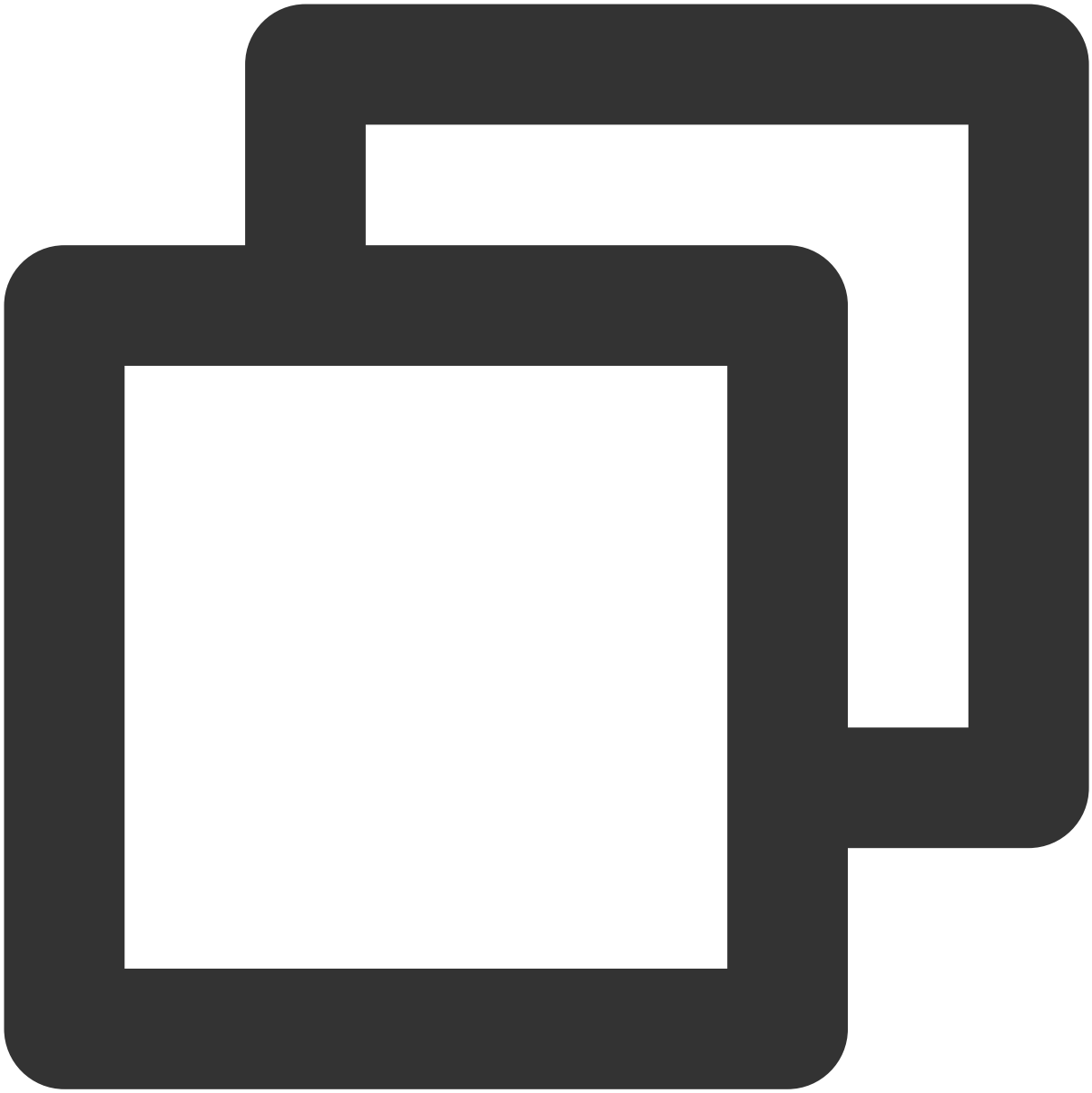
7. 「ISAPIおよびCGIの制限」に上記の2つのDLLファイルを追加し、制限を許可に設定します。



8. IISサーバーを再起動し、設定が有効になるのを待ちます。

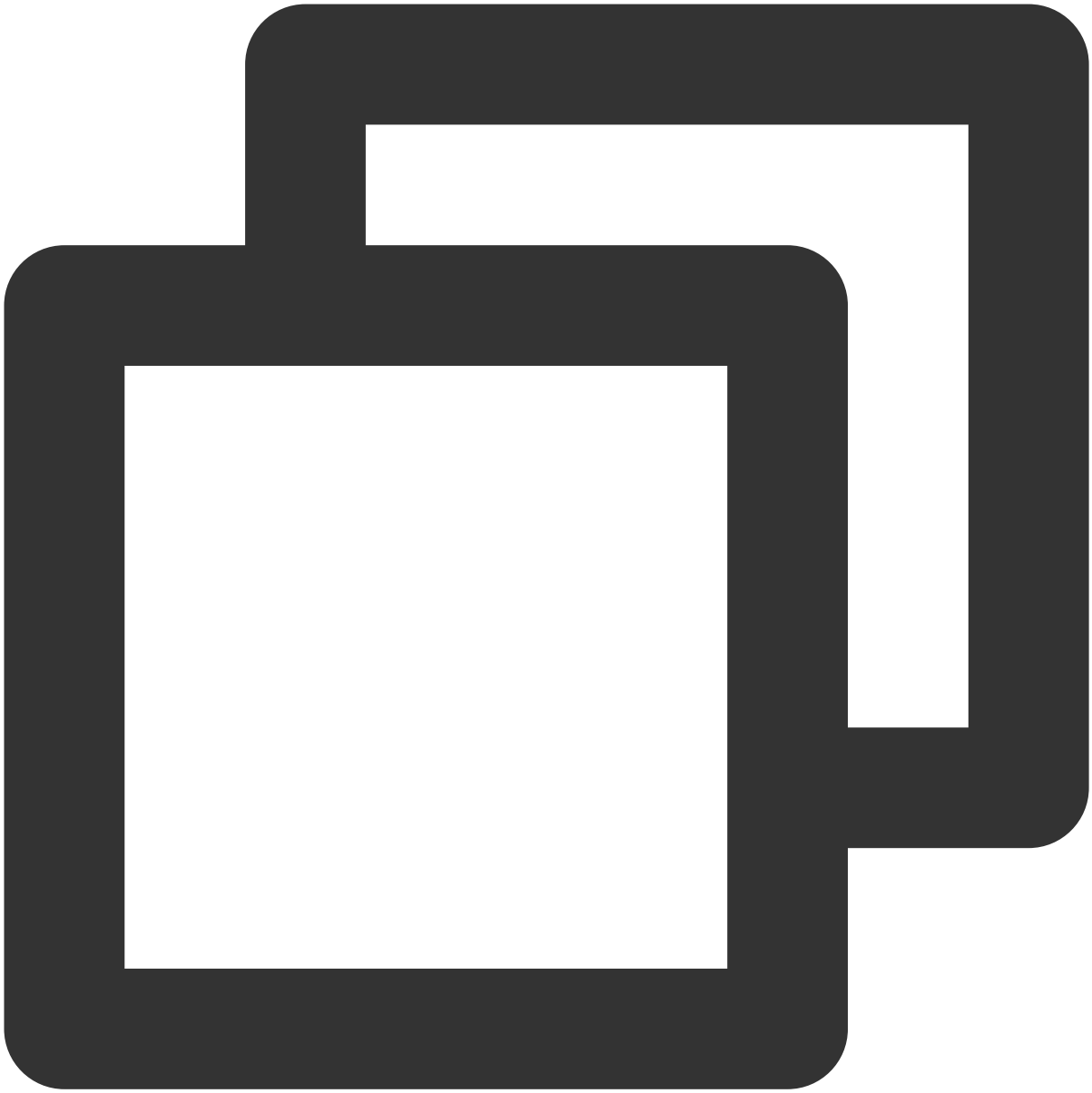
Apache設定スキーム

1. Apacheのサードパーティモジュール「mod_rpaf」をインストールします。



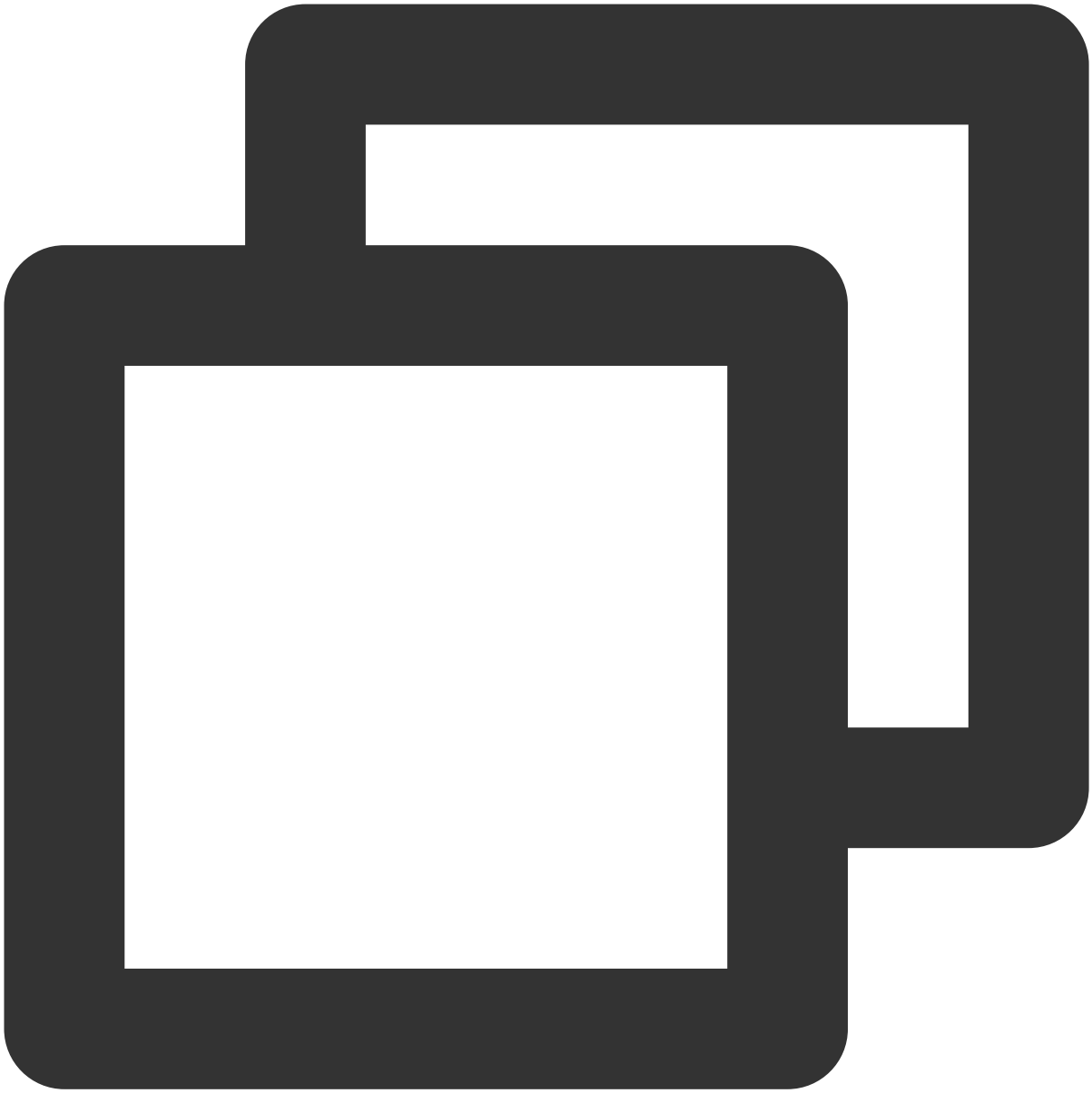
```
wget http://stderr.net/apache/rpaf/download/mod_rpaf-0.6.tar.gz
tar zxvf mod_rpaf-0.6.tar.gz
cd mod_rpaf-0.6
/usr/bin/apxs -i -c -n mod_rpaf-2.0.so mod_rpaf-2.0.c
```

2. Apacheの設定 `/etc/httpd/conf/httpd.conf` を変更し、末尾に次を追加します。



```
LoadModule rpaf_module modules/mod_rpaf-2.0.so
RPAFenable On
RPAFsethostname On
RPAFproxy_ips IPアドレス（このIPアドレスは最初はCLBが提供するパブリックIPではありません。具体的
RPAFheader X-Forwarded-For
```

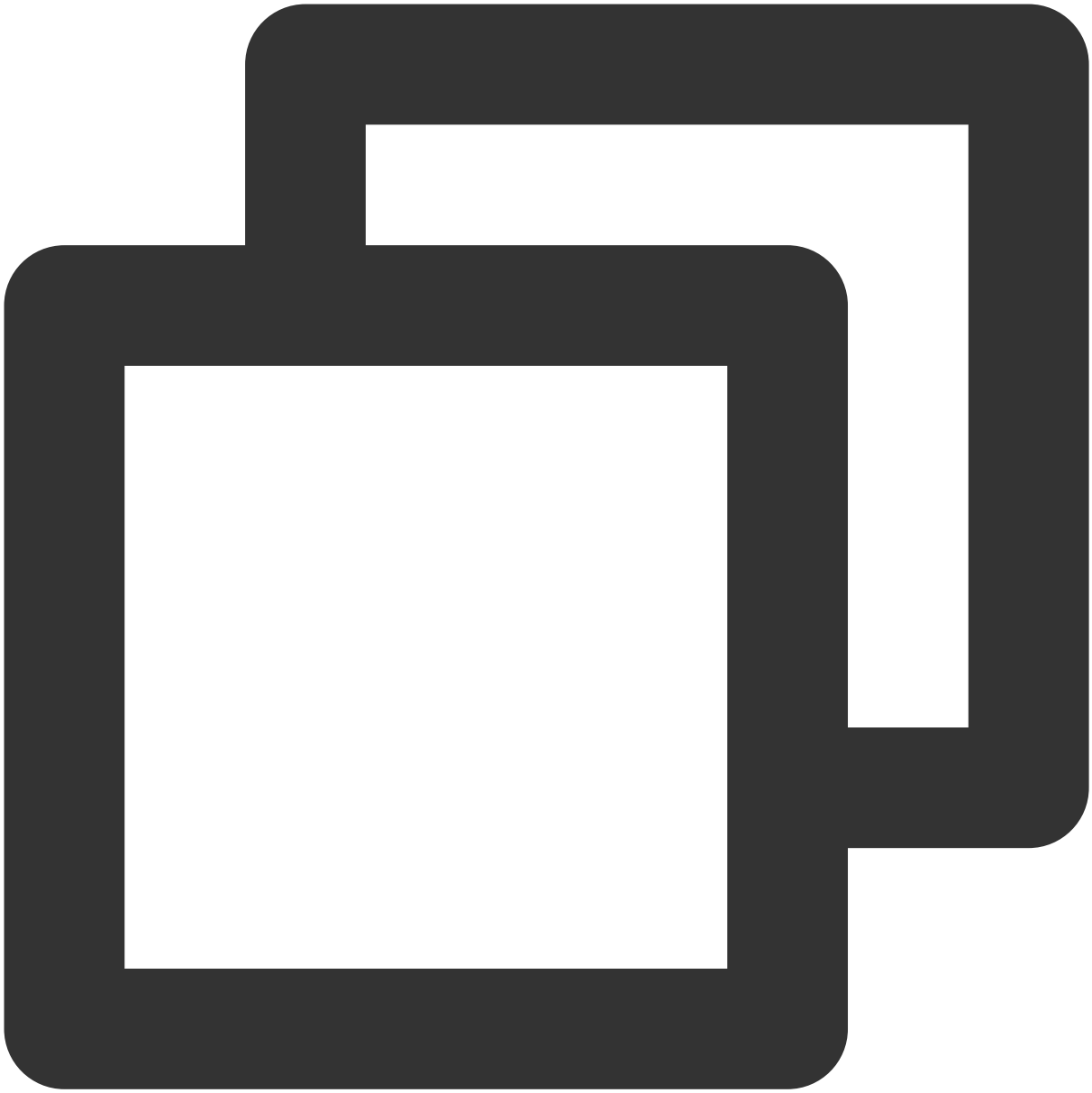
3. 追加が完了したら、Apacheを再起動します。



```
/usr/sbin/apachectl restart
```

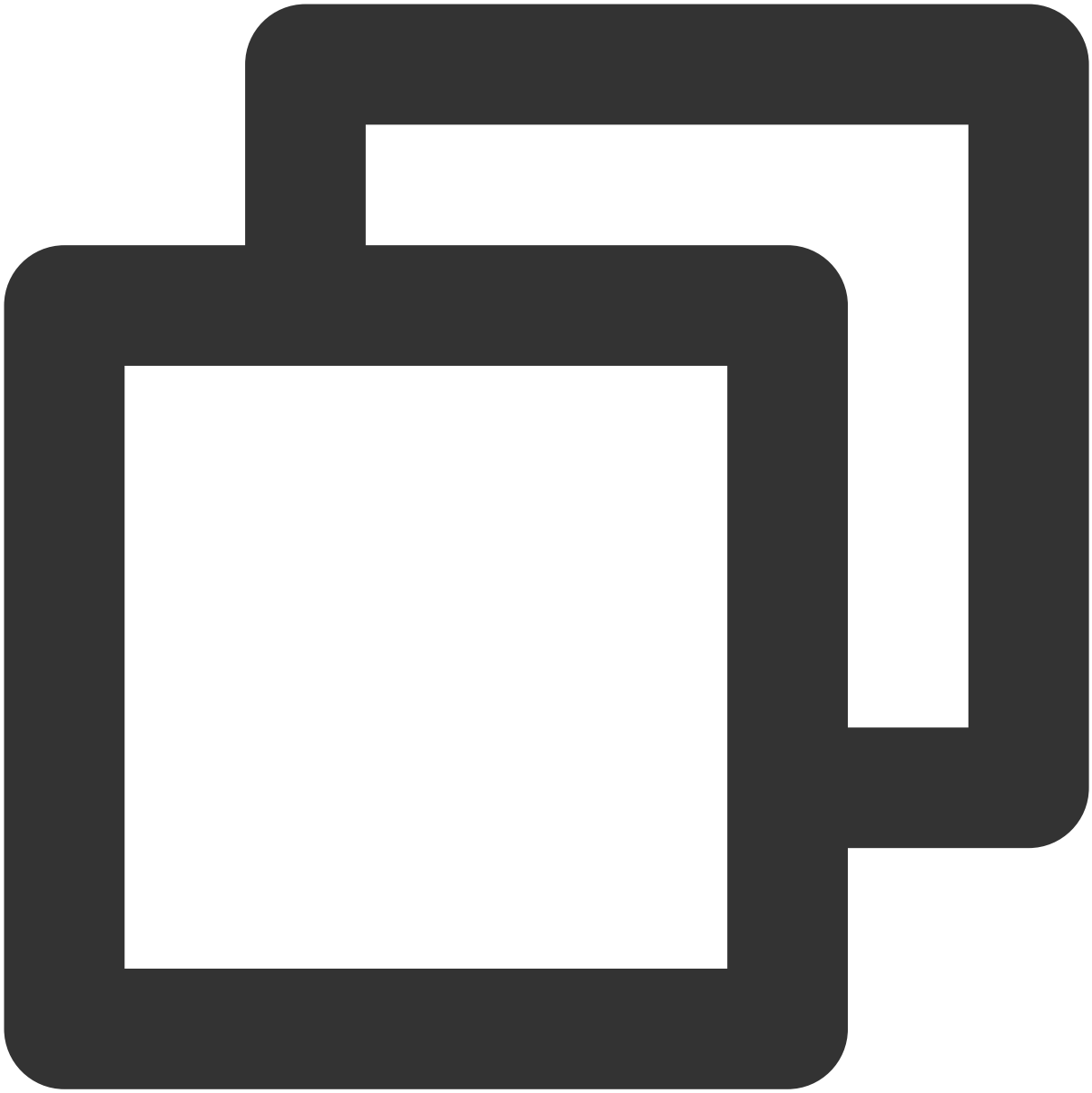
Nginx設定スキーム

1. Nginxをサーバーにする場合、クライアントのリアルIPを取得するにはhttp_realip_moduleを使用します。デフォルトでインストールされているNginxにはこのモジュールがインストールされていないため、Nginxを再コンパイルして `--with-http_realip_module` を追加する必要があります。



```
yum -y install gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel
wget http://nginx.org/download/nginx-1.17.0.tar.gz
tar zxvf nginx-1.17.0.tar.gz
cd nginx-1.17.0
./configure --prefix=/path/server/nginx --with-http_stub_status_module --without-ht
make
make install
```

2. nginx.confファイルを変更します。

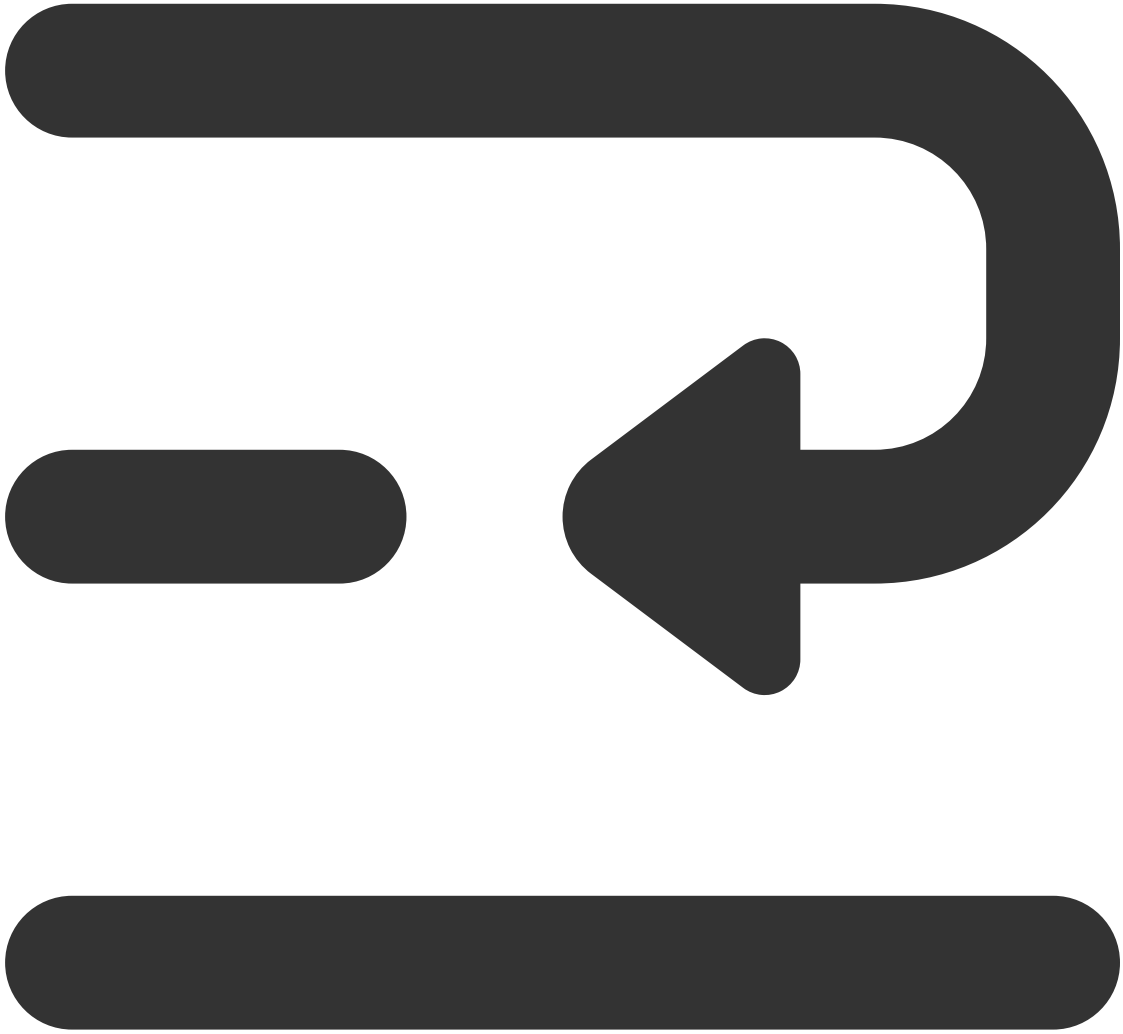


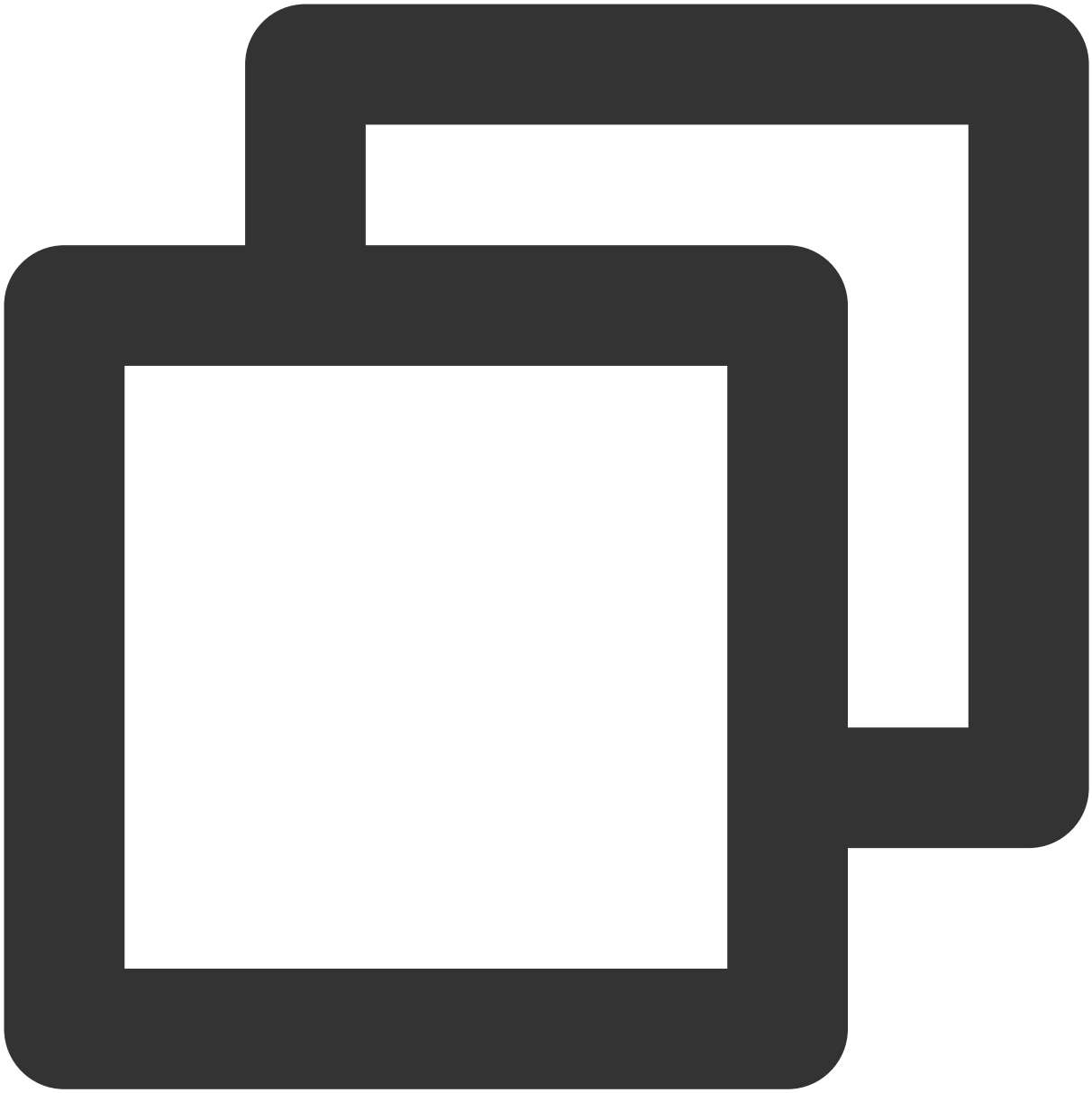
```
vi /etc/nginx/nginx.conf
```

設定フィールドと情報を変更します。

説明：

このうち `xx.xx.xx.xx` は実際のIPアドレスに変更する必要があります。このIPアドレスはCLBが提供するパブリックIPではありません。具体的なIPアドレスはNginxログで確認できます。複数のIPアドレスがある場合は、すべて入力が必要です。

