第 3 版開發人員指南

適用於 PHP 的 AWS SDK

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

適用於 PHP 的 AWS SDK: 第 3 版開發人員指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 適用於 PHP 的 AWS SDK？ ... 1
- 開始使用 SDK ... 1
- 其他資源 ... 1
- API 文件 ... 2
- 開發套件主要版本的維護與支援 ... 2

開始使用 ... 3
- 先決條件 ... 3
  - 要求 ... 3
  - 建議 ... 3
  - 相容性測試 ... 4
- 安裝 ... 4
  - 透過 Composer 安裝 適用於 PHP 的 AWS SDK 做為相依性 ... 5
  - 使用封裝的 phar 安裝 ... 6
  - 使用 ZIP 檔案安裝 ... 6
- 使用 驗證 AWS ... 7
  - 使用 IAM Identity Center 設定身分驗證以進行本機開發 ... 7
  - 啟動 AWS 存取入口網站工作階段 ... 8
  - 進一步了解身分驗證 ... 8
- 建立簡單的應用程式 ... 9
  - 先決條件 ... 9
  - 在您的程式碼中包含 SDK ... 9
  - 撰寫程式碼 ... 9
  - 執行程式 ... 10
  - 後續步驟 ... 10
- AWS Cloud9 搭配 SDK 使用 ... 11
  - 步驟 1：設定您的 AWS 帳戶 以使用 AWS Cloud9 ... 11
  - 步驟 2：設定您的 AWS Cloud9 開發環境 ... 12
  - 步驟 3：設定 適用於 PHP 的 AWS SDK ... 12
  - 步驟 4：下載範例程式碼 ... 13
  - 步驟 5：執行範例程式碼 ... 14

設定服務用戶端 ... 15
- 外部用戶端組態 ... 15
  - 用戶端組態的組態提供者鏈結 ... 15
  - 建立使用外部設定設定的 服務用戶端 ... 16

適用於 PHP 的 AWS SDK 第 3 版環境變數 17
程式碼中的用戶端組態 17
程式碼中的基本組態 17
使用 Sdk類別 18
建構器選項 19
AWS 區域 45
區域解析鏈 45
最佳實務 46
登入資料提供者 47
第 3 適用於 PHP 的 AWS SDK 版中的登入資料提供者是什麼？ 47
了解第 3 適用於 PHP 的 AWS SDK 版中的預設登入資料提供者鏈結 48
第 3 適用於 PHP 的 AWS SDK 版中的內建登入資料提供者 49
在適用於 PHP 的 SDK 中鏈結登入資料提供者 58
建立自訂登入資料提供者以搭配適用於 PHP 的 SDK 使用 59
在適用於 PHP 的 SDK 中記住登入資料 60
擔任 IAM 角色 60
在適用於 PHP 的 SDK AWS STS 中使用 的臨時登入資料 67
在適用於 PHP 的 SDK 中建立匿名用戶端 69
使用 AWS CRT 延伸模組 69
我需要 AWS CRT 延伸模組嗎？ 70
如何安裝 AWS CRT 延伸模組？ 70
使用開發套件 71
提出 AWS 服務 請求 71
SDK 請求工作流程概觀 71
建立基本服務用戶端 71
提出要求 72
使用結果物件 74
命令物件 75
非同步程式設計 82
Promise 83
處理錯誤 89
同步錯誤處理 89
非同步錯誤處理 91
處理常式和中介軟體 92
處理常式 92
中介軟體 94

建立自訂處理常式 ... 101
串流 ... 102
串流裝飾項目 ... 102
分頁 ... 107
分頁程式物件 ... 107
從結果列舉資料 ... 107
非同步分頁 ... 108
等待程式 ... 109
等待程式組態 ... 110
非同步等待 ... 111
JMESPath 表達式 ... 112
從結果擷取資料 ... 113
從分頁程式擷取資料 ... 117
呼叫 AWS 服務 ... 118
使用功能和選項 ... 118
Amazon DynamoDB ... 118
Amazon S3 ... 124
含指引的程式碼範例 ... 144
登入資料 ... 144
Amazon CloudFront 範例 ... 145
Amazon CloudSearch ... 173
Amazon CloudWatch 範例 ... 175
Amazon EC2 範例 ... 200
Amazon OpenSearch Service ... 213
AWS Identity and Access Management 範例 ... 214
AWS Key Management Service ... 238
Kinesis 範例 ... 260
AWS Elemental MediaConvert ... 276
Amazon S3 範例 ... 282
AWS Secrets Manager ... 315
Amazon SES 範例 ... 324
Amazon SNS 範例 ... 354
Amazon SQS 範例 ... 373
Amazon EventBridge ... 385
程式碼範例 ... 387
API Gateway ... 388