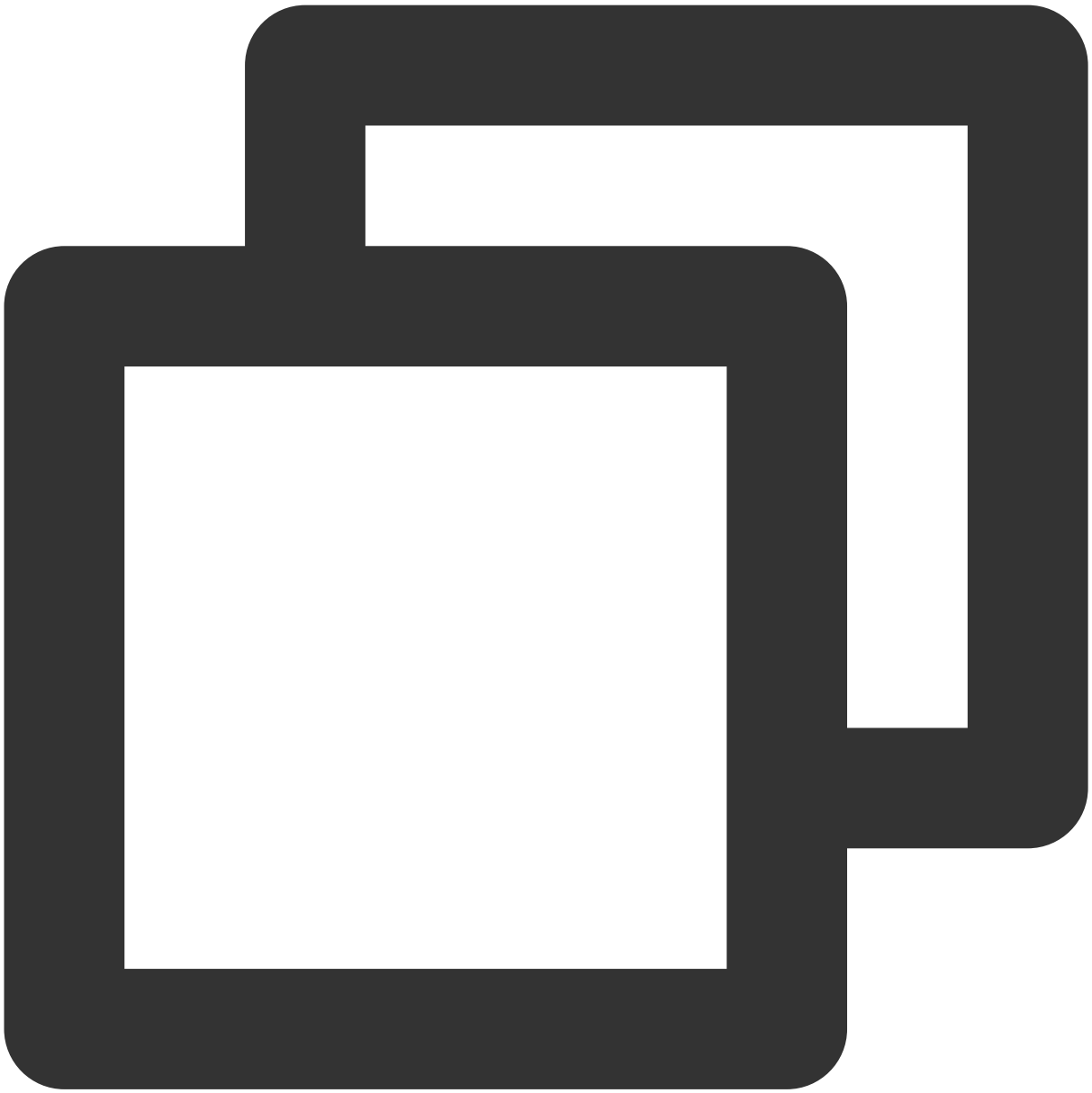




```
fastcgi connect_timeout 300;
fastcgi send_timeout 300;
fastcgi read_timeout 300;
fastcgi buffer_size 64k;
fastcgi buffers 4 64k;
fastcgi busy_buffers_size 128k;
fastcgi temp_file_write_size 128k;

# 設定フィールドと情報を変更します
set_real_ip_from xx.xx.xx.xx;
real_ip_header X-Forwarded-For;
```

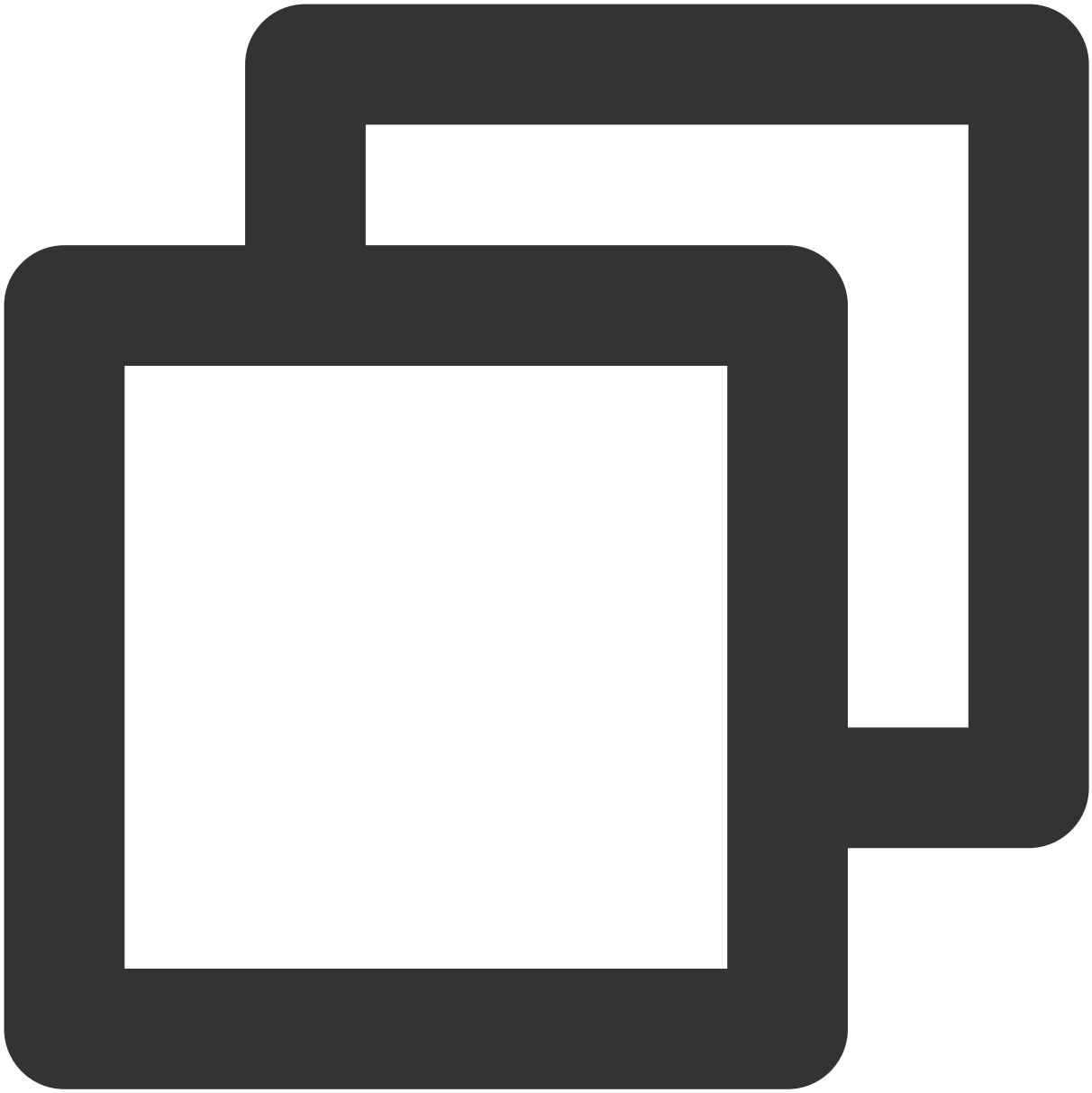
```
real_ip_recursive on;
```

3. Nginxを再起動します。



```
service nginx restart
```

4. Nginxのアクセスログを確認し、クライアントのリアルIPを取得できます。



```
cat /path/server/nginx/logs/access.log
```

ハイブリッドクラウドのデプロイシーンでのTOAによるクライアントリアルIPの取得

最終更新日：：2024-01-04 17:48:23

ここではハイブリッドクラウドのデプロイシーンおよびNAT64 CLBのシーンでのCLBのレイヤー4（TCPのみ）サービスにおいて、TOAによってクライアントのリアルソースIPを取得する方法についてご説明します。

コンソールでTOAを開く

TOAモジュールをロードする

バックエンドサービスの適用

（オプション）TOAモジュールのステータス監視

説明：

北京、上海、広州リージョンのNAT64 CLBに限り、TOAによるクライアントリアルソースIPの取得をサポートしています。

TOAによるクライアントのリアルソースIPの取得をサポートしているのはレイヤー4のTCPのみです。UDPおよびレイヤー7（HTTP/HTTPS）での取得はサポートしていません。

この機能は現在ベータ版テスト段階です。利用される場合は、チケット申請を送信してください。

ユースケース

ハイブリッドクラウドのデプロイシーン

ハイブリッドクラウドのデプロイにおいて、IDCのIPとクラウド上のVPCのIPの間でアドレスの重複が起こることがあります。このためSNAT IPを設定し、SNATを行ってソースIPの変換を行う必要があります。サーバーではリアルソースIPを取得できないため、TOAによって取得する必要があります。

NAT64 CLBのシーン

NAT64 CLBの場合、クライアントのリアルなIPv6ソースIPはIPv4のパブリックIPに変換されるため、リアルサーバーのサービス側からはクライアントのリアルIPv6 IPを取得することができません。

Tencent Cloud NAT64 CLBはクライアントのリアルIPを取得する機能を備えています。クライアントのリアルソースIPをTCPプロトコルのカスタムoptionに含めることで、リアルソースIPが埋め込まれたTCPデータパケットがサーバーに送信された際、サーバーが挿入したTOAカーネルモジュールによってTCPデータパケット内のクライアントリアルソースIPを取得することができます。このとき、クライアントのアプリケーションは、TOAカーネルモジュールが提供するインターフェースを呼び出すだけでクライアントリアルソースIPを取得できます。

制限事項

リソースの制限
TOAカーネルモジュールのコンパイル環境のカーネルバージョンはサービス環境のカーネルバージョンと一致している必要があります。
コンテナ環境では、TOAカーネルモジュールはホストにロードする必要があります。
TOAカーネルモジュールをロードする環境にはroot権限が必要です。
互換性の制限
UDPリスナーではTOAによるソースIPの取得はサポートされていません。
クライアントとリアルサーバーの間にあるデバイスの中に、TOAに関連する操作を行ったことがあるデバイスがある場合は、競合が起きる可能性があるため、サーバーによるリアルIP取得の有効性を保証することはできません。
TOAを挿入後、有効となるのは挿入後に新たに確立した接続のみであり、既存の接続に対しては無効です。
TOAモジュールはTCP option内のアドレスからの抽出などの処理を多く行う必要があるため、TOAモジュールによってサーバーの一部のパフォーマンスが低下することがあります。
Tencent CloudのTOAモジュールは、ユーザーがカスタマイズした他のカーネルモジュールとの互換性が保証されません。また、他のメーカーまたはオープンソースのTOAモジュールとの互換性も保証されません。
Tencentが自社開発したTencentOS埋め込みTOAモジュールはハイブリッドクラウドのデプロイシーンでのリアルソースIP取得をサポートしているため、サーバーのシステムがTencentOSであり、かつハイブリッドクラウドにデプロイされている場合は、`modprobe toa` コマンドを直接実行してロードおよび使用する方法を試すことができます。TencentOSとその他のLinuxディストリビューションシステムのTOAは別のものであり、同時に使用できないことに注意が必要です。

コンソールでTOAを開く

1. 作成済みのNAT64バージョンCLBインスタンスについて、詳しくはIPv6 NAT64 Cloud Load Balancerインスタンスの作成をご参照ください。
2. CLBコンソールにログインし、TCPリスナーを作成します。詳しくは TCPリスナーの設定をご参照ください。
3. 「リスナーの作成」のダイアログボックスにて、TOAスイッチをオンにします。

TOAモジュールのロード

1. Tencent Cloud上のLinuxのバージョンに応じて、対応するTOA パッケージを解凍します。
centos
CentOS 8.0 64
CentOS 7.6 64
CentOS 7.2 64
debian

Debian 9.0 64

suse linux

SUSE 12 64

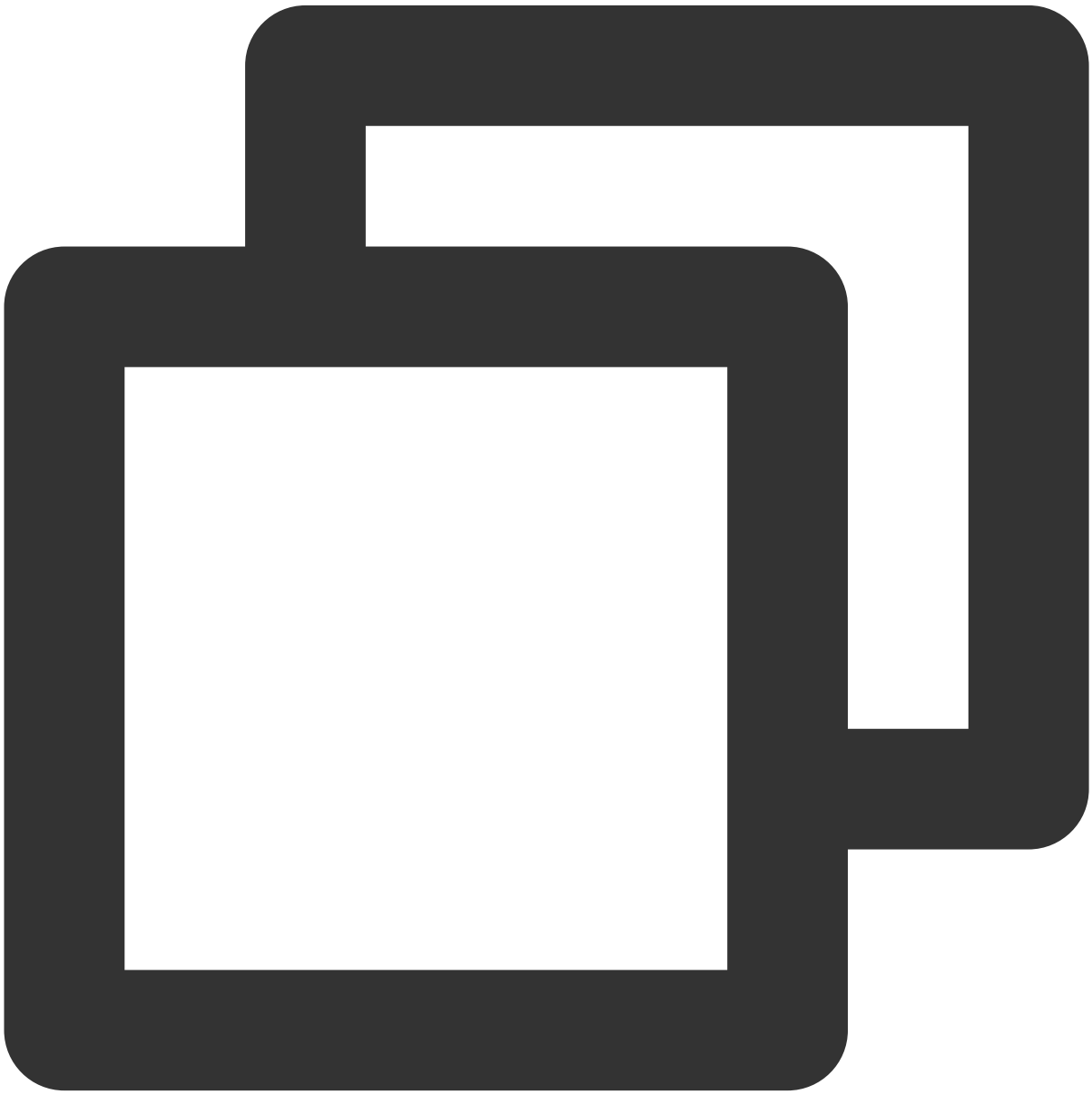
SUSE 11 64

ubuntu

Ubuntu 18.04.4 LTS 64

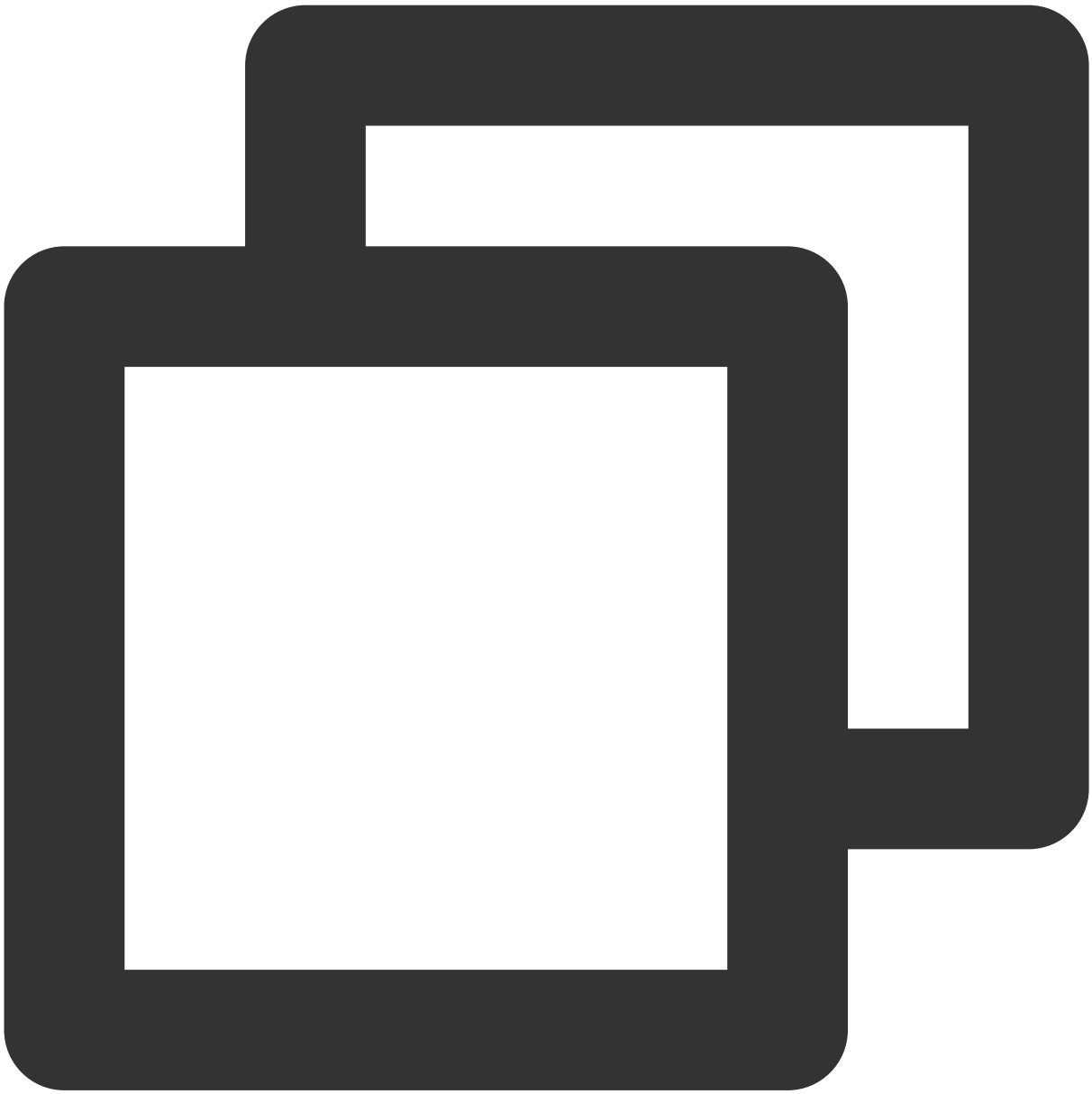
Ubuntu 16.04.7 LTS 64

2. 解凍完了後、cdコマンドを実行して解凍されたフォルダに進み、次のコマンドを実行してモジュールをロードします。



```
insmod toa.ko
```

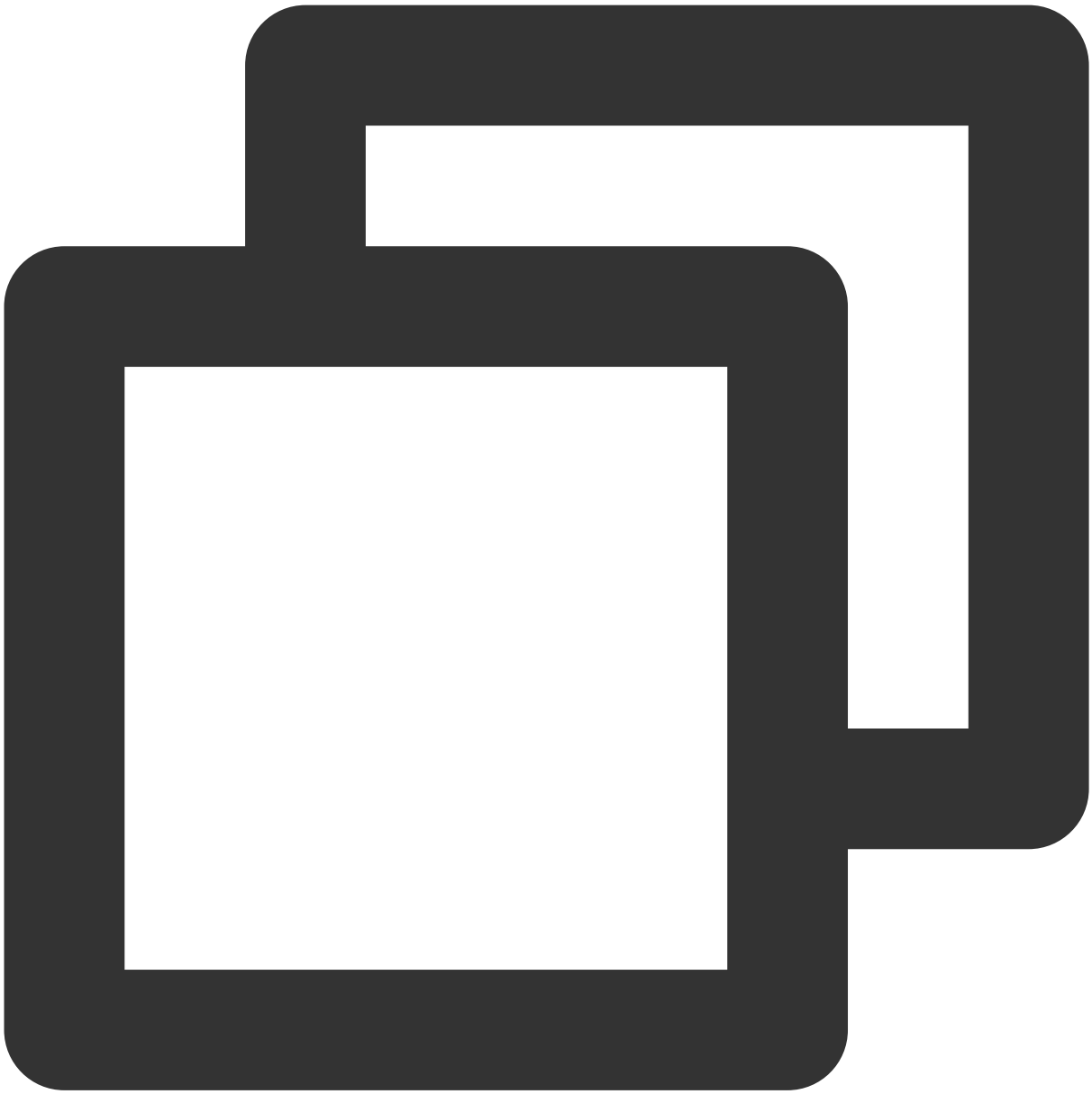
3. 次のコマンドを実行し、TOAモジュールのロードに成功したかどうかを確認します。「toa load success」と表示されれば、ロードは成功です。



```
dmesg -T | grep TOA
```

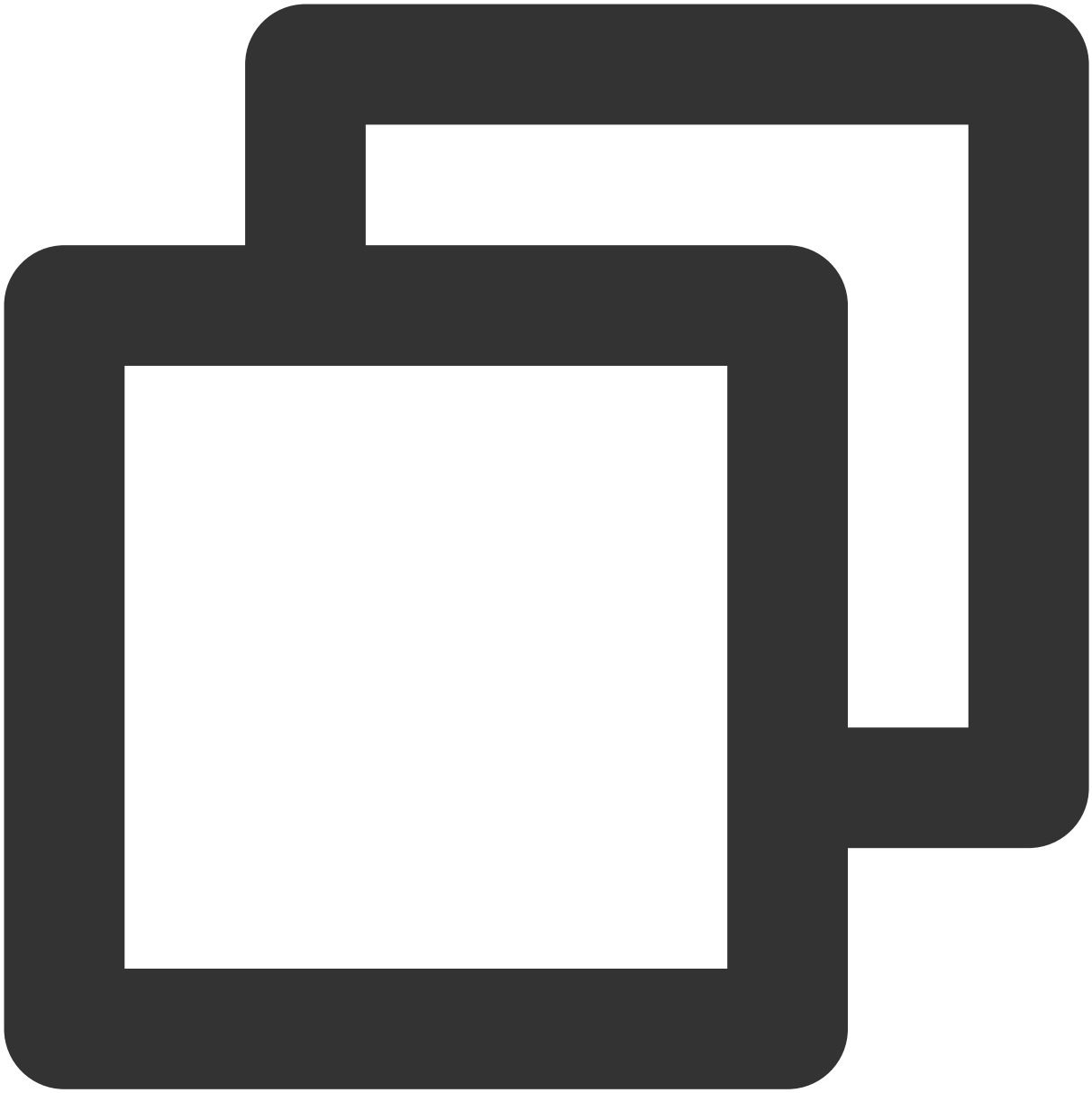
4. ロードに成功後、起動スクリプト内に `toa.ko` ファイルをロードします（マシンを再起動した場合はkoファイルの再ロードが必要です）。

5.（オプション）それ以降TOAモジュールを使用する必要がない場合は、次のコマンドを実行してアンインストールします。



```
rmmod toa
```

6.（オプション）次のコマンドを実行し、TOAモジュールのアンインストールに成功したかどうかを確認します。「TOA unloaded」と表示されれば、アンインストールは成功です。



```
dmesg -T
```

上記のダウンロードファイルに、ご自身のOSバージョンに対応するインストールパッケージがない場合は、Linux汎用版のソースコードパッケージをダウンロードし、コンパイル後に対応するkoを取得することができます。このバージョンはCentos8、Centos7、Ubuntu18.04、Ubuntu16.04などの数多くの代表的なLinuxディストリビューションをサポートしています。

説明：

Linuxのカーネルバージョンは数多くあり、LinuxディストリビューションのOS市場も巨大でバージョンも多岐にわたるため、カーネルモジュールの互換性の問題を考慮し、ご利用のシステム上でTOAソースコードパッケージ

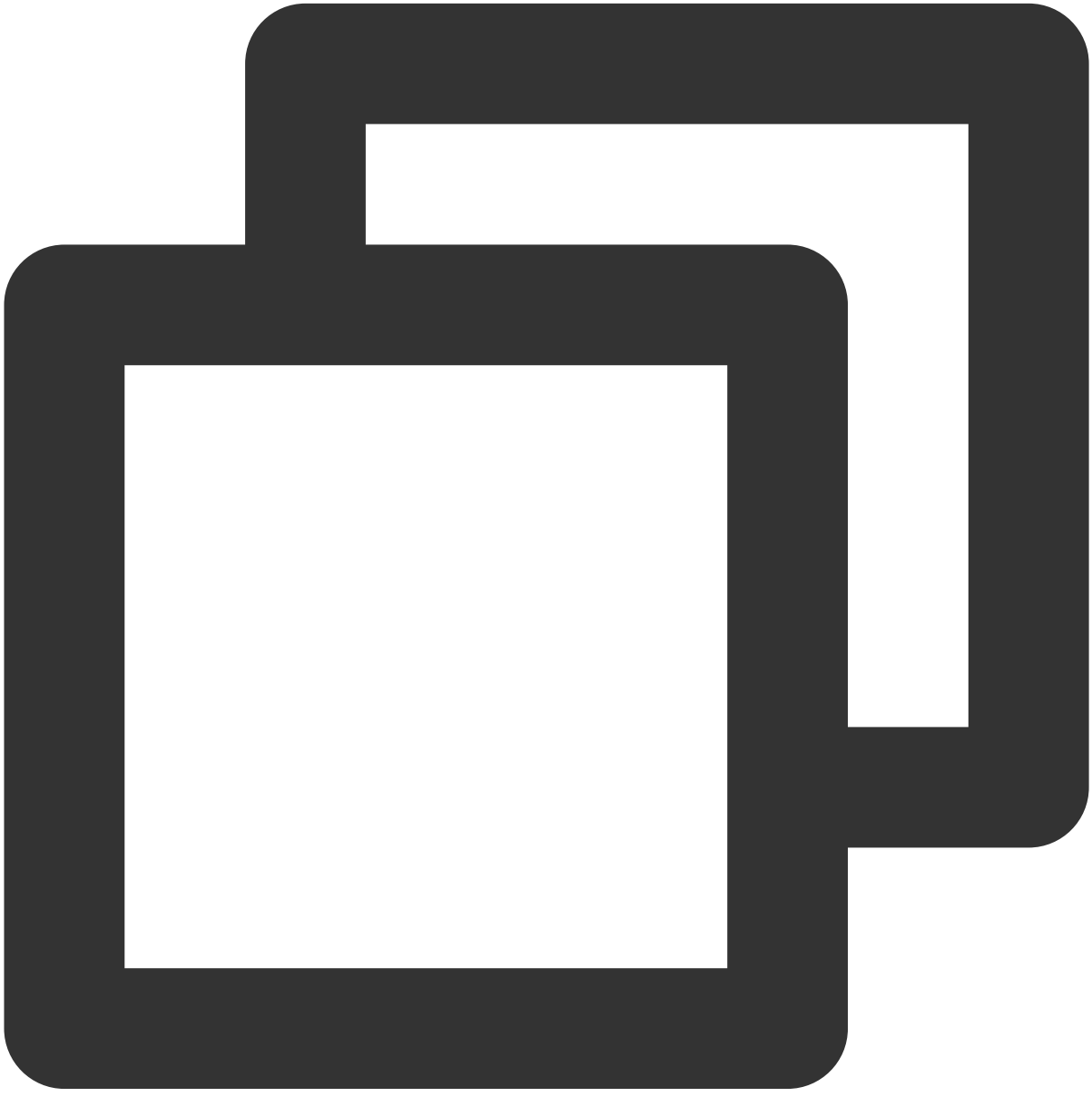
のコンパイルを行ってから使用することをお勧めします。

1. ソースコードパッケージのダウンロード

ご注意：

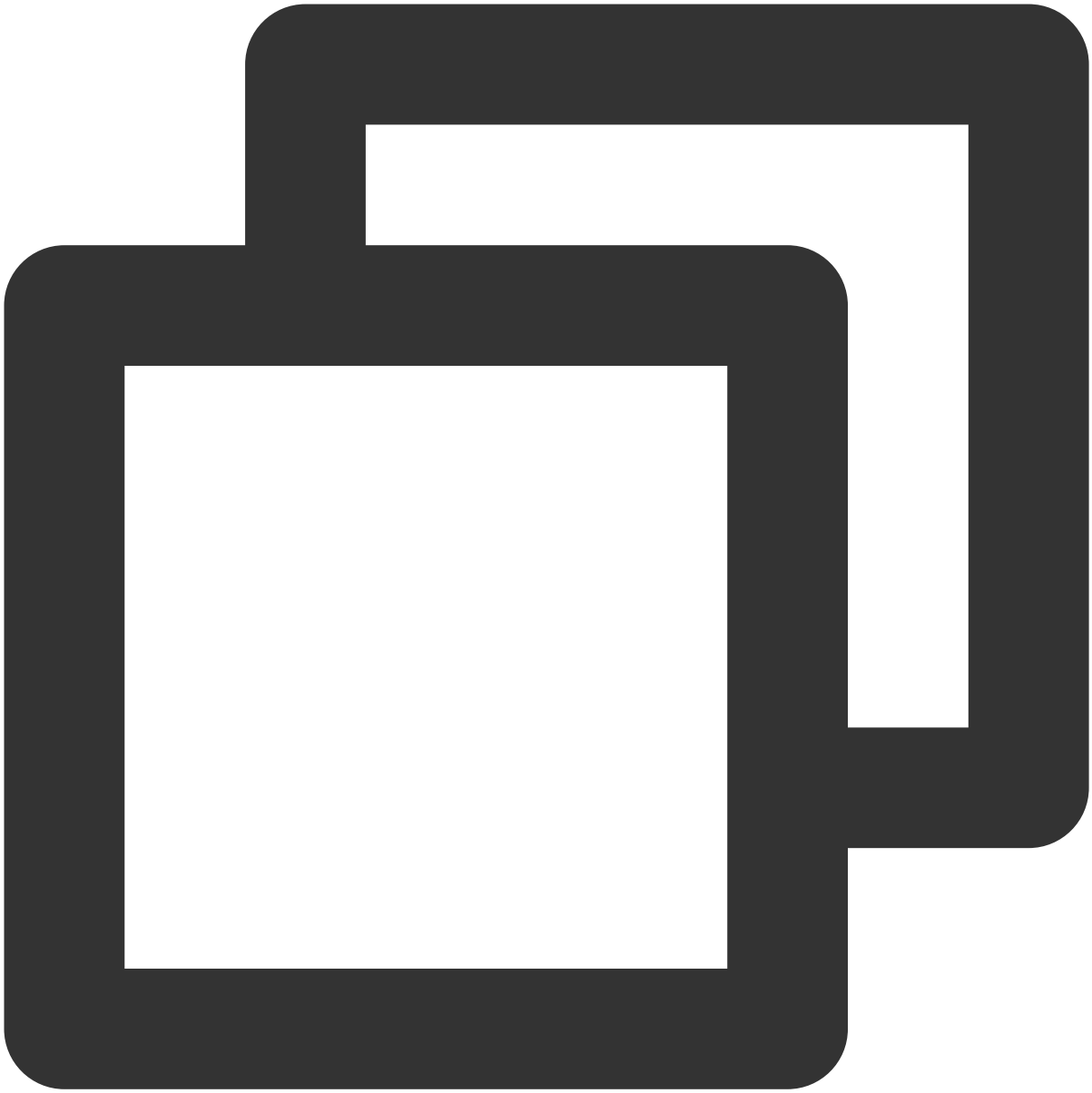
LinuxとTencentのTLinuxのTOAモジュールは併用できません。システムに応じて対応するTOAモジュールソースコードパッケージを選択してください。

Linux



```
wget "https://clb-toa-1255852779.file.myqcloud.com/tgw_toa_linux_ver.tar.gz"
```

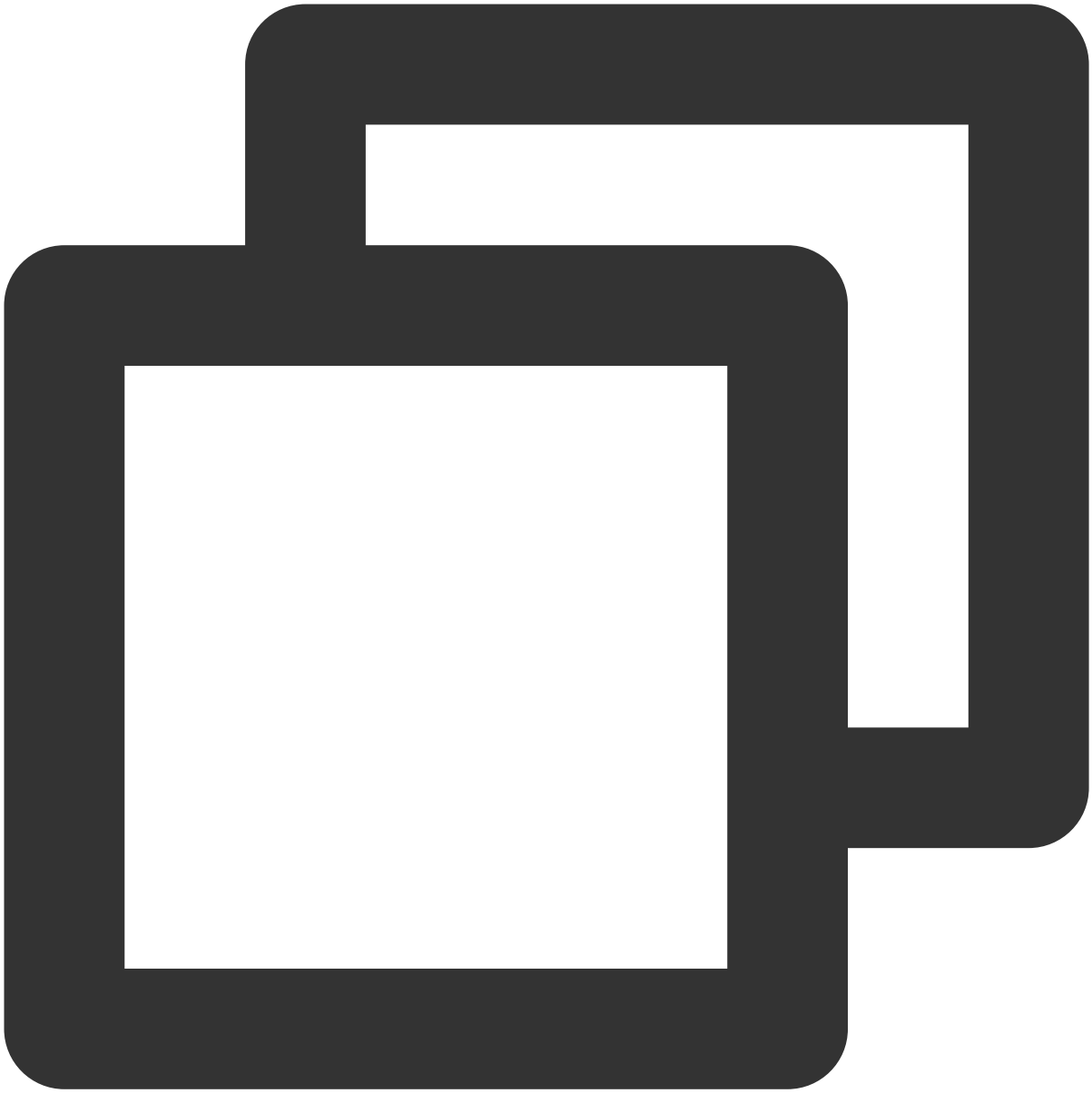
Tencent TLinux



```
wget "https://clb-toa-1255852779.file.myqcloud.com/tgw_toa_tlinux_ver.tar.gz"
```

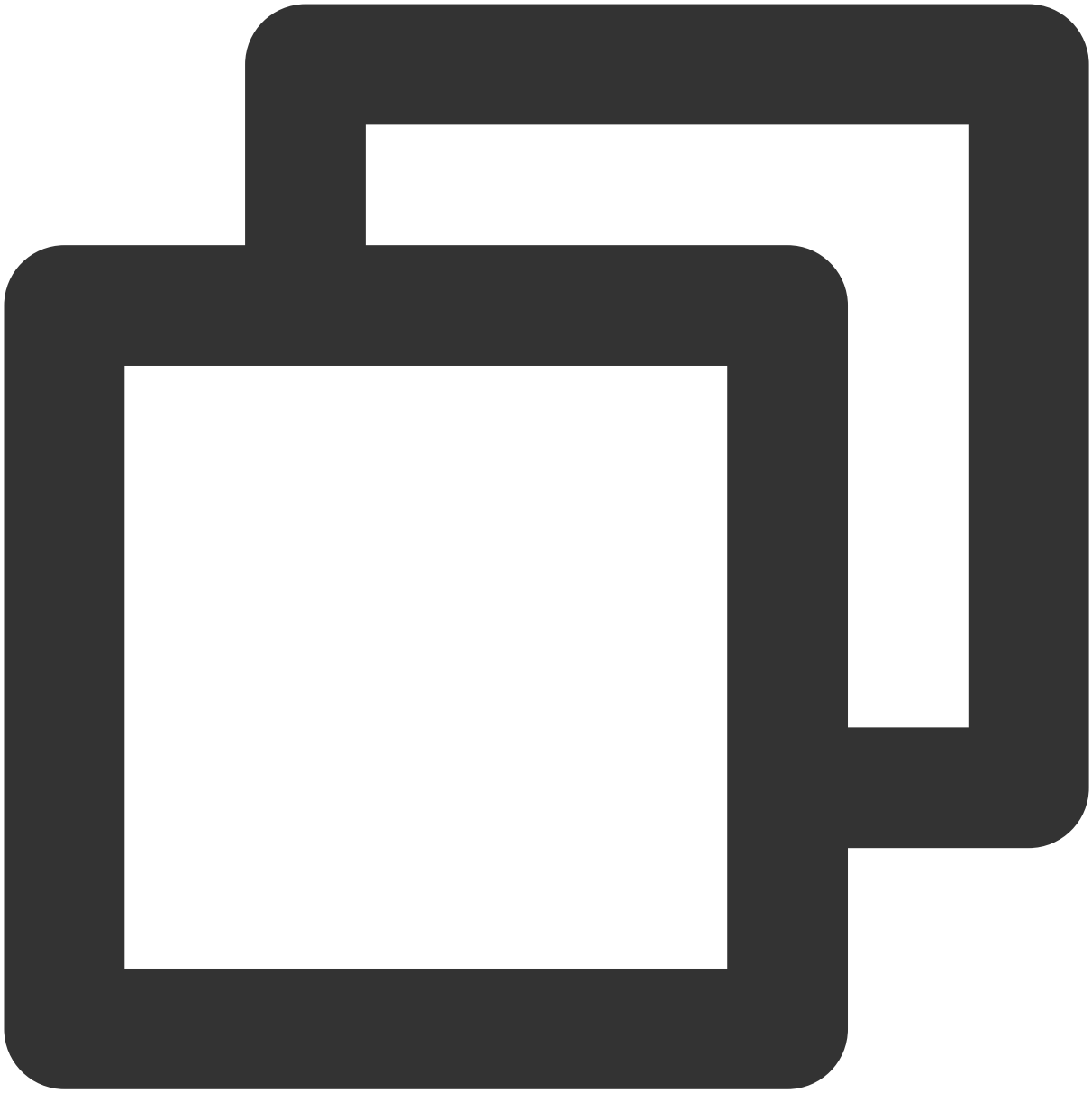
2. TOAカーネルモジュールのLinux環境のコンパイルの前に、GCCコンパイラ、Makeツールおよびカーネルモジュール開発パッケージをインストールする必要があります。

CentOS環境下でのインストール操作



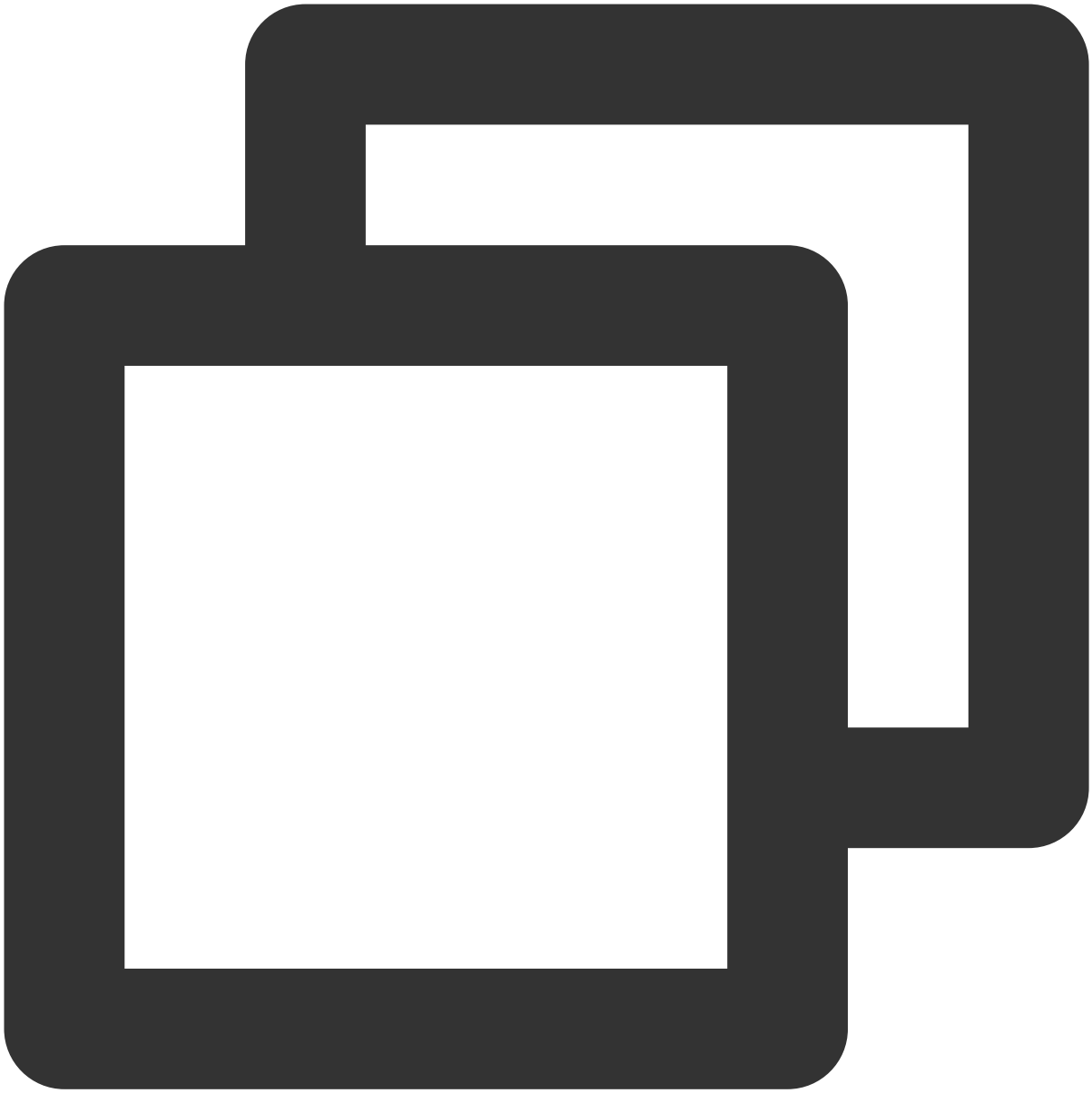
```
yum install gcc
yum install make
//カーネルモジュール開発パッケージをインストールします。開発パッケージのヘッダーファイルとライブラリ
yum install kernel-devel-`uname -r`
yum install devtoolset-8
```