動作 388
案例 393
Aurora 394
案例 393
Auto Scaling 395
基本概念 396
動作 388
Amazon Bedrock 411
動作 388
Amazon Bedrock 執行時間 412
案例 393
Amazon Nova 414
Amazon Titan Image Generator 416
Anthropic Claude 417
Stable Diffusion 419
Amazon DocumentDB 420
無伺服器範例 420
DynamoDB 422
基本概念 396
動作 388
案例 393
無伺服器範例 420
Amazon EC2 454
動作 388
AWS Glue 459
基本概念 396
動作 388
IAM 479
基本概念 396
動作 388
Kinesis 496
無伺服器範例 420
AWS KMS 500
基本概念 396
動作 388
Lambda 536

基本概念 396
動作 388
案例 393
無伺服器範例 420
Amazon MSK 565
無伺服器範例 420
Amazon RDS 567
動作 388
案例 393
無伺服器範例 420
Amazon RDS 資料服務 575
案例 393
Amazon Rekognition 576
案例 393
Amazon S3 577
基本概念 396
動作 388
案例 393
無伺服器範例 420
S3 目錄儲存貯體 600
基本概念 396
Amazon SES 616
案例 393
Amazon SNS 617
動作 388
案例 393
無伺服器範例 420
Amazon SQS 638
無伺服器範例 420
從第 2 版遷移 642
簡介 642
第 3 版有哪些新功能？ 642
分離式 HTTP 層級 642
非同步請求 82
與第 2 版有何不同？ 643
專案相依性已更新 643

現在需要區域和版本選項 643
用戶端執行個體化使用 建構函數 644
用戶端組態已變更 645
某些 API 結果已變更 645
列舉類別已移除 648
已移除精細的例外類別 649
靜態外觀類別已移除 649
分頁程式取代疊代運算 649
許多更高層級的抽象概念已變更 650
比較兩種 SDK 版本的程式碼範例 651
範例：Amazon S3 `ListObjects`操作 651
範例：使用全域組態執行個體化用戶端 653
安全 655
資料保護 655
身分和存取權管理 656
目標對象 656
使用身分驗證 657
使用政策管理存取權 659
AWS 服務 如何使用 IAM 661
對 AWS 身分和存取進行故障診斷 661
合規驗證 663
恢復能力 664
基礎設施安全性 664
Amazon S3 加密用戶端遷移 665
遷移概觀 665
更新現有用戶端以讀取新格式 665
將加密和解密用戶端遷移至 V2 666
遷移範例 667
常見問答集 670
在用戶端上可使用哪些方法？ 670
我該如何處理 cURL SSL 憑證的錯誤？ 670
用戶端可使用哪些 API 版本？ 670
用戶端可使用哪些區域版本？ 670
為什麼我無法上傳或下載大於 2 GB 的檔案？ 671
我要如何查看透過線路傳輸的資料內容？ 671
我要如何在請求中設定任意標頭？ 671

我要如何簽署任意請求？ 672
我要如何在傳送命令之前先加以修改？ 672
什麼是 CredentialsException？ 672
是否 適用於 PHP 的 AWS SDK 適用於 HHVM？ 672
我要如何停用 SSL？ 673
我該如何處理「剖析錯誤」？ 673
為什麼 Amazon S3 用戶端解壓縮 gziped 檔案？ 673
如何在 Amazon S3 中停用內文簽署？ 674
在 適用於 PHP 的 AWS SDK中如何處理重試機制？ 674
我要如何處理具有錯誤碼的例外狀況？ 675
詞彙表 676
文件歷史紀錄 679
..... dclxxxiii

什麼是第 3 適用於 PHP 的 AWS SDK 版？

第 3 適用於 PHP 的 AWS SDK 版可讓 PHP 開發人員在其 PHP 程式碼中使用 Amazon Web Services，並使用 Amazon S3、Amazon DynamoDB 和 S3 Glacier 等服務建置強大的應用程式和軟體。您可以透過 Composer 安裝 SDK，方法是要求 `aws/aws-sdk-php`套件，或是下載獨立 `aws