Ubuntu、Debian環境下でのインストール操作



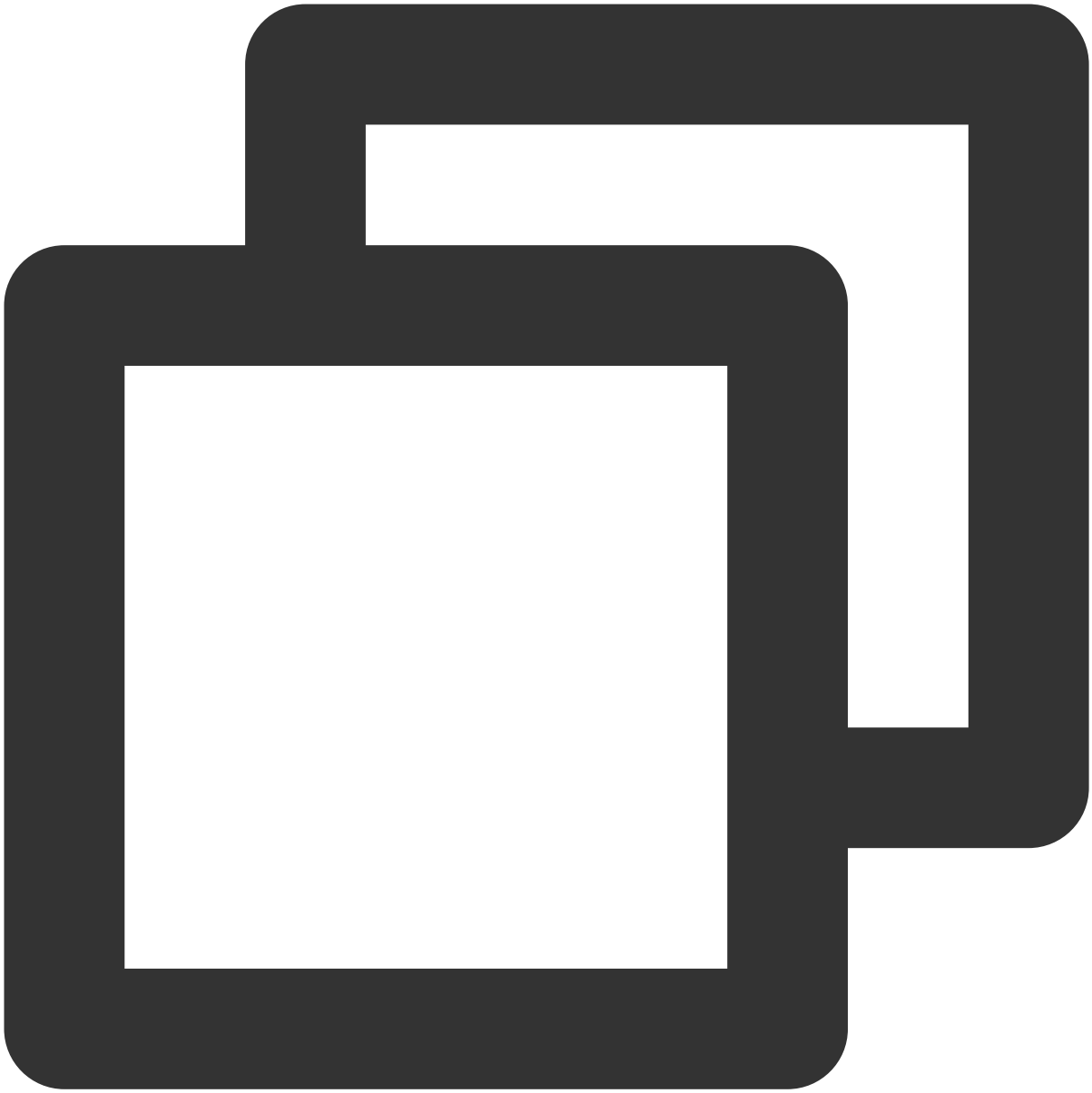
```
apt-get install gcc
apt-get install make
//カーネルモジュール開発パッケージをインストールします。開発パッケージのヘッダーファイルとライブラリ
apt-get install linux-headers-`uname -r`
apt-get install devtoolset-8
```

SUSE環境下でのインストール操作



```
zypper install gcc
zypper install make
//カーネルモジュール開発パッケージをインストールします。開発パッケージのヘッダーファイルとライブラリ
zypper install kernel-default-devel
zypper install devtoolset-8
```

3. ソースコードをコンパイルしてtoa.koファイルを生成します。コンパイル中に `warning` および `error` が表示されなければ、コンパイルは成功です。Linuxシステムに対応するソースコードパッケージの例を挙げます。



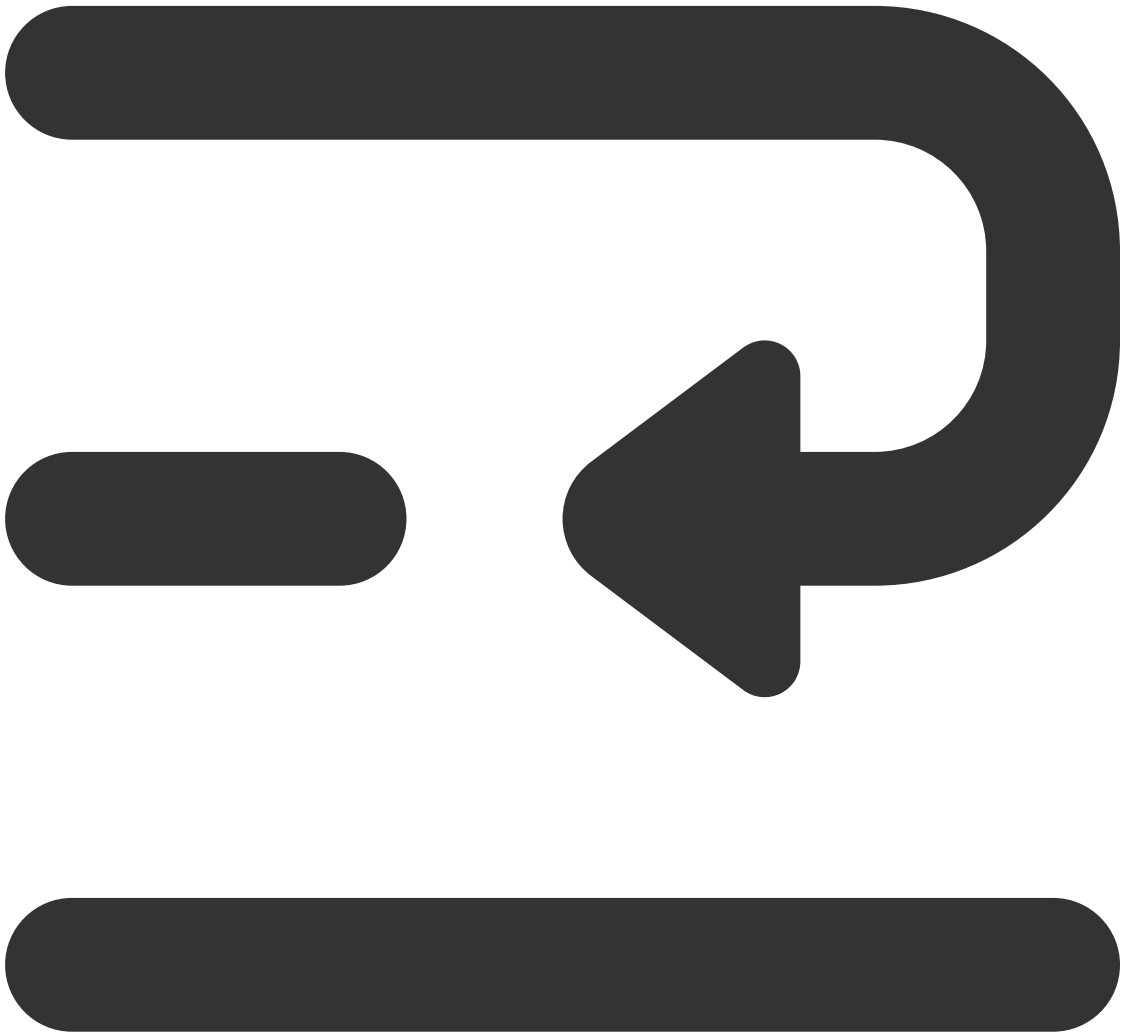
```
tar zxvf tgw_toa_linux_ver.tar.gz
cd tgw_toa_linux_ver//解凍後のtgw_toaディレクトリに進みます
make
```

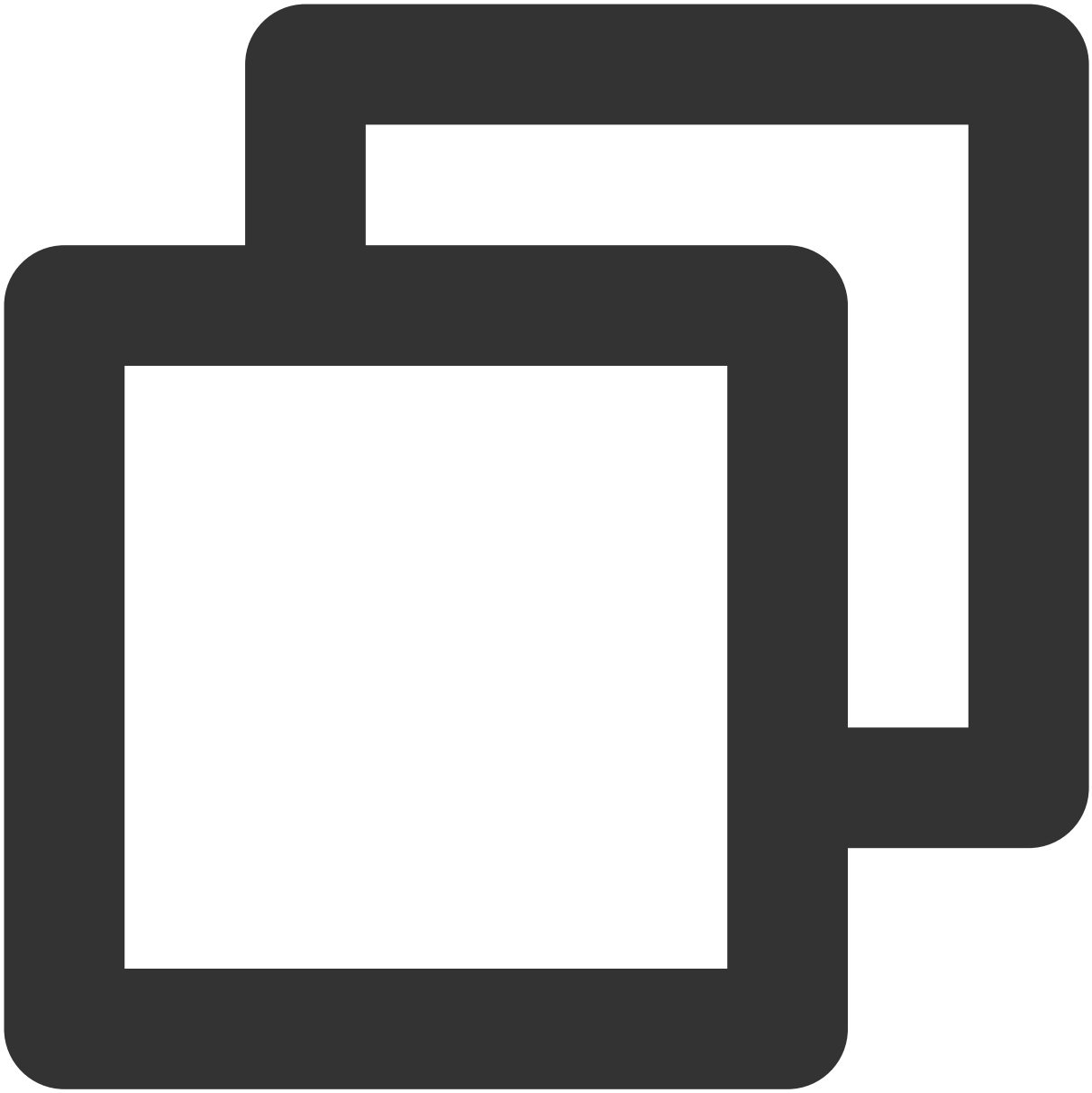
4. toa.koのコンパイルに成功後、上記のステップ2のTOAモジュールロード操作を実行します。

バックエンドサービスの適用

ハイブリッドクラウドのデプロイシーン

ハイブリッドクラウドのデプロイシーンでバックエンドサービスを適用する場合、コードの変更は必要なく、Linuxネットワークプログラミングの標準インターフェースを呼び出すだけでアクセスユーザーのリアルソースIPを取得できます。Cコードの例は次のとおりです。



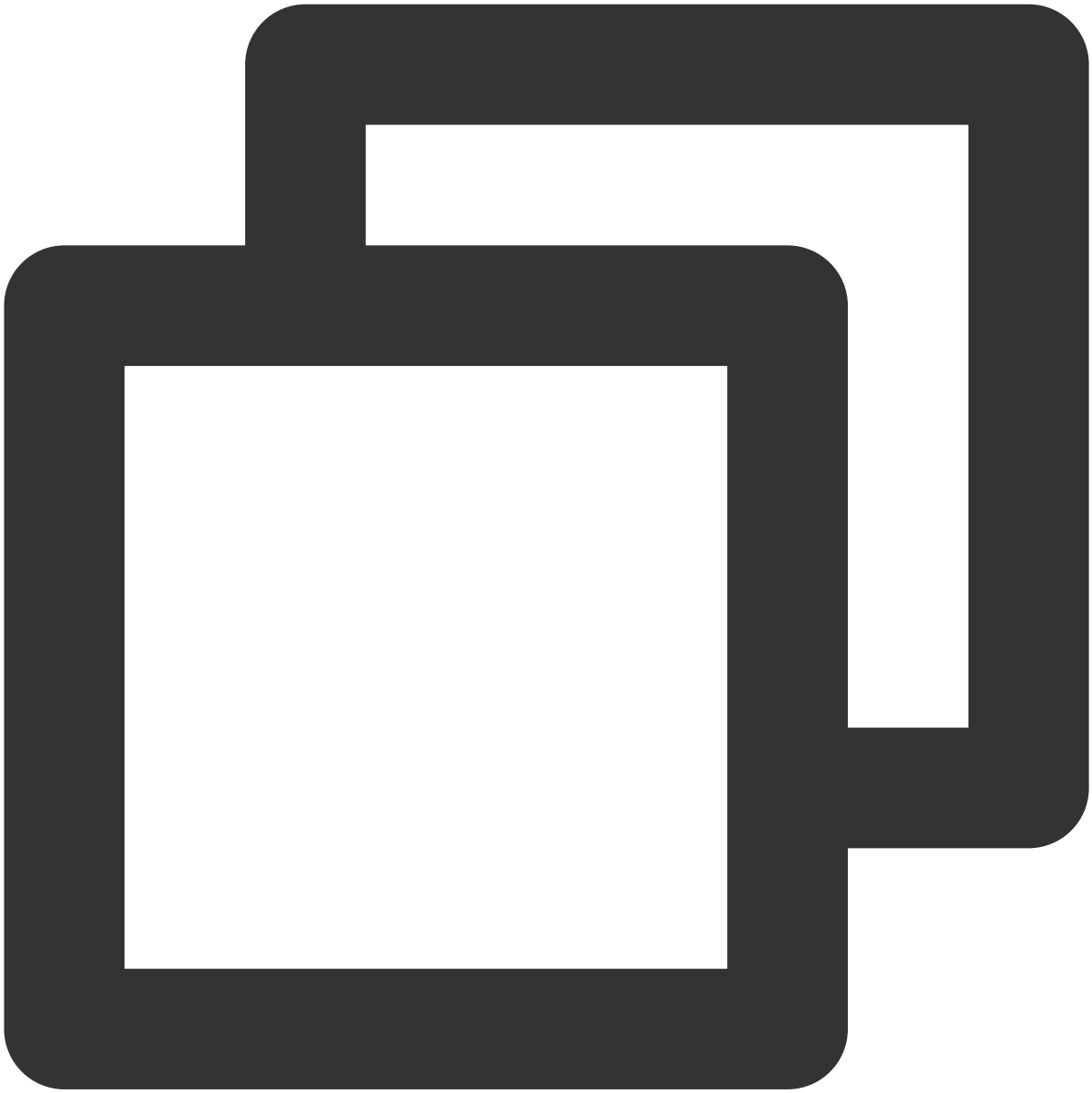


```
struct sockaddr v4addr;
len = sizeof(struct sockaddr);
//get_peer_nameはLinuxネットワークプログラミングの標準インターフェースです。
if (get_peer_name(client_fd, &v4addr, &len) == 0) {
    inet_ntop(AF_INET, &(((struct sockaddr_in *)&v4addr)->sin_addr), from, sizeof(f
    printf("real client v4 [%s]:%d\\n", from, ntohs(((struct sockaddr_in *)&v4addr)
}
```

NAT64 CLBの

シーンNAT64 CLBのシーンでは、TOAのソースアドレスパススルー機能を使用し、バックエンドサーバーにtoa.koカーネルモジュールを挿入した後、アプリケーションのソースコードを変更してリアルソースIP取得機能を適用させる必要があります。

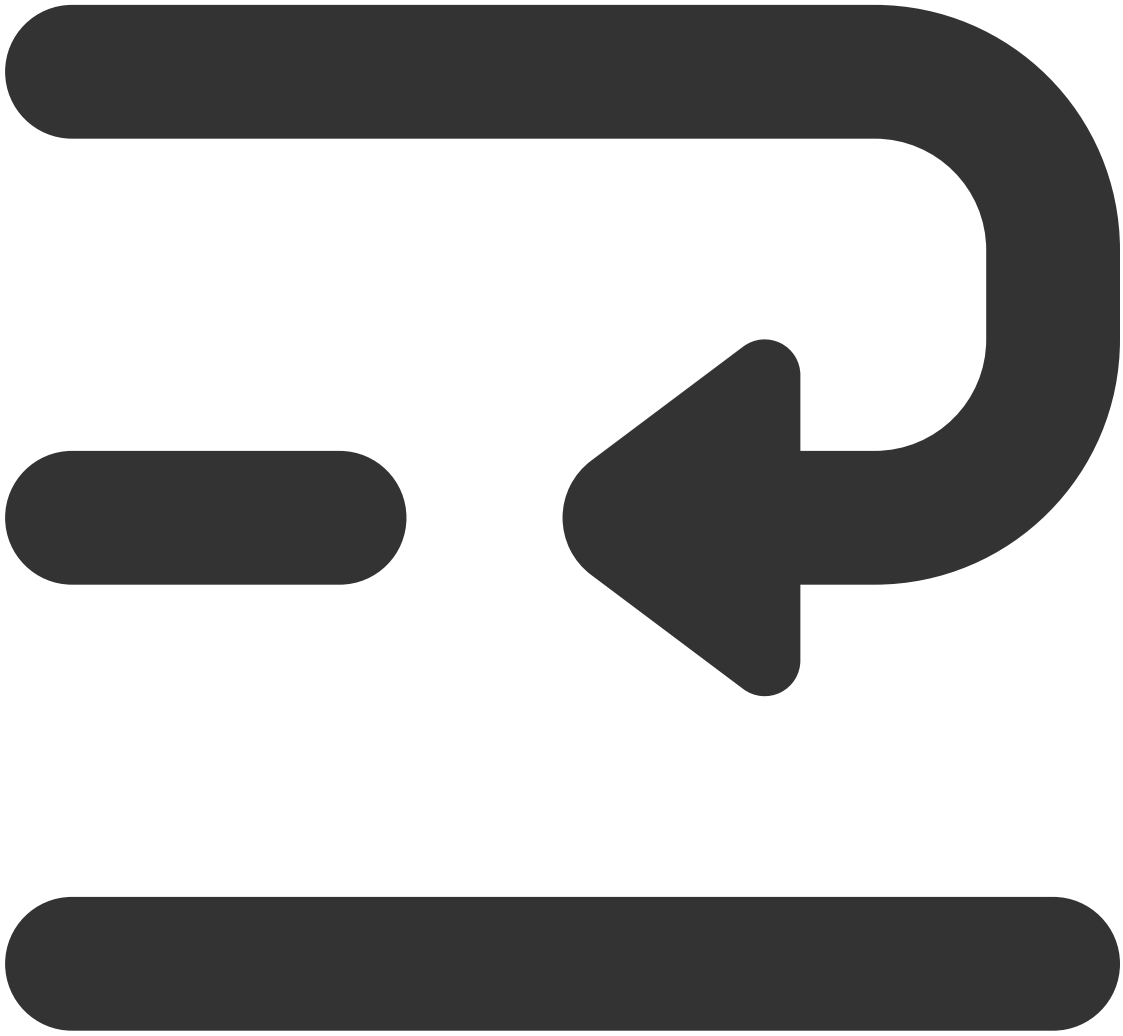
1. 初めに、アドレスの保存に用いるデータ構造を定義します。

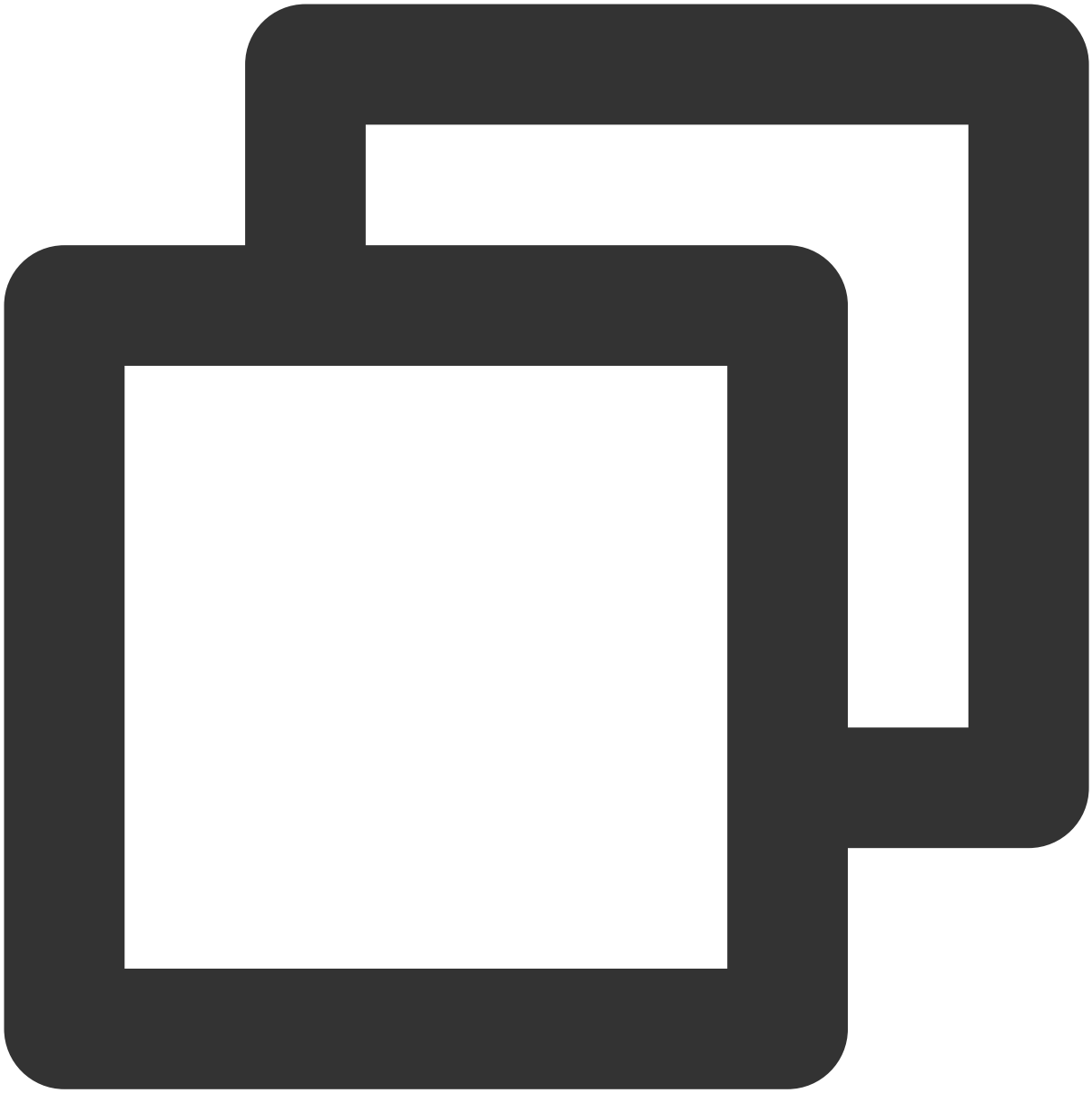


```
struct toa_nat64_peer {
    struct in6_addr saddr;
    uint16_t sport;
};
....
struct toa_nat64_peer client_addr;
```

```
....
```

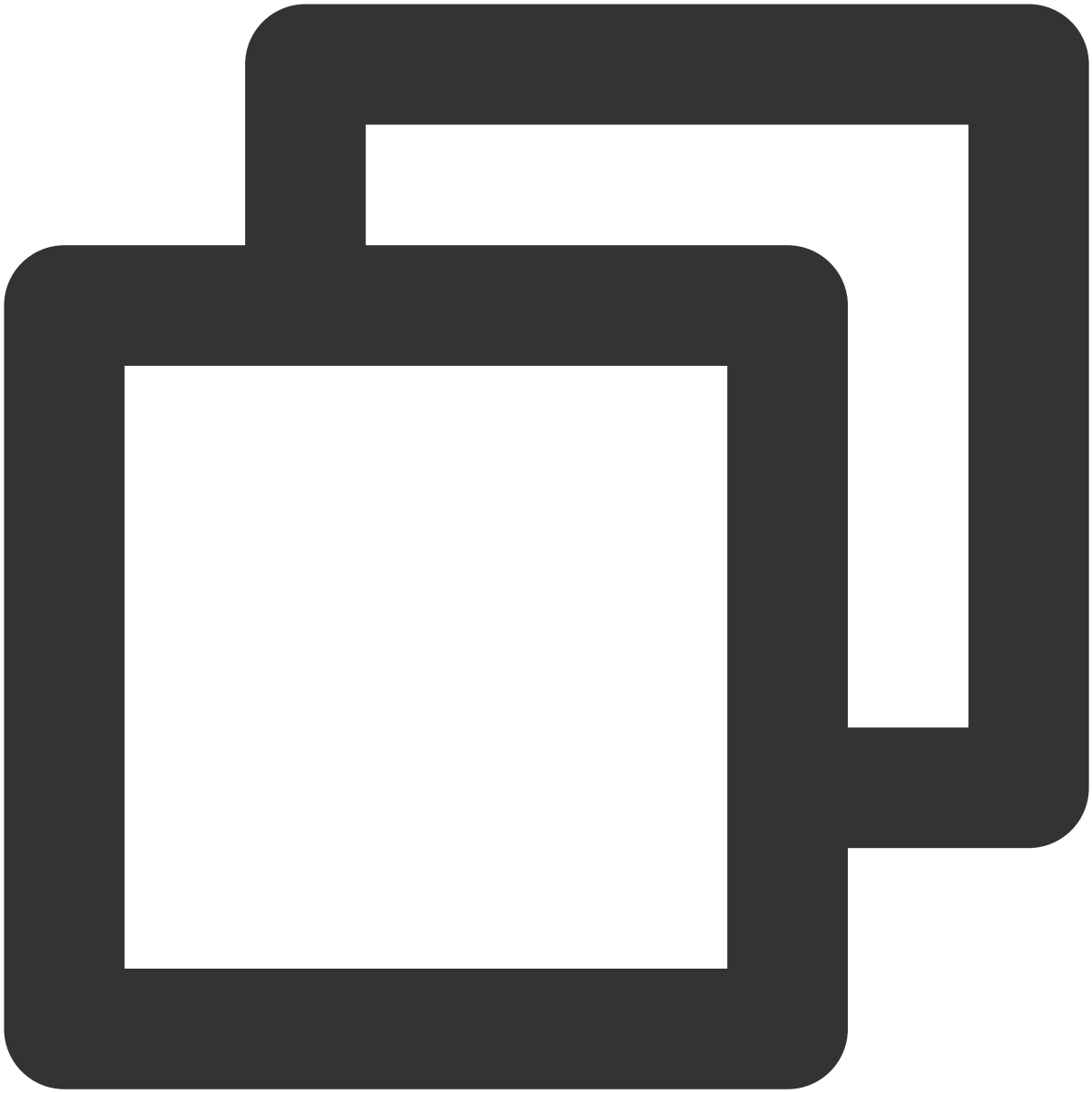
2. 次にメッセージを定義し、関数を呼び出してリアルIPv6ソースアドレスを取得します。





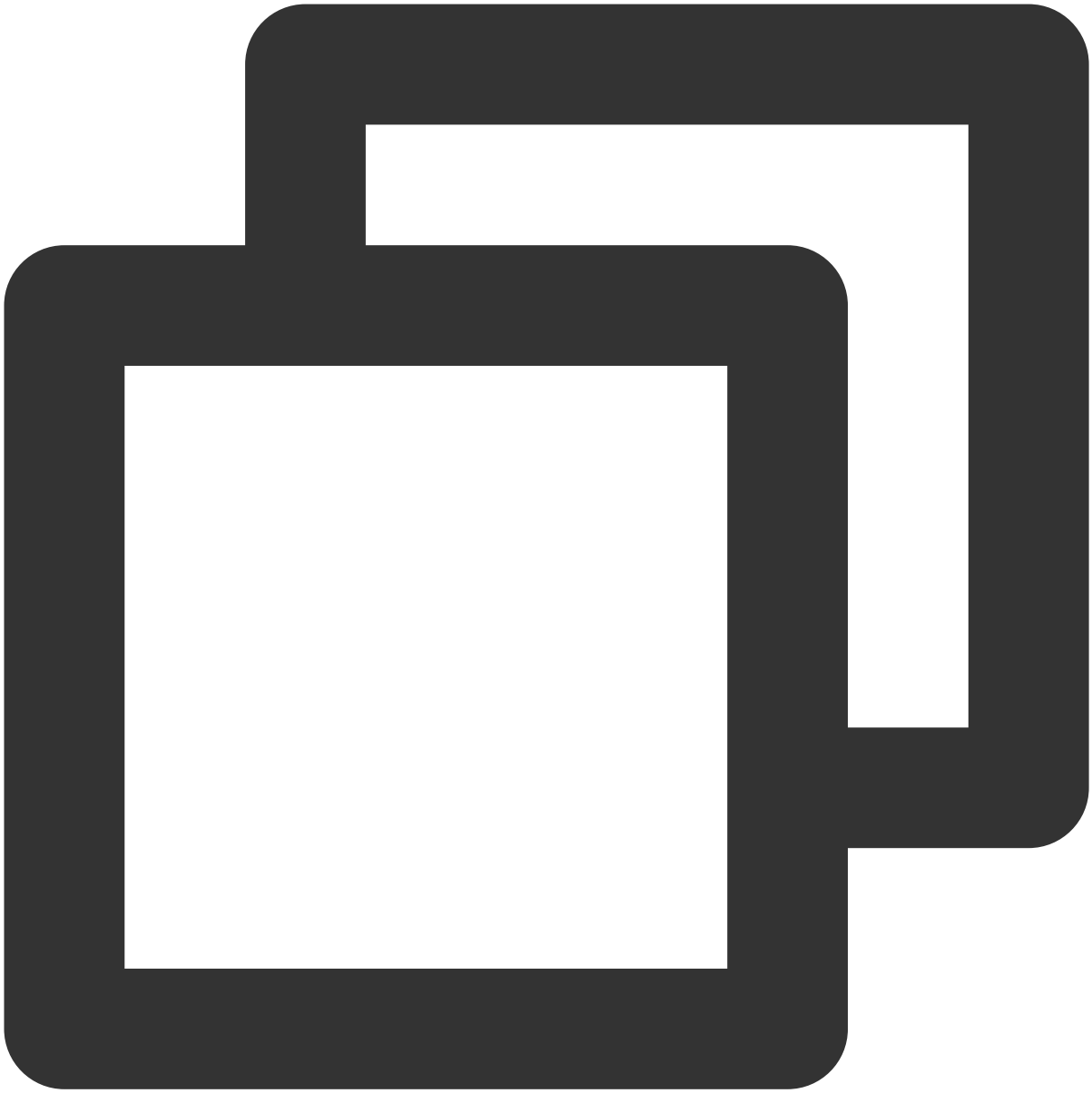
```
enum {
    TOA_BASE_CTL            = 4096,
    TOA_SO_SET_MAX          = TOA_BASE_CTL,
    TOA_SO_GET_LOOKUP       = TOA_BASE_CTL,
    TOA_SO_GET_MAX          = TOA_SO_GET_LOOKUP,
};
getsockopt(client_fd, IPPROTO_IP, TOA_SO_GET_LOOKUP, &client_addr, &len);
```

3. 最後にアドレスを取得します。



```
real_ipv6_saddr = client_addr.saddr;
real_ipv6_sport = client_addr.sport;
```

完全な例は以下のとおりです。



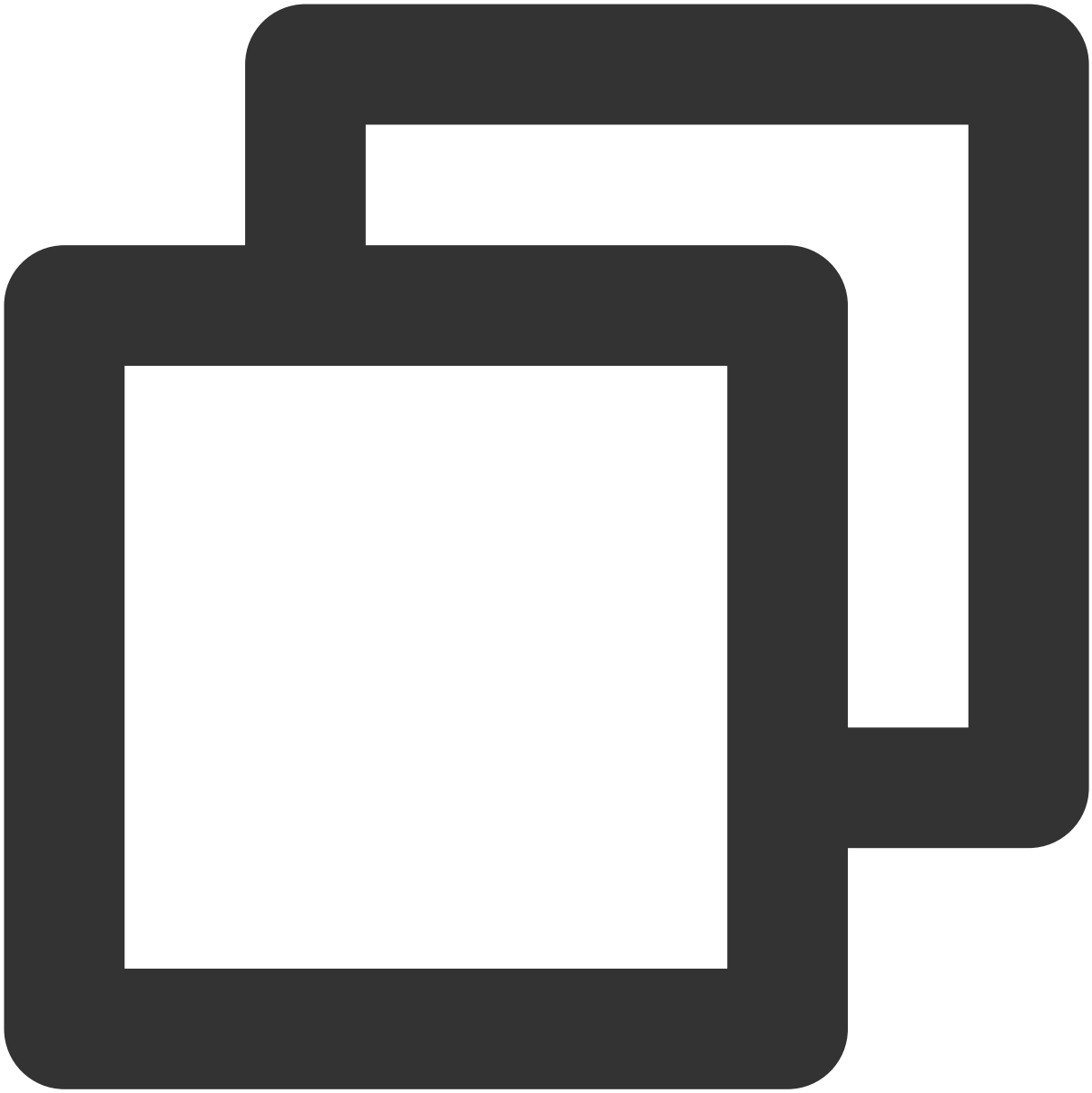
```
//リアルIP取得用関数の呼び出しメッセージを定義する必要があります。値は4096とします。
enum {
    TOA_BASE_CTL            = 4096,
    TOA_SO_SET_MAX          = TOA_BASE_CTL,
    TOA_SO_GET_LOOKUP       = TOA_BASE_CTL,
    TOA_SO_GET_MAX          = TOA_SO_GET_LOOKUP,
};
//アドレスの保存に用いるデータ構造を定義する必要があります。
struct toa_nat64_peer {
    struct in6_addr saddr;
    uint16_t sport;
```

```
};
//アドレスの保存に用いる変数を宣言します。カスタムタイプのアドレス保存用のデータ構造です。
struct toa_nat64_peer client_addr;
.……
//クライアントのファイルディスクリプタを取得します。このうちlistenfdはサーバーのリスニングファイル
client_fd = accept(listenfd, (struct sockaddr*)&caddr, &length);
//関数を呼び出して、対応するNAT64シーンのユーザーのリアルソースIPを取得します。
char from[40];
int len = sizeof(struct toa_nat64_peer);
if (getsockopt(client_fd, IPPROTO_IP, TOA_SO_GET_LOOKUP, &client_addr, &len) == 0)
    inet_ntop(AF_INET6, &client_addr.saddr, from, sizeof(from));
    //ソースIPとソースportの情報を取得します
    printf("real client [%s]:%d\\n", from, ntohs(client_addr.sport));
}
```

ハイブリッドクラウドのデプロイとNAT64 CLBの併用シーン

ハイブリッドクラウドのデプロイとNAT64 CLBの併用シーンでは、TOAのソースアドレスパススルー機能を使用し、バックエンドサーバーにtoa.koカーネルモジュールを挿入した後、アプリケーションのソースコードを変更してリアルソースIP取得機能を適用させる必要があります。

完全な例は以下のとおりです。



```
//リアルIP取得用関数の呼び出しメッセージを定義する必要があります。値は4096とします。
enum {
    TOA_BASE_CTL = 4096,
    TOA_SO_SET_MAX = TOA_BASE_CTL,
    TOA_SO_GET_LOOKUP = TOA_BASE_CTL,
    TOA_SO_GET_MAX = TOA_SO_GET_LOOKUP,
};

//アドレスの保存に用いるデータ構造を定義する必要があります。
struct toa_nat64_peer {
    struct in6_addr saddr;
```

```
    uint16_t sport;
};

//アドレスの保存に用いる変数を宣言します。カスタムタイプのアドレス保存用のデータ構造です。
struct toa_nat64_peer client_addr_nat64;
.......
//クライアントのファイルディスクリプタを取得します。このうちlistenfdはサーバーのリスニングファイル
//関数を呼び出して、対応するNAT64シーンのリアルなユーザーソースIPを取得します。
char from[40];
int len = sizeof(struct toa_nat64_peer);
int ret;
ret = getsockopt(client_fd, IPPROTO_IP, TOA_SO_GET_LOOKUP, &client_addr_nat64, &len
if (ret == 0) {
    inet_ntop(AF_INET6, &(client_addr_nat64.saddr), from, sizeof(from));
    //ソースIPとソースPortの情報を取得します。
    printf("real client v6 [%s]:%d\\n", from, ntohs(client_addr_nat64.sport));
} else if (ret != 0) {
    struct sockaddr v4addr;
    len = sizeof(struct sockaddr);
    //ソースIPとソースPortの情報を取得します。この関数が取得するソースアドレスについては以下に注意
    //ハイブリッドクラウドのデプロイシーンのSNAT IPのリンクを経たものはリアルソースアドレスです。
    //ハイブリッドクラウドのデプロイシーンのSNAT IPを経ておらず、NAT64のリンクも経ていないものは
    //このため、この関数のセマンティクスはリアルなクライアントアドレス、ポートを取得するためのもの
    if (get_peer_name(client_fd, &v4addr, &len) == 0) {
        inet_ntop(AF_INET, &(((struct sockaddr_in *)&v4addr)->sin_addr), from, size
        printf("real client v4 [%s]:%d\\n", from, ntohs(((struct sockaddr_in *)&v4a
    }
}
```

（オプション）TOAモジュール状態の監視

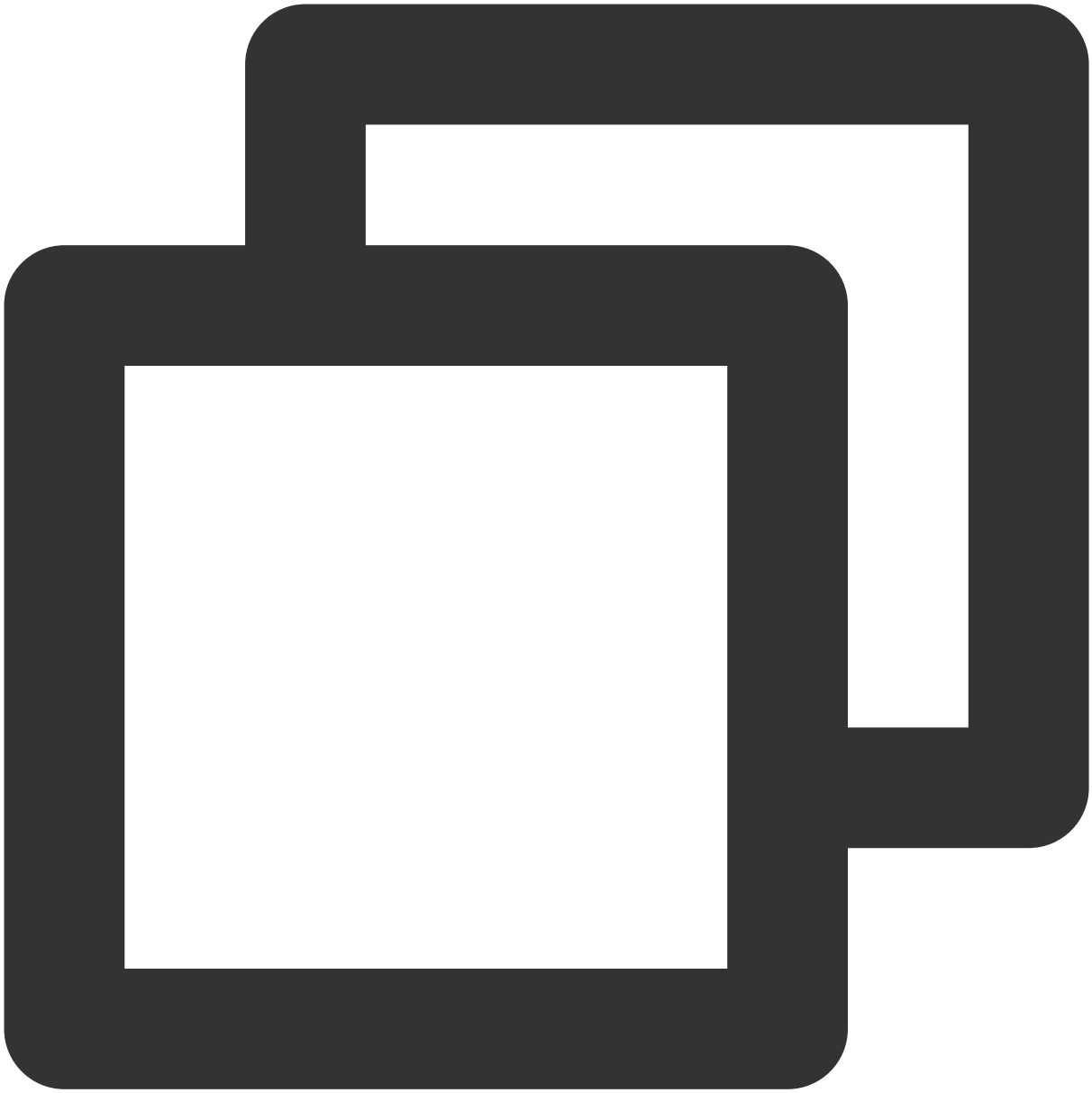
TOAカーネルモジュールはその実行の安定性を保障するための監視機能も備えています。 toa.koをカーネルモジュールに挿入後、コンテナのあるホスト上で、次の2つの方法によりTOAモジュールの動作状態を監視することができます。

方法1：TOAに保存されている接続のIPv6アドレスを確認

し、次のコマンドを実行してTOAに保存されている接続のIPv6アドレスを確認します。

ご注意：

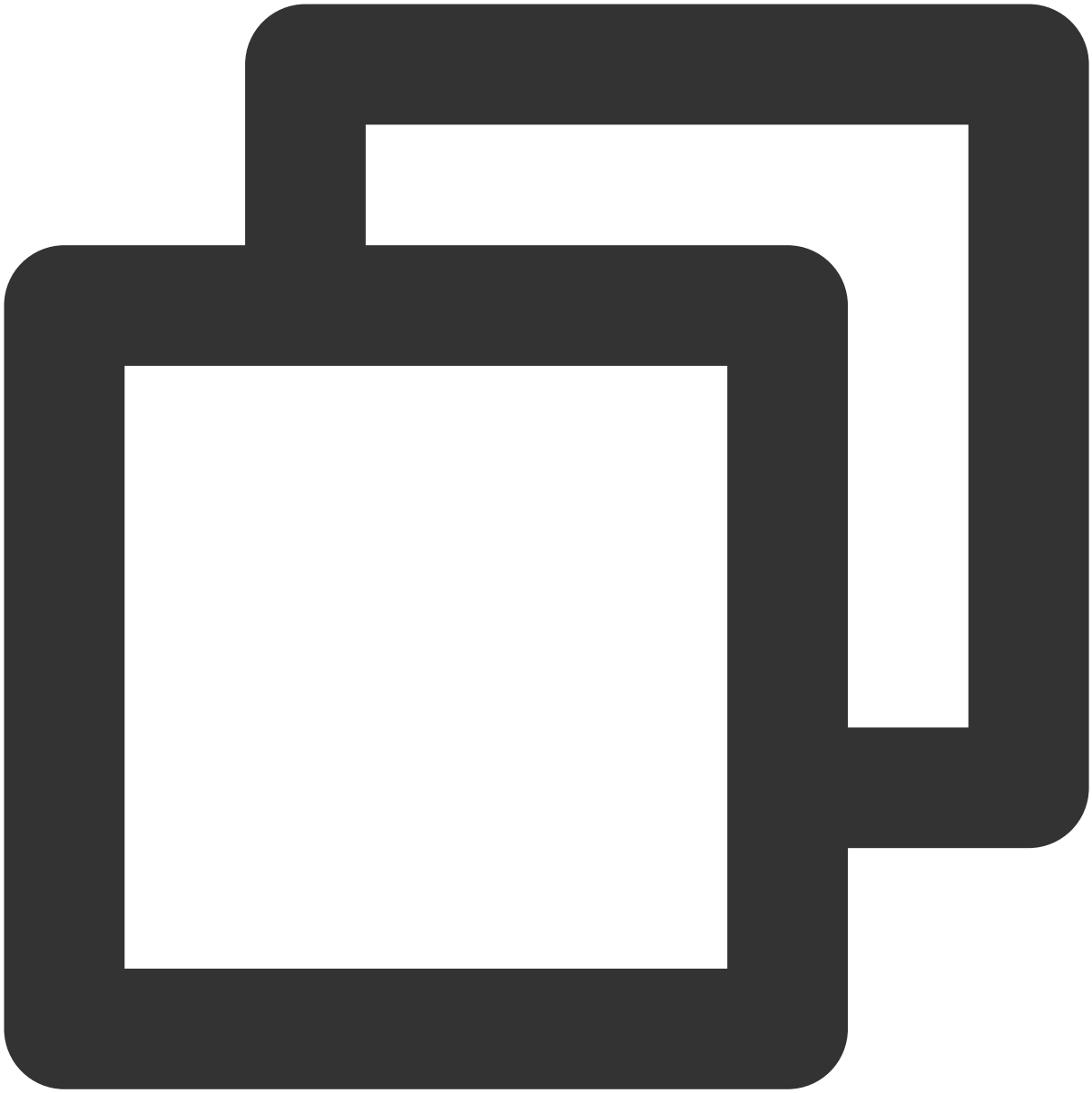
このコマンドはパフォーマンスを低下させる可能性があるため、このコマンドを頻繁に呼び出して確認することは避けてください。



```
cat /proc/net/toa_table
```

方法2：TOAに関連するカウントステータスを確認

し、次のコマンドを実行してTOAに関連するカウントステータスを確認します。



```
cat /proc/net/toa_stats
```

このうちの主な監視指標と、その意味は次のとおりです。

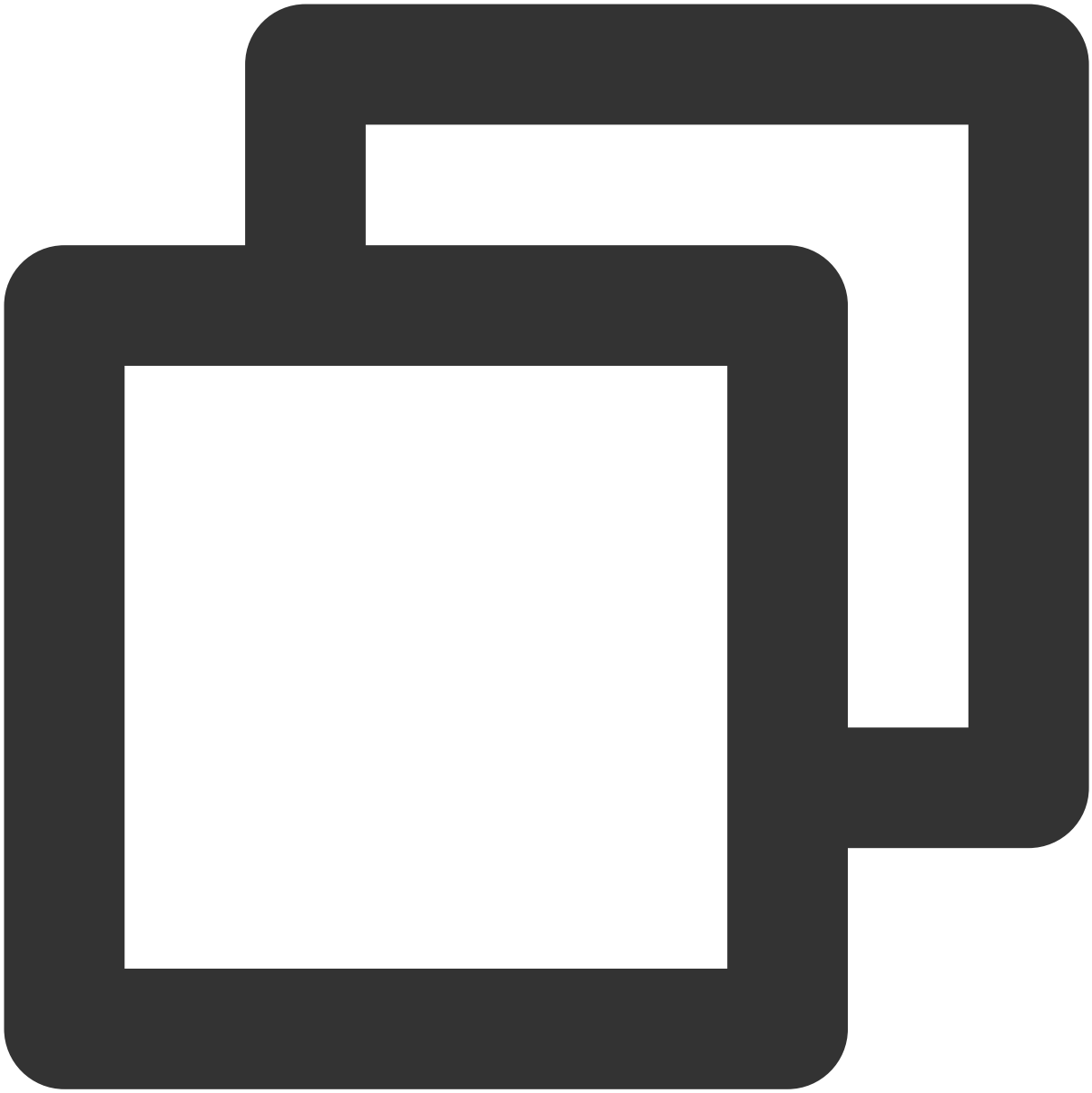
指標名	説明
syn_recv_sock_toa	受信したTOA情報を含む接続の数です。
syn_recv_sock_no_toa	受信したTOA情報を含まない接続の数です。
getname_toa_ok	getsockoptを呼び出してソースIPの取得に成功するとこのカウントが増加するほ

	か、accept関数を呼び出してクライアントリクエストを受信した場合もカウントが増加します。
getname_toa_mismatch	getsockoptを呼び出してソースIPを取得した際、タイプがマッチしないとこのカウントが増加します。例えば、あるクライアント接続内に含まれるものがIPv6アドレスではなくIPv4ソースIPであった場合、このカウントが増加します。
getname_toa_empty	TOAを含まないクライアントファイルディスクリプタがgetsockopt関数を呼び出した場合に、このカウントが増加します。
ip6_address_alloc	TOAカーネルモジュールがTCPデータパケットに保存されたソースIP、ソースPortを取得した際、情報を保存するためのスペースを申請します。
ip6_address_free	接続がリリースされた際、TOAカーネルモジュールはそれまでソースIP、ソースPortの保存に使用していたメモリをリリースします。すべての接続が切断された状態で、全CPUのこのカウント数の合計はip6_address_allocのカウントと同じでなければなりません。

FAQ

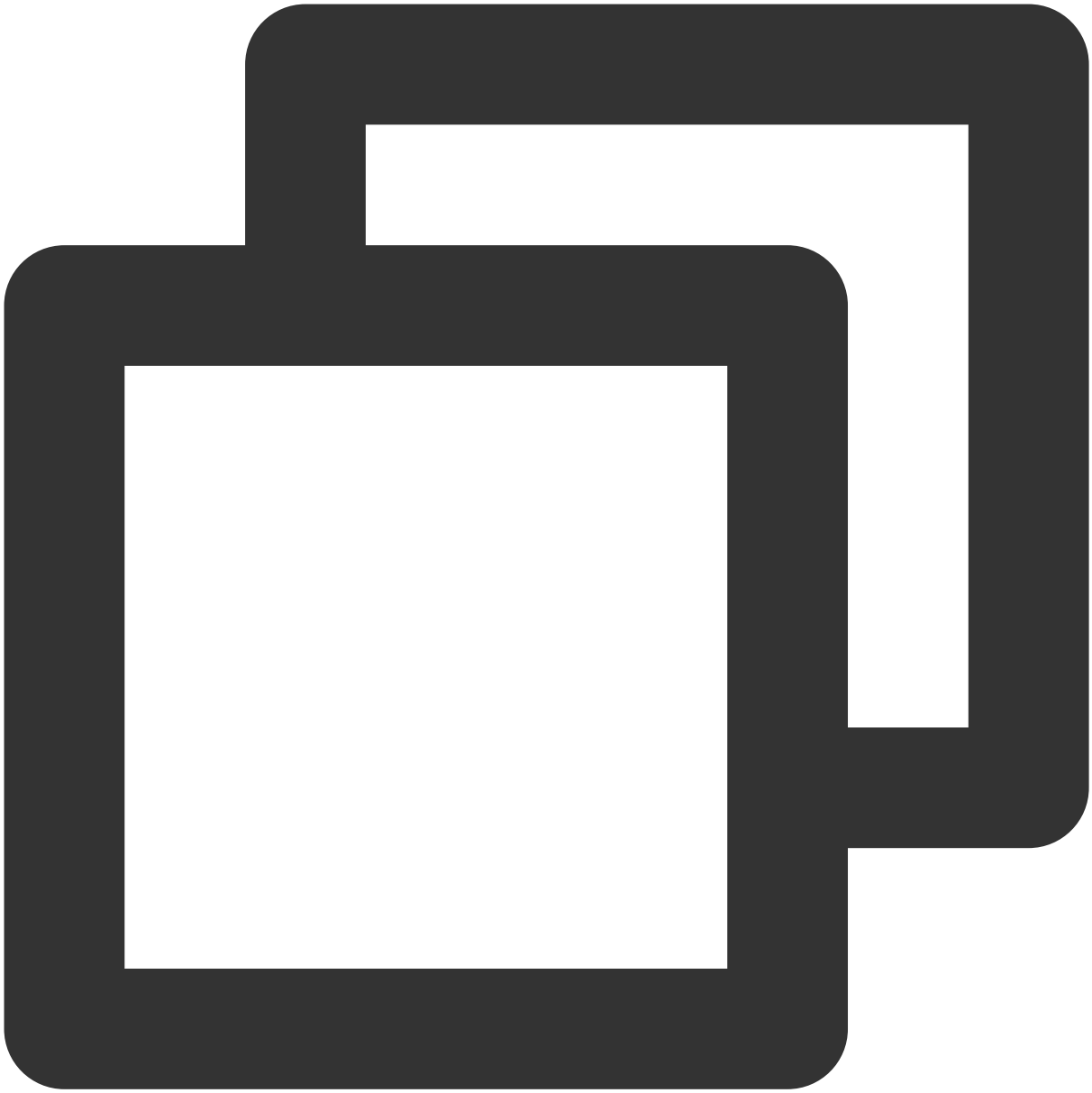
NAT64 CLBシーンにTOAモジュールを挿入すると、サーバープログラムの変更が必要になるのはなぜですか。
これはIPのタイプが変わったことによるものです。ハイブリッドクラウドのデプロイシーンでIPv4のFullnat変換が行われた場合、このシーンではクライアントのリアルソースIPは依然としてIPv4のIPから変換された別のIPv4のIPであるため、IPのタイプに変化はありません。しかし、NAT64 CLBのシーンでは、クライアントのリアルソースIPはIPv6からIPv4に変換されたものであり、IPのタイプが変化しています。このためこれがIPv6のIPであるとサーバーに理解させるためには、IPv6アドレスの意味を理解できるようにサーバープログラムを変更する必要があります。

利用中のシステムがLinuxが発行するバージョンなのか、Tencent TLinuxのカーネルなのかを確認するにはどうすれば良いですか。
次のコマンドを実行してカーネルバージョンを確認できます。実行結果のバージョンに `tlinux` が含まれればTLinuxシステム、そうでなければLinuxディストリビューションです。



```
uname -a
```

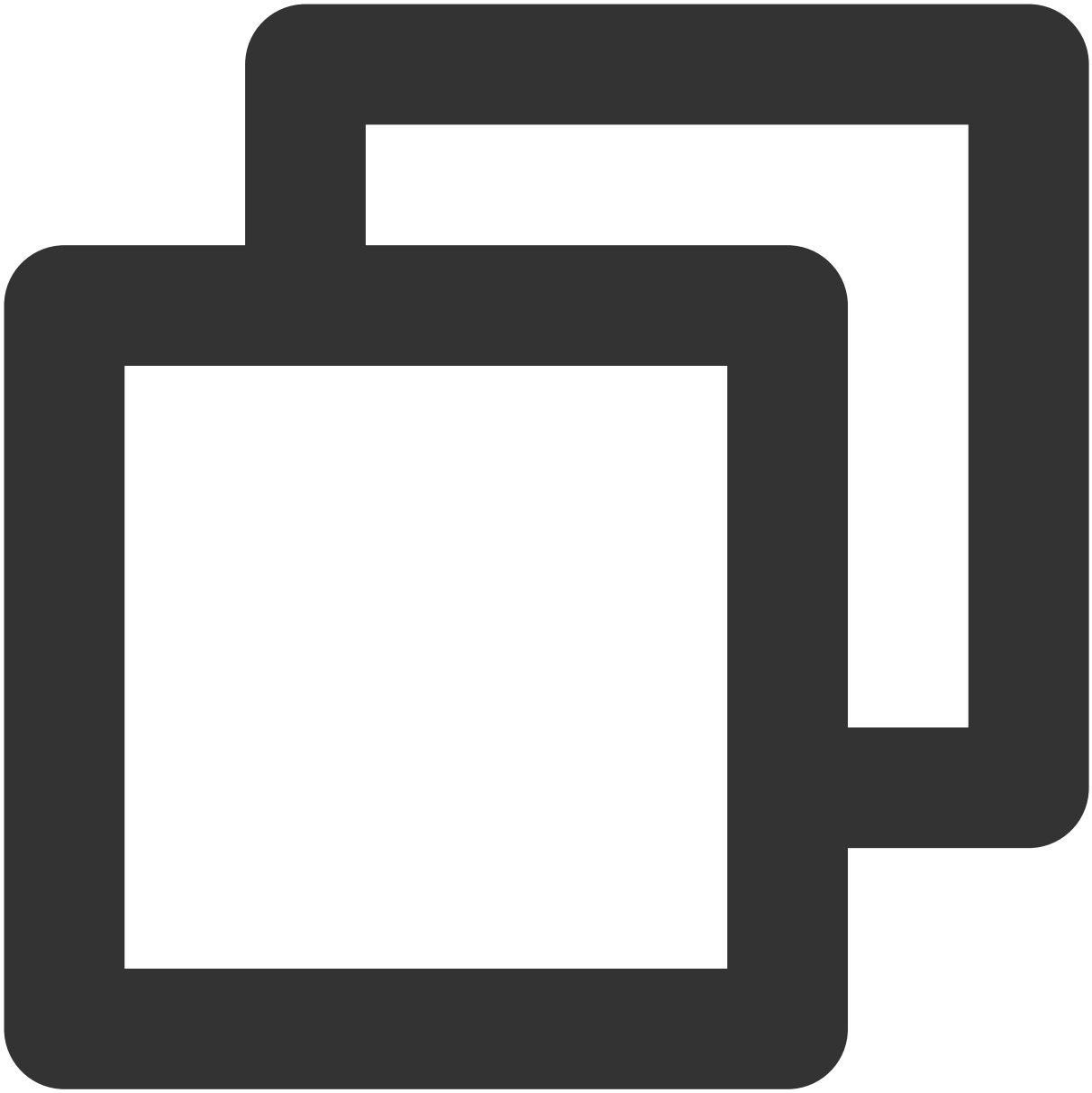
もしくは次のコマンドを実行し、実行結果に `tlinux` または `tl2` が含まれていれば、TLinuxシステムです。



```
rpm -qa | grep kernel
```

ソースアドレスが取得できません。

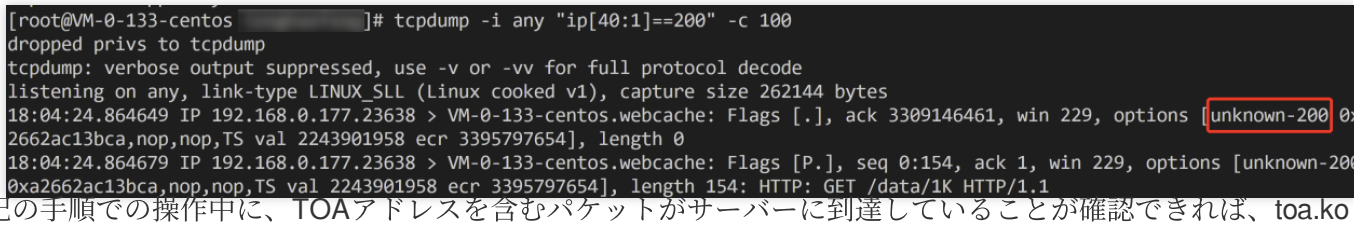
1. **初期のトラブルシューティングはどのように行えばよいですか。**次のコマンドを実行し、TOAモジュールがすでにロードされているかどうかを確認します。



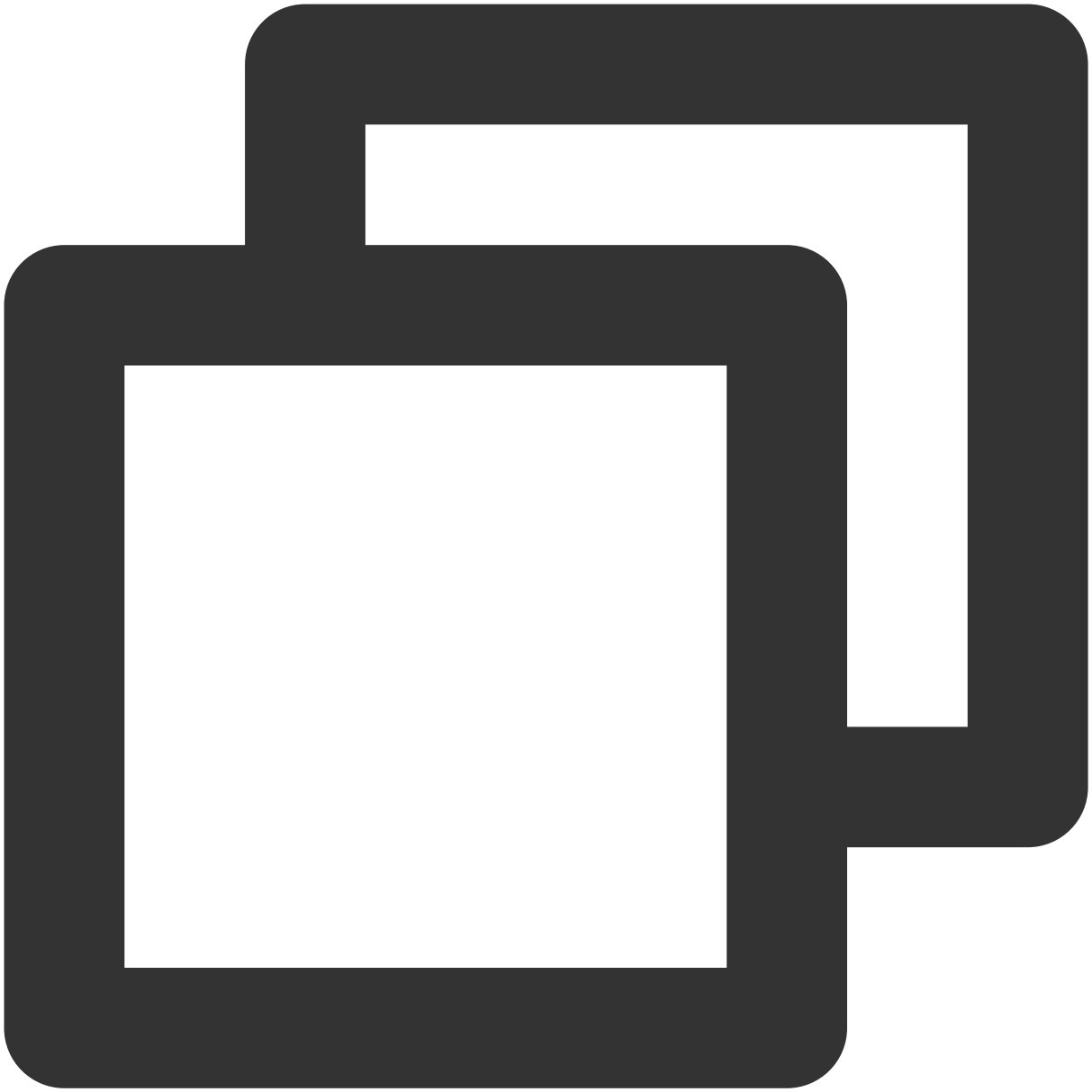
```
lsmod | grep toa
```

2. サーバープログラムがインターフェースを正しく呼び出してソースアドレスを取得しているかどうかを確認します。上記のバックエンドサービスの適用の内容をご参照ください。
3. サーバーでパケットをキャプチャしてトラブルシューティングを行い、リアルソースアドレスを含むTCPパケットが到着しているかどうかを確認します。
tcp optionの中に `unknown-200` が表示されている場合は、SNATを経たリアルソースIPがTCP optionに挿入済みであることを意味しています。

`unknown-253` が表示されている場合は、NAT64シーンでリアルなIPv6のソースIPが挿入済みであることを意味しています。

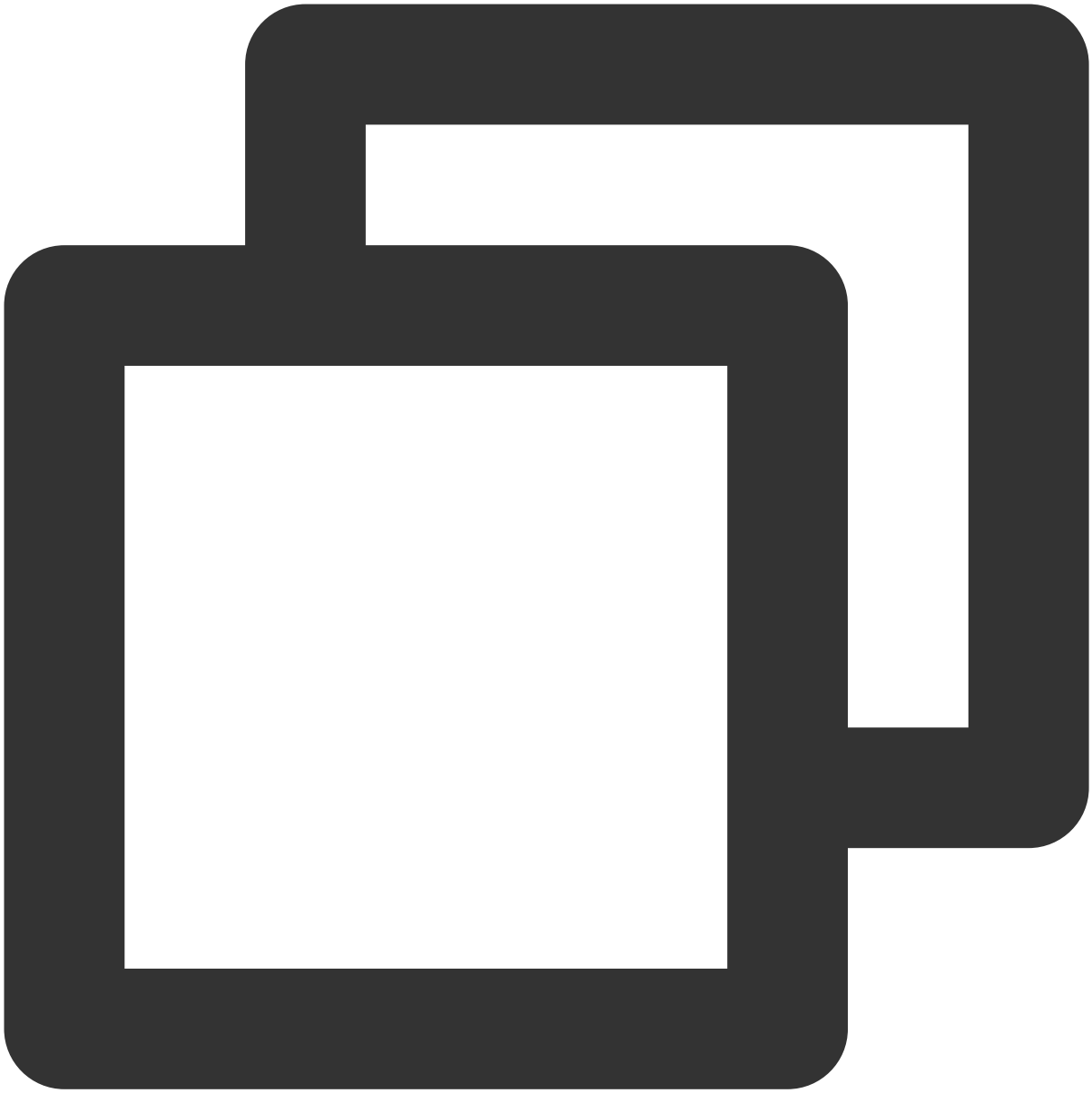


4. 上記の手順での操作中に、TOAアドレスを含むパケットがサーバーに到達していることが確認できれば、toa.koをDEBUGバージョンにコンパイルし、カーネルログによってさらに特定を進めることができます。ダウンロードしたTOAソースコードディレクトリで、MakefileにDEBUGコンパイルオプションを追加します。
5. 次のコマンドを実行して再コンパイルします。



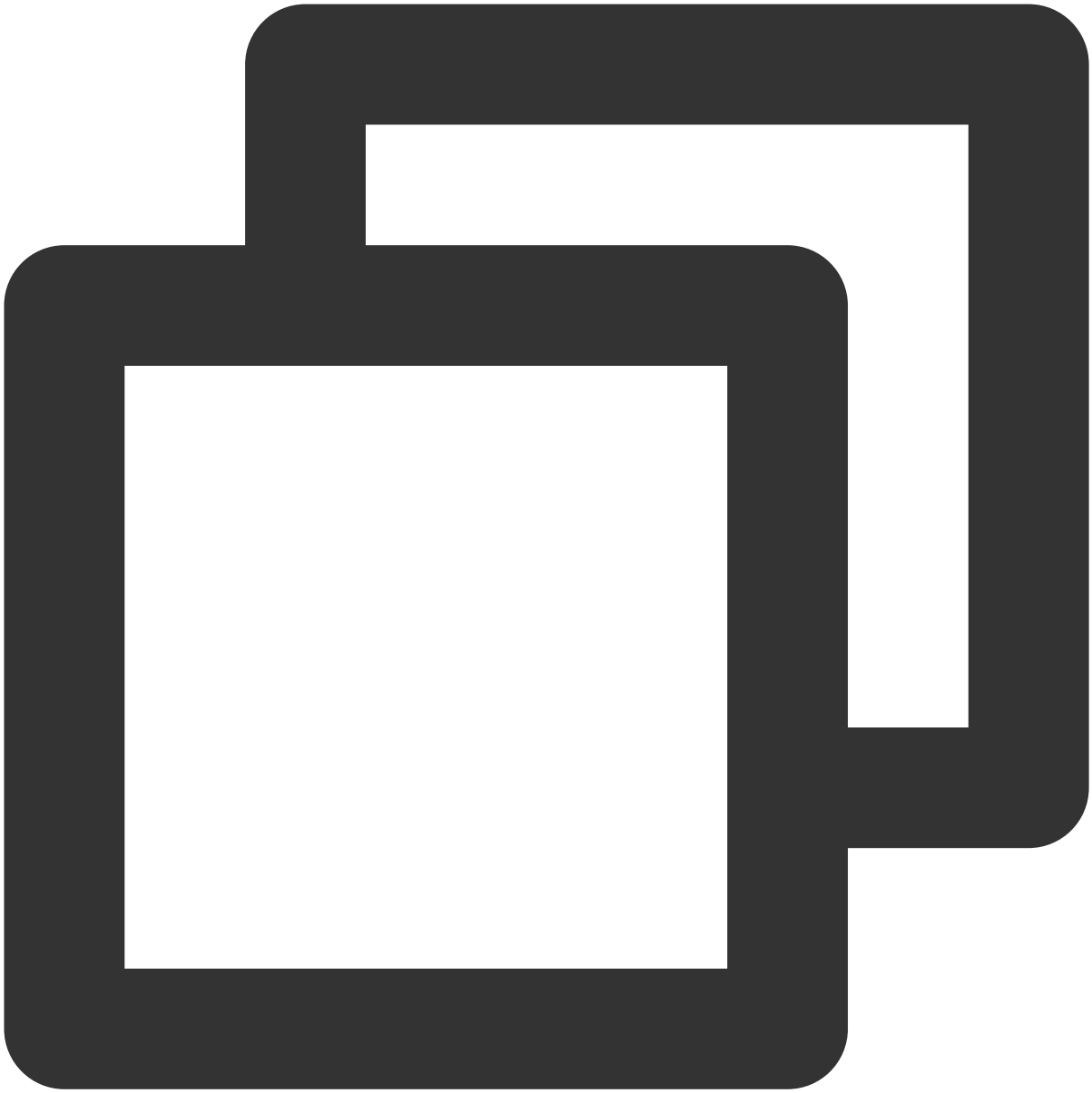
```
make clean
make
```

6. 次のコマンドを実行して既存のkoをアンインストールし、コンパイルした最新のkoを再度挿入します。



```
rmmod toa
insmod ./toa.ko
```

7. 次のコマンドを実行してカーネルログを観察します。



```
dmesg -Tw
```

次の内容が表示された場合、TOAモジュールは正常に動作しています。サーバープログラムがインターフェースを呼び出してリアルソースIPを取得していないか、あるいは間違ったインターフェースを使用していないかのトラブルシューティングをさらに行ってください。

```
[Wed Dec 29 18:07:11 2021] [DEBUG] TOA: inet_getname_toa called, sk->sk_user_data is 000000003088927f
[Wed Dec 29 18:07:11 2021] [DEBUG] TOA: inet_getname_toa: set new sockaddr, ip 192.168.0.177 -> 42.193.59.202,
618
```

8. 上記の手順をすべて行っても具体的な原因が判明しない場合は、 お問い合わせください。

ロードバランサーのモニタリングアラート設定のベストプラクティス

最終更新日：：2024-01-04 17:48:23

ロードバランサーCLBのビジネスモニタリングシステムを改善するために、Tencent Cloud オブザーバビリティプラットフォームのデータ収集およびアラート機能を活用し、一体化された警告メカニズムを構築します。Tencent Cloud オブザーバビリティプラットフォームを使用することで、ロードバランサーCLBのリソース利用状況、パフォーマンス、および動作状態を全面的に把握することができます。関心のあるインスタンスに対してモニタリング及びアラートを設定し、モニタリング指標やイベントのアラートトリガールールを設定することができます。当該するインスタンスのモニタリング指標が異常な場合、アラート通知をタイムリーに受け取り、迅速に障害対応を行うことができます。

使用シナリオ

関心を持つインスタンス指標にアラームを作成し、ロードバランサーCLBインスタンスが特定の条件に達した際に、関心のあるユーザーグループにタイムリーにアラーム情報を送信できます。より便利かつ迅速に予期せぬ状況を把握し、運営維持効率を向上させ、運営維持コストを削減します。
本文では、標準型を例として、パフォーマンス容量型にアップグレードされた公共ネットワークロードバランサーCLB インスタンスにアラームを設定する方法を紹介します。パフォーマンス容量型の仕様については、パフォーマンス容量型仕様の紹介を参照してください。

前提条件

ロードバランサーインスタンスを作成し、リスナーを設定しました。詳細はロードバランサーのクイックスタートを参照してください。
バックエンドサーバーを正常にバインドしていることが必要です。詳細はバックエンドサーバーのバインドを参照してください。
この例では、ターゲットインスタンスがパフォーマンス容量型にアップグレードされている必要があります。詳細はパフォーマンス容量型インスタンスへのアップグレードを参照してください。

基本概念

用語	定義

アラームポリシー	ポリシー名、ポリシータイプ、アラーム対象、トリガー条件、通知テンプレートで構成されます。
ポリシータイプ	アラームポリシーのタイプは、ポリシーの分類を識別するために使用され、タイプはクラウド製品に対応しています。例えば、クラウドサーバーポリシーを選択すると、CPU使用率、ディスク使用率などの指標アラームをカスタマイズできます。
トリガー条件	トリガー条件は、指標、比較関係、閾値、統計粒度、および連続N個の監視データポイントで構成される意味のある条件。
監視タイプ	監視タイプには、クラウド製品監視、フロントエンドパフォーマンス監視が含まれます。
通知テンプレート	複数のポリシーがワンクリックで利用できるテンプレート、様々なシナリオでアラーム通知を受け取るために適用されます。詳細は通知テンプレートの新規作成を参照してください。

指標紹介

パフォーマンス容量型インスタンスが限界超えであるかどうかを判断するコア指標には、クライアントからLBへの同時接続数、クライアントからLBへの新規接続数、リクエスト数／秒、クライアントからLBへの送信帯域幅、クライアントからLBへの受信帯域幅が含まれます。したがって、上記のコア指標の利用率アラーム指標に注目する必要があります。以下の表に示されています。なお、廃棄／利用率監視指標は内部テスト段階にあり、使用をご希望の場合はチケットを提出してください。より多くのアラーム指標の説明については、アラーム指標の説明を参照してください。

ディメンション	アラームポリシータイプ	アラームポリシー	アラーム指標	指標説明
インスタンス	公共ネットワークロードバランサーインスタンス	廃棄／利用率監視	受信帯域幅利用率	統計粒度内で、クライアントが外部ネットワークを通じてロードバランサーを利用する際の帯域幅利用率。
			送信帯域幅利用率	統計粒度内で、ロードバランサーが外部ネットワークを利用する際の帯域幅使用率。
			最大接続数使用率	統計粒度内のある時点で、クライアントからロードバランサーへの同時接続数が、パフォーマンス容量型仕様の同時接続数性能限界に対する利用率。
			新規接続数使用	統計粒度内のある時点で、クラ

			率	イアントからロードバランサーへの新規接続数が、パフォーマンス容量型仕様の新規接続数性能限界に対する利用率。
		QPS関連監視	QPS利用率	統計粒度内のある時点で、ロードバランサーのQPSが、パフォーマンス容量型仕様のQPS性能限界に対する利用率。

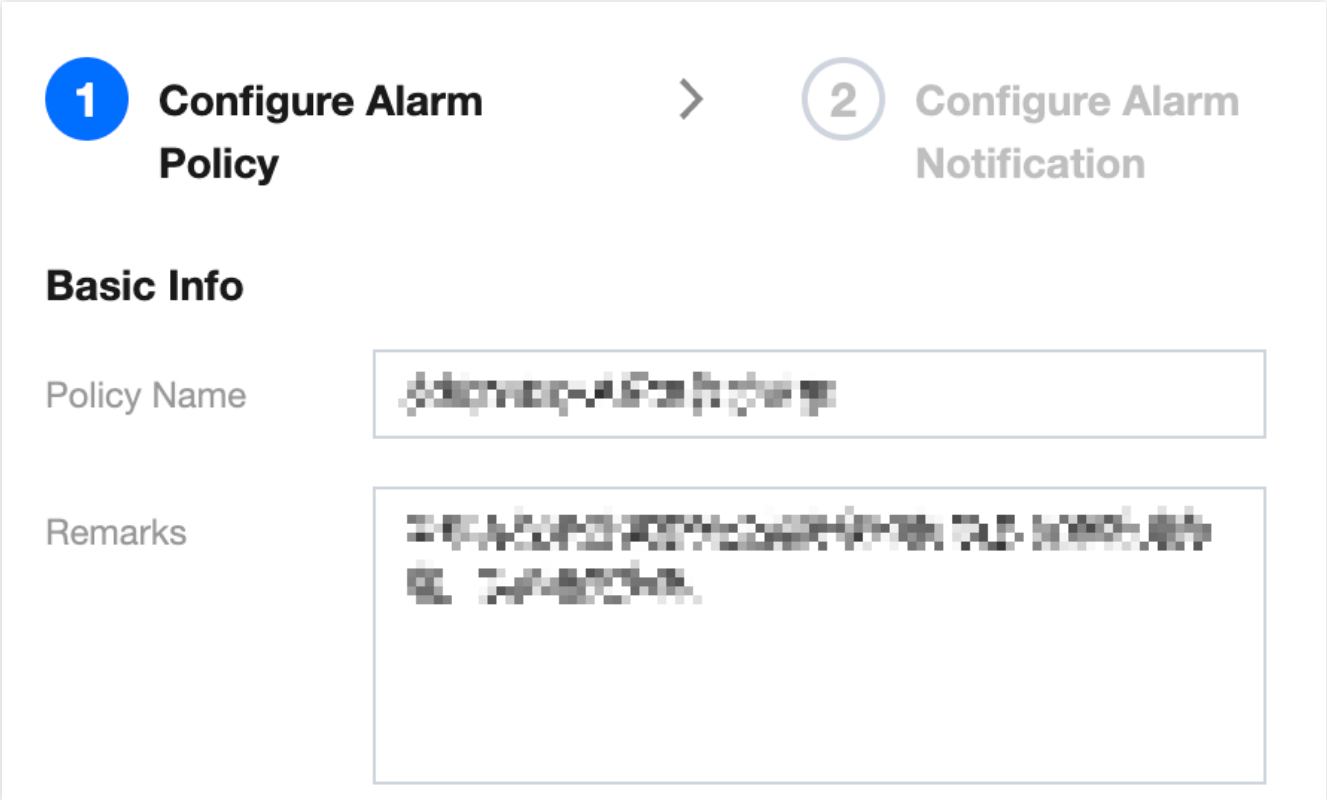
操作手順

1. Tencent Cloud オブザーバビリティプラットフォームにログインします。
2. 左側のナビゲーションバーで、**アラーム管理→アラーム設定→****アラームポリシー**をクリックし、管理ページに入ります。
3. **ポリシーの**新規**作成**をクリックし、以下のオプションを設定します。

3.1 基本情報

ポリシー名：最大60文字までのポリシー名を入力します。

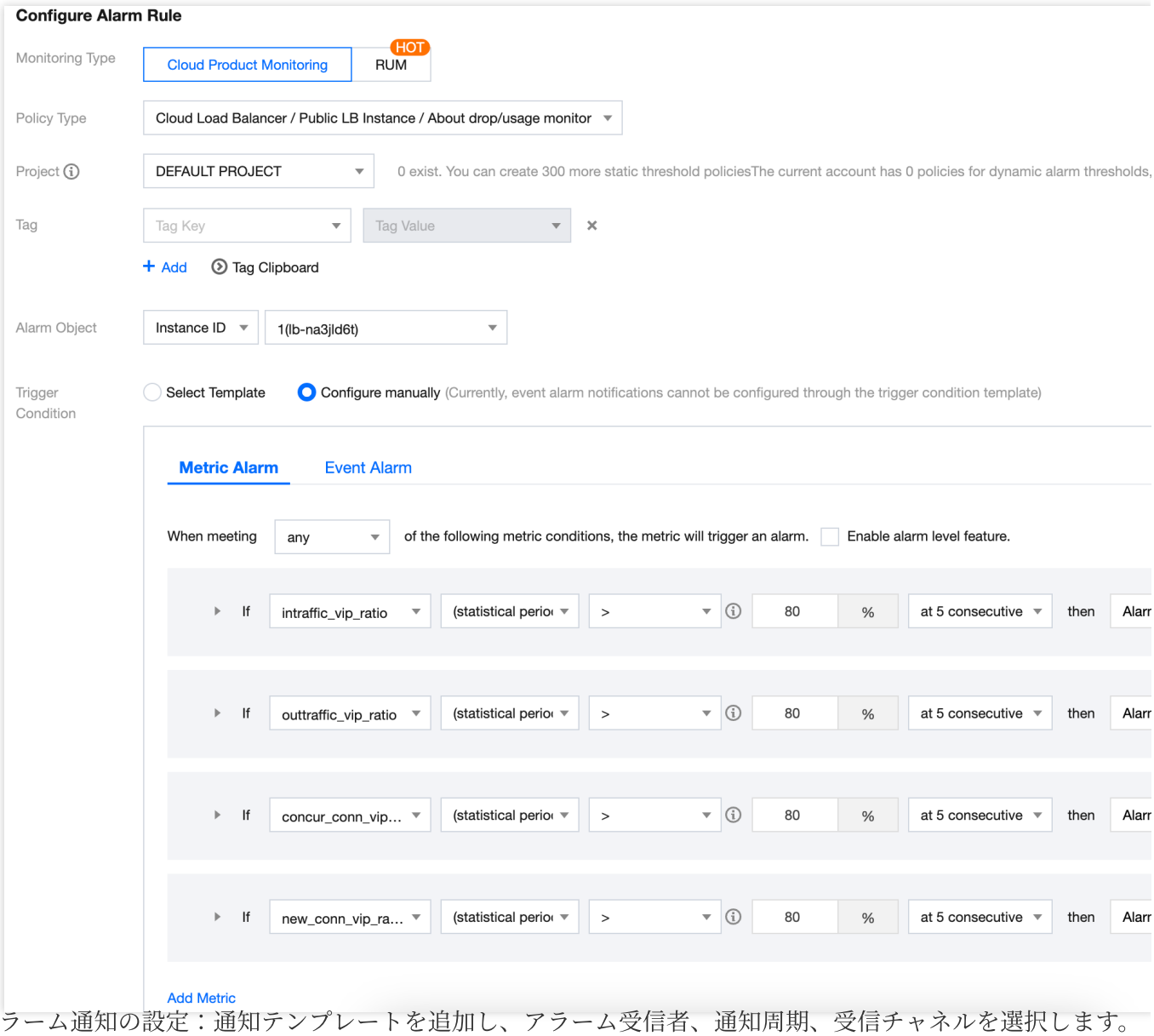
備考：最大100文字までの備考を入力します。



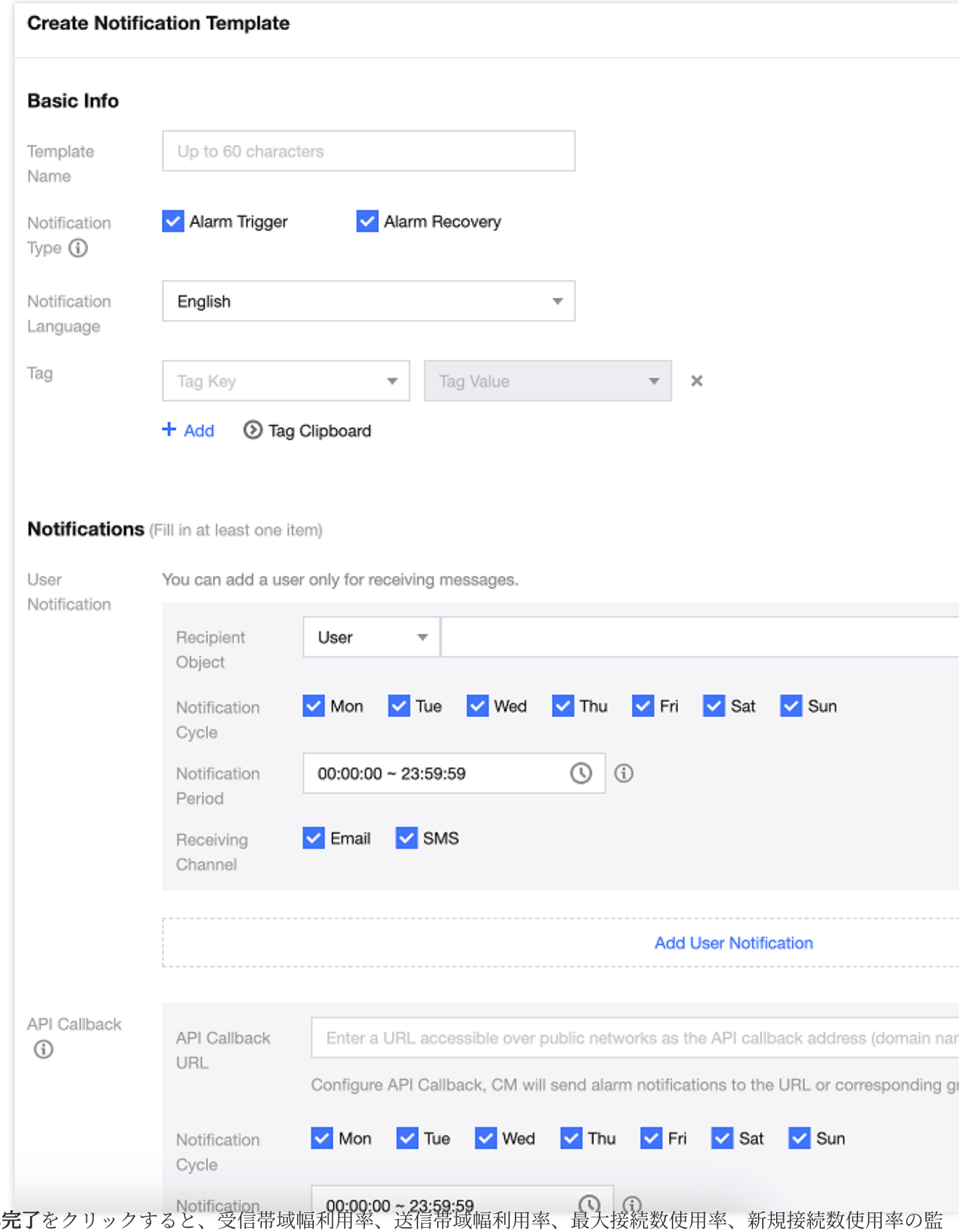
3.2 アラームルールの設定

監視タイプ：クラウド製品監視を選択します。

ポリシータイプ：**ロードバランサー→公共ネットワークロードバランサーインスタンス→****廃棄／利用率監視**を選択します。
ポリシー所属プロジェクト：ポリシー所属プロジェクトを選択します。所属プロジェクトは、アラームポリシーの分類と権限管理に使用され、クラウド製品インスタンスのプロジェクトとは強い結びつきはありません。
所属タグ：ポリシー所属タグを選択します。
アラーム対象：アラーム対象として目標インスタンスを選択します。
トリガー条件：アラーム指標、比較関係、閾値、連続N個の監視データポイント、アラーム頻度で構成される意味のある条件。
例えば、アラーム指標が**受信帯域幅利用率**、比較関係が**より大きい**、閾値が**80%、連続監視データポイントが5つ**、アラーム頻度が**1時間に1回**とします。この場合、あるロードバランサーインスタンスの入帯域幅利用率が連続5回で80%を超えた場合、アラームがトリガーされ、アラームは1時間に1回発行されます。
受信帯域幅利用率、送信帯域幅利用率、最大接続数使用率、新規接続数使用率を設定するために、以下の図に参照してください。

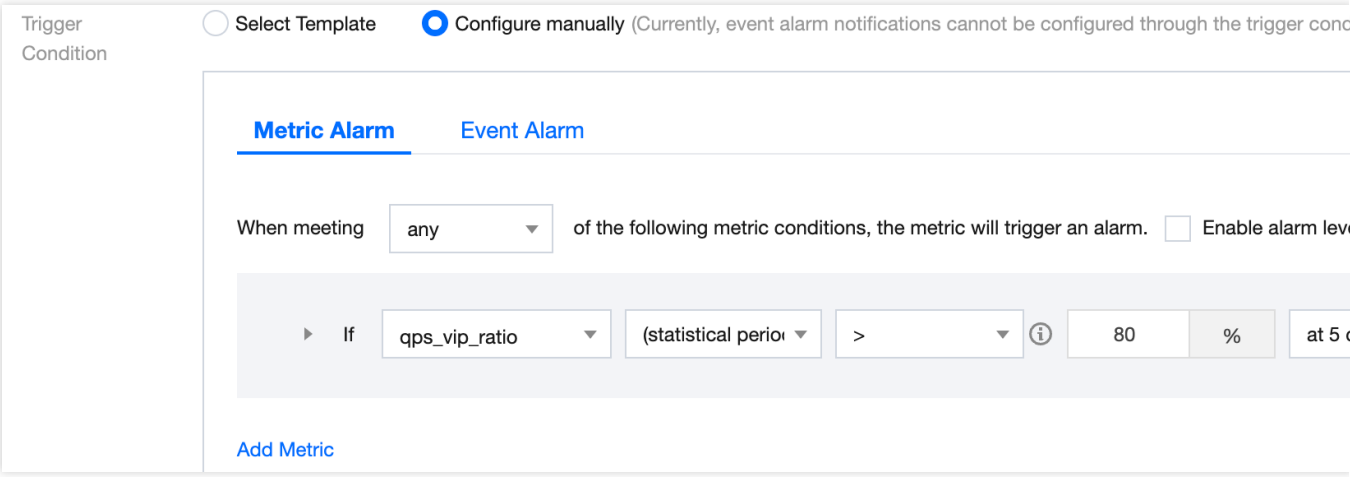


3.3 アラーム通知の設定：通知テンプレートを追加し、アラーム受信者、通知周期、受信チャネルを選択します。通知テンプレートを作成していない場合は、**テンプレート****の新規作成**をクリックして作成し、詳細は通知テンプレートの新規作成を参照してください。



3.4 **3完了**をクリックすると、受信帯域幅利用率、送信帯域幅利用率、最大接続数使用率、新規接続数使用率の監視アラームの設定が完了します。QPS利用率監視アラームについては、上記の手順で新規アラームポリシーを作

成し、ポリシータイプを**ロードバランサー→公共ネットワークロードバランサーインスタンス→QPS関連監視**に変更し、トリガー条件を以下のように設定してください。



解決策

上記のアラームを受け取った場合、業務量が増加しており、現在の標準型のパフォーマンス容量型インスタンスの仕様が性能限界に達し、業務の需要に満たせなくなっています。パフォーマンス容量型インスタンスの規格調整に進んで、業務に影響が出ないようにしてください。

マルチアベイラビリティーゾーンの高可用性設定の説明

最終更新日：：2024-01-04 17:48:23

CLBのマルチアベイラビリティーゾーンにおける高可用性

CLBロードバランサはマルチアベイラビリティーゾーン障害復旧機能をサポートしています。例えば広州3区と広州4区という2つのアベイラビリティーゾーン（同一リージョン）にそれぞれ複数のクラスターをデプロイすることで、同一リージョン下の異なるアベイラビリティーゾーン間での障害復旧を実現します。この機能により、アベイラビリティーゾーン全体に障害が発生した場合も、CLBが10秒以内にフロントエンドのアクセストラフィックを同一リージョン下の異なるアベイラビリティーゾーンに切り替え、サービス機能を復旧させることができます。

具体的なケースとQ&A

質問1：広州3区にCLB test1がある場合、Client側のパブリックネットワークインバウンドトラフィックのポリシーはどのようなものですか。

広州3区と広州4区のデータセンターには1対のIPリソースプールがあり、これらは対等なIPリソースであると理解することができます。2つのクラスターは同等の負荷機能を持ち、3区、4区のどちらがマスタークラスターでどちらがスレーブクラスターかを開発者が知っておく必要はありません。開発者がCLBを購入し、CVMをバインドした際に、2セットのルールが生成されて2つのクラスターに書き込まれます。この時点ですでに高可用性が獲得されています。

質問2：広州3区にCLB test1があり、バックエンドはA、Bのアベイラビリティーゾーンでそれぞれ100台のサーバーをバインドしています。業務の実行中に、それぞれが100万のHTTP長時間接続数を確立しています（TCP接続はオフになっていません）。このとき、広州3区のCLBクラスター全体がダウンして使用できなくなった場合、業務への影響はどうなりますか。

広州3区のCLBがサービス機能を喪失すると、その時点での長時間接続はすべて切断されます。短時間接続には影響はありません。障害復旧アーキテクチャによって10秒以内に広州3区と広州4区の各100台のサーバーが広州4区のCLBに自動的にバインドされ、業務機能がただちに復旧します。手動での介入は必要ありません。

質問3：マルチアベイラビリティーゾーン障害復旧機能はどのタイプのCLBでサポートされていますか。追加料金はかかりますか。

マルチアベイラビリティーゾーン障害復旧は無料のベータ版テスト段階のため、追加料金はかかりません。現在、マルチアベイラビリティーゾーン障害復旧機能はパブリックネットワークCLBでのみサポートしています。パ

ブリックネットワークCLBのマルチアベイラビリティーゾーン障害復旧と異なり、プライベートネットワークCLBでは最寄りのサーバーへのアクセスをデフォルトでサポートしています。

バランシングアルゴリズムの選択と重みの設定の例

最終更新日：：2024-01-04 17:43:01

ロードバランシングアルゴリズムの比較分析

重み付けラウンドロビンアルゴリズム Weighted Round-Robin Scheduling

重み付けラウンドロビンアルゴリズムは、ラウンドロビンの方式によって、リクエストを順に異なるサーバーにスケジューリングするものです。重み付けラウンドロビンスケジューリングアルゴリズムでは、サーバー間でパフォーマンスに違いがある状態を解決することができます。サーバーの処理パフォーマンスをそれに応じた重みの値で表し、重みの高低とラウンドロビンの方式によってリクエストを各サーバーに分配します。重み付けラウンドロビンアルゴリズムでは新規接続数に基づいてスケジューリングを行います。重みが高いサーバーは先に接続を確立することができ、重みが高いほどラウンドロビンの回数が多く（確率が高く）なります。同じ重みのサーバーは同等の接続数を処理します。

メリット：シンプルで実用的であり、現時点のすべての接続ステータスを記録する必要がない、ステートレスなスケジューリングです。

デメリット：相対的にシンプルなため、リクエストのサービス時間の変化が大きい場合や、各リクエストの消費時間が一致しない場合に、サーバー間の負荷がアンバランスになりやすいです。

適用ケース：各リクエストがバックエンドを占有する時間が基本的に同じ場合に、負荷の状態が最適になります。HTTPなどの短時間接続サービスによく用いられます。

ユーザーへの推奨事項：各リクエストのバックエンド占有時間が基本的に同じであり、バックエンドサーバーが処理するリクエストタイプが同一または類似している場合は、重み付けラウンドロビン方式を選択することをお勧めします。リクエスト時間の差があまりない場合も重み付けラウンドロビン方式を使用することをお勧めします。この実現方式は低消費かつトラバーサルの必要がなく、高効率なためです。

重み付け最小接続 Weighted Least-Connection Scheduling

原理

実際の状況では、クライアントの各サービスリクエストがサーバーにとどまる時間には比較的大きな差異があります。シンプルなラウンドロビンやランダムなバランシングアルゴリズムを用いた場合、動作時間が長くなるに従って、各サーバー上の接続プロセス数に大きな違いが生じるようになり、これでは実際には負荷分散の真の効果を得ることができません。最小接続スケジューリングは一種の動的スケジューリングアルゴリズムであり、サーバーのその時点でアクティブな接続数によってサーバーの負荷状況を評価するもので、ラウンドロビンスケジューリングアルゴリズムとは異なります。スケジューラーが各サーバーの確立した接続数を記録する必要があり、あるサーバーにリクエストがスケジューリングされると接続数に1をプラスし、接続が中断またはタイムアウトになると、接続数から1をマイナスします。重み付け最小接続スケジューリングアルゴリズムは最小接続スケ

ジューリングアルゴリズムをベースに、サーバーの処理能力に応じて各サーバーに異なる重みを割り当て、各サーバーがその重みに応じた数のサービスリクエストを受け付けることができるようにするもので、最小接続スケジューリングアルゴリズムをベースに改善を加えたものです。
1.1 仮に各RSの重みを順にWi（i=1...n）とし、現在の接続数を順にCi（i=1...n）とした場合、Ci/Wiの値が最小のRSが、次に割り当てられるRSとなります。
1.2 Ci/Wiと同一のRSが存在する場合、これらのRSにはさらに重み付けラウンドロビン方式を使用してスケジューリングを行います。

メリット

このタイプのアルゴリズムは、FTPなどのアプリケーションのような、処理時間の長いリクエストサービスに適しています。

デメリット

ポートの制限により、現在最小接続とセッション維持機能を同時に有効にすることはできません。

適用ケース

各リクエストがバックエンドを占有する時間の差が比較的大きいケースです。長時間接続サービスによく用いられます。

ユーザーへの推奨事項

ユーザーがさまざまなリクエストを処理する必要があり、かつリクエストがバックエンドを占有する時間の差が比較的大きい場合（例えば3ミリ秒と3秒のように、単位レベルの違いがある場合など）では、重み付け最小接続アルゴリズムを使用して負荷分散を実現することをお勧めします。

ソースIPハッシュスケジューリングアルゴリズム ip_hash

原理

リクエストのソースIPアドレスをハッシュキー（Hash Key）として、静的に割り当てられたハッシュテーブルから対応するサーバーを見つけます。そのサーバーが使用可能であり、かつオーバーロードになっていない場合はリクエストがそのサーバーに送信され、そうではない場合は空が返されます。

メリット

ip_hashでは一部のセッション維持の効果が実現できます。ソースIPを記憶することで、あるclientリクエストを、hashテーブルによって同一のrs上に一貫してマッピングできます。このため、セッション維持がサポートされていないシーンでも、ip_hashを使用したスケジューリングが可能です。

ユーザーへの推奨事項

リクエストのソースアドレスに対しhash計算を行い、バックエンドサーバーの重みに応じて、リクエストをマッチしたサーバーに転送することで、同一のクライアントIPのリクエストが常に特定のサーバーに転送されるようになります。この方式はCLBのCookie機能を持たないTCPプロトコルに適しています。

バランシングアルゴリズムの選択と重みの設定の例

CLBの近日中にリリース予定の新機能では、**レイヤー7転送での最小接続バランシング方式のサポート**を開始します。ユーザーのRSクラスターがさまざまなシナリオの下で安定して業務を担うことができるよう、CLBの選択と重みの設定の例を参考までにいくつかご紹介します。

シナリオ1

同一の設定（CPU/メモリ）のRSが3台あるとします。パフォーマンスが同じであるため、ユーザーはRSの重みをすべて10に設定することができます。現在、各RSとclientの間に100のTCP接続を確立しており、さらにRSを1台増設するとします。このシナリオでは、最小接続のバランシング方式の使用をお勧めします。この設定で速やかに4台目のRSの負荷を増大させ、他の3台のRSの負荷を低減することができます。

シナリオ2

ユーザーがクラウドサービスを初めて使用する場合で、なおかつウェブサイト構築からあまり時間が経っておらず、サイトの負荷が比較的小さい場合は、同一の設定のRSの購入をお勧めします。RSはすべて同様のアクセス層サーバーだからです。このシナリオでは、ユーザーはRSの重みをすべて10に設定し、重み付けラウンドロビンのバランシング方式によってトラフィックを振り分けることができます。

シナリオ3

ユーザーは単純な静的ウェブサイトへのアクセスを担う5台のサーバーを持っており、5台のサーバーのコンピューティング能力の比率は9：3：3：3：1（CPU、メモリ換算）です。このシナリオでは、RSの重みの割合を、順に90、30、30、30、10に設定することができます。静的ウェブサイトへのアクセスの大半は短時間接続のリクエストであるため、重み付けラウンドロビンのバランシング方式を使用して、RSのパフォーマンス比率に従ってCLBにリクエストを分配させることができます。

シナリオ4

あるユーザーは、大量のWebアクセスリクエストの処理を担う10台のRSを持っていますが、RS増設のための追加支出は望んでいません。あるRSはオーバーロードのために頻繁にサーバー再起動が発生しているとします。このシナリオでは、RSのパフォーマンスに応じた重みを設定し、オーバーロードになっているRSに低い重みを設定することをユーザーにお勧めします。このほか、最小接続のロードバランシング方式を用いて、リクエストをアクティブ接続数が比較的少ないRSに分配することで、問題のRSのオーバーロードを解決することもできます。

シナリオ5

あるユーザーは、いくつかの長時間接続リクエストの処理に用いる3台のRSを持っており、これらの3台のサーバーのコンピューティング能力の比率は3：1：1（CPU、メモリ換算）です。この場合、パフォーマンスが最も良好なサーバーが多くのリクエストを処理しますが、ユーザーはこのサーバーがオーバーロードにならないように、新しいリクエストをアイドル状態のサーバーに分配したいと考えています。このシナリオでは、最小接続のバランシング方式を使用し、かつビジーなサーバーの重みを適宜低下させることで、CLBがリクエストをアクティブ接続数の比較的少ないRSに分配し、負荷分散を実現できるようにすることが可能です。

シナリオ6

あるユーザーは、後続のクライアントリクエストを同一のサーバー上に分配したいと考えています。重み付けラウンドロビンまたは重み付け最小接続の方式を用いると、同一のクライアントからのリクエストを固定のサーバーに転送することが保証されません。顧客の特定のアプリケーションサーバーのニーズに合わせるため、クライアントのセッションの「粘着性」あるいは「継続性」を保証する必要があります。このシナリオでは、ip_hashのバランシング方式を用いてトラフィックを振り分けることができます。この方法では、同一のクライアントからのリ

クエストが常に同一のRSに振り分けられるようにすることが可能です（サーバー数に変化があった場合またはこのサーバーが使用不能になった場合を除きます）。

重みを0に設定することとRSのバインド解除の違い

重みを0に設定：TCPリスナーの既存の接続は転送を継続し、UDPリスナーは同一の5つ組による転送を継続し、HTTP/HTTPSリスナーの既存の接続は転送を継続します。
RSのバインド解除：TCP/UDPリスナーの既存の接続は転送を停止し、HTTP/HTTPSリスナーの既存の接続は転送を継続します。

関連ドキュメント

バックエンドサービスの重みの変更

CLBのリスニングドメイン名に対してWebセキュリティ保護を実行するようにWAFを設定する

最終更新日：：2024-01-04 17:48:23

CLBタイプのWeb Application Firewall(WAF)はドメイン名とCLBリスナーをバインドし、CLBリスナーを経由するHTTPまたはHTTPSトラフィックのチェックとブロックを実現するものです。ここではCLBタイプのWAFによって、CLBに追加されたドメイン名のWebセキュリティ保護を行う方法についてご説明します。

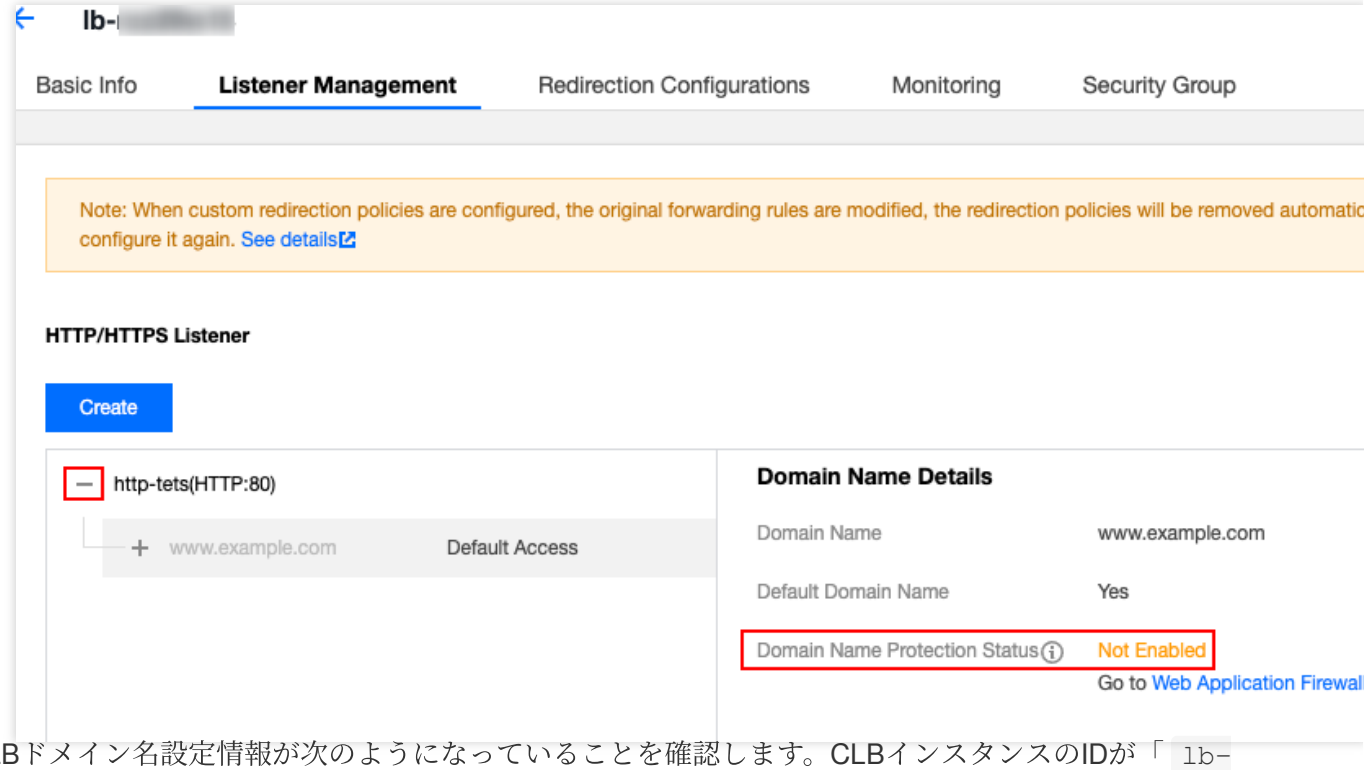
前提条件

HTTPリスナーまたはHTTPSリスナーを作成済みであり、ドメイン名に正常にアクセスできること。操作の詳細については、CLBクイックスタートをご参照ください。
CLBタイプのWAFを購入済みであること。購入方法については購入方法をご参照ください。

操作手順

ステップ1：CLBドメイン名設定の確認

ここではドメイン名 `www.example.com` を保護する場合を例にとります。
1. CLBコンソールにログインし、左側ナビゲーションバーの**インスタンス管理**をクリックします。
2. **インスタンス管理**ページで所在リージョンを選択し、インスタンスリストで対象のインスタンスの右側にある「操作」列の**リスナーの設定**をクリックします。
3. **リスナー管理**タブの**HTTP/HTTPSリスナー**エリアで、対象のリスナーの左側にある**+**をクリックしてドメイン名の詳細を確認します。

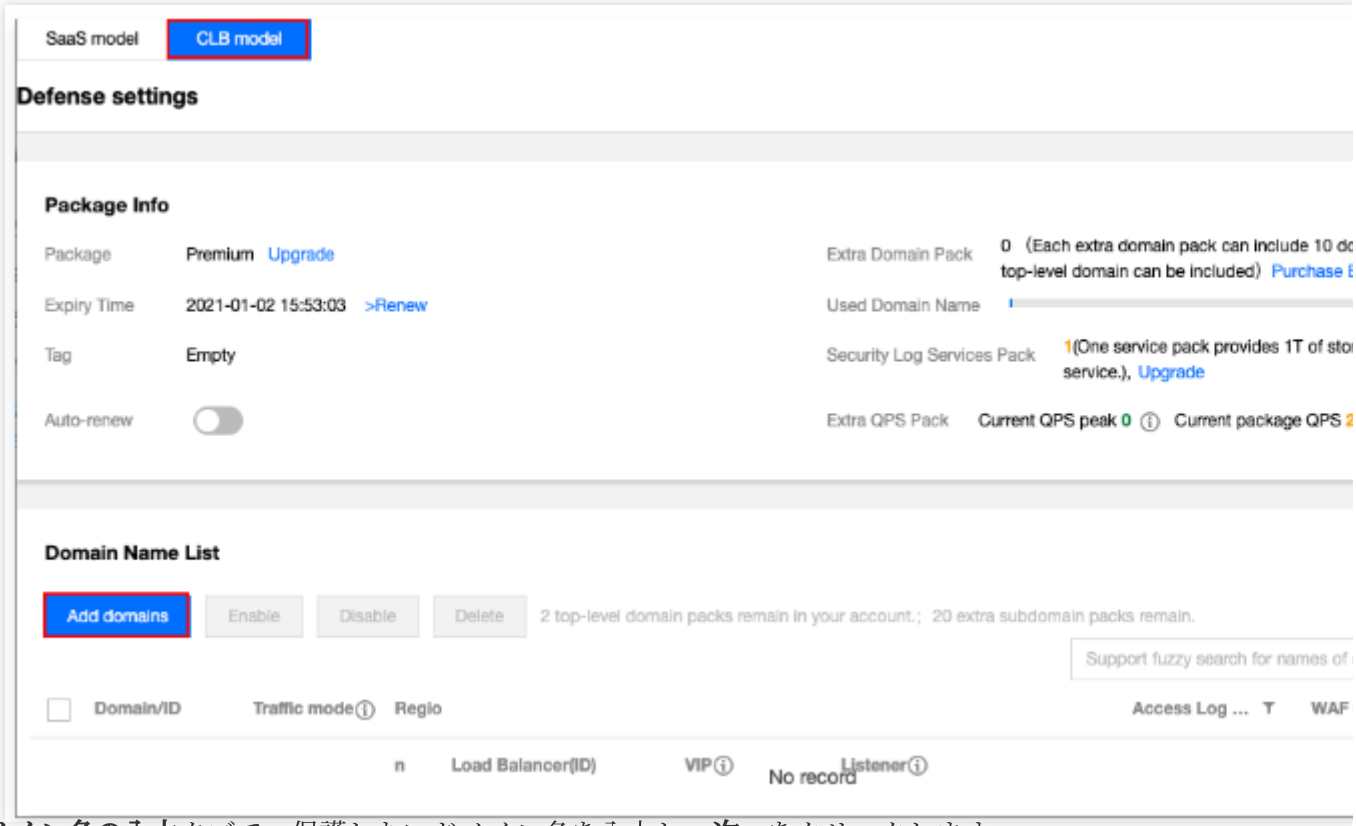


4. CLBドメイン名設定情報が次のようになっていることを確認します。CLBインスタンスのIDが「`lb-f8lm****`」、リスナー名が「http-test」、リスナー転送ルールでリスニングするドメイン名が`www.example.com`、ドメイン名の保護ステータスが「無効」（ID、名前、ドメイン名はすべて実際のものに準じてください）。

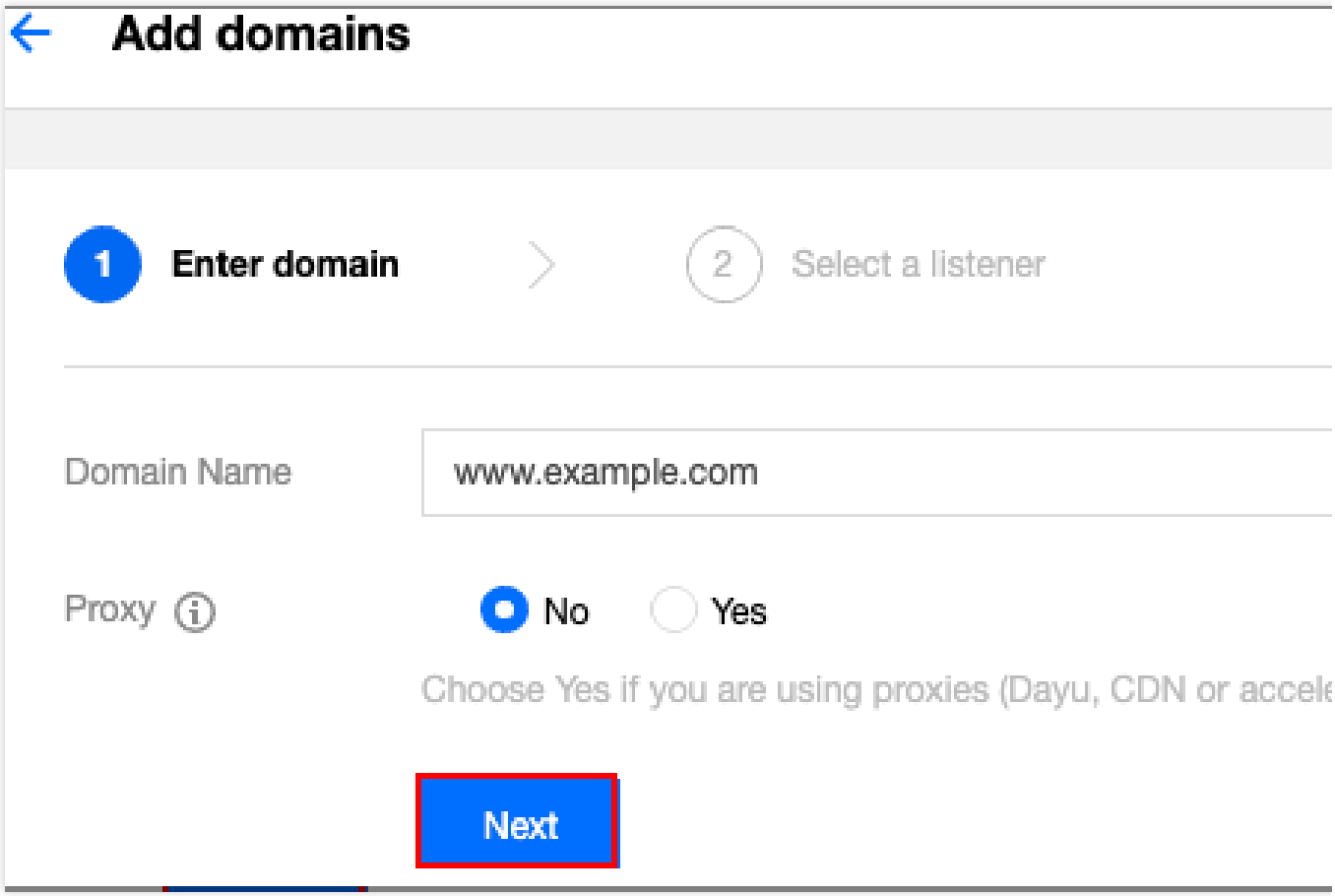
ステップ2：WAFにドメイン名を追加し、CLBにバインド

CLBタイプのWAFが保護対象のドメイン名を認識できるようにするためには、CLBがリスニングするドメイン名をWAFに追加し、CLBリスナーにバインドする必要があります。

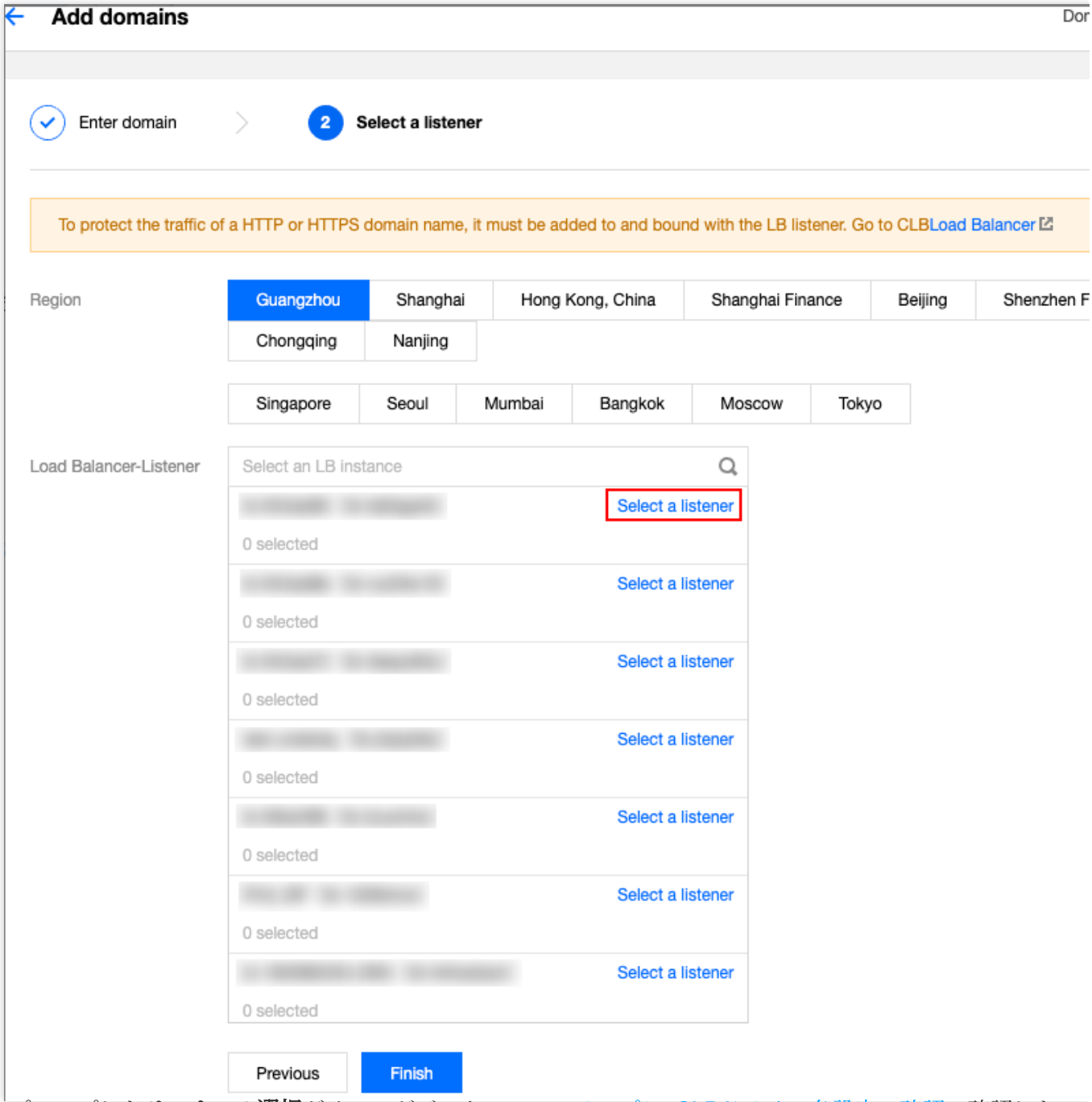
1. WAFコンソールにログインし、左側ナビゲーションバーで**Web Application Firewall** > **保護設定**を選択します。
2. **保護設定**ページで、**CLBタイプ**タブをクリックします。
3. **CLB**タブで、**ドメイン名の追加**をクリックします。



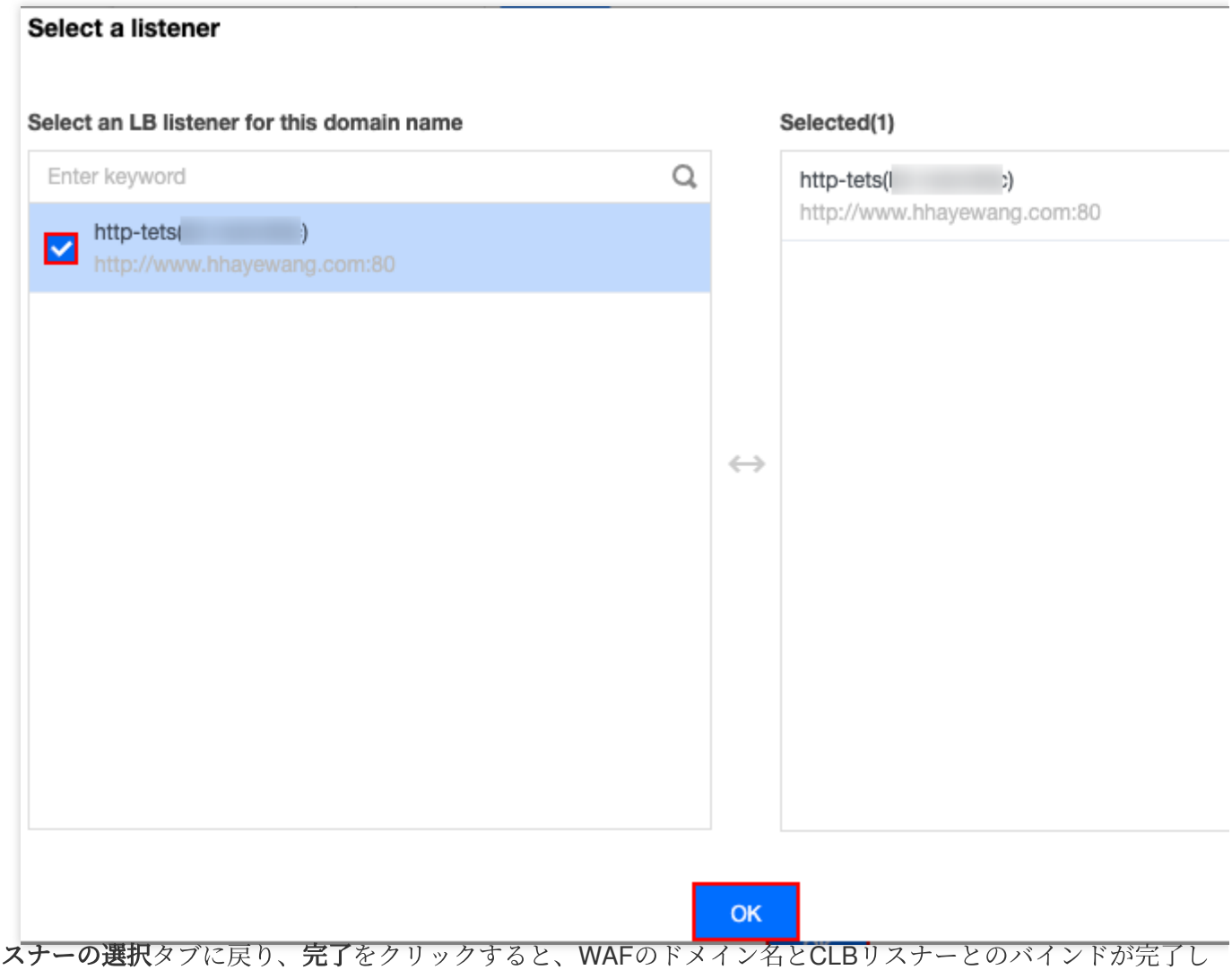
4. **ドメイン名の入力**タブで、保護したいドメイン名を入力し、**次へ**をクリックします。



5. リスナーの選択タブで、CLBの対応するリージョンを選択し、ステップ1：CLBドメイン名設定の確認で確認したCLBインスタンスを選択し、リスナーの選択をクリックします。



6. ポップアップした**リスナーの選択**ダイアログボックスで、ステップ1：CLBドメイン名設定の確認で確認したCLBインスタンスを選択し、**OK**をクリックします。



7. **リスナーの選択**タブに戻り、**完了**をクリックすると、WAFのドメイン名とCLBリスナーとのバインドが完了します。

8. **ドメイン名リスト**ページに戻り、ドメイン名、リージョン、バインドしたCLBインスタンスIDおよびリスナーなどの情報が正しいかどうか確認します。

ステップ3：結果の検証

1. ステップ1：CLBドメイン名設定の確認の操作手順を参照し、**リスナー管理**タブで、表示されたドメイン名の保護ステータスが「有効」、トラフィックモードが「イメージモード」になっていれば、ドメイン保護は有効化されています。

ドメイン名にDNS解決を設定していない場合は、WAFクイックスタートのステップ2：ローカルテストを参照し、WAFによる保護が有効かどうかを検証することができます。

ドメイン名がDNS解決済みであれば、WAFによる保護が有効かどうかの検証は次の手順に従って行うことができます。

2. ブラウザに、アドレス `http://www.example.com/?test=alert(123)` を入力してアクセスします。

3. WAFコンソールにログインし、左側ナビゲーションバーで **攻撃ログ**を選択します。

4. **ログの照会**タグで、保護対象のドメイン名 `www.example.com` を選択して追加し、**照会**をクリックします。ログリストの中に、攻撃タイプが「XSS攻撃」のログがあった場合は、WAFがCLBに設定したドメイン名保護が有効になっていることを意味します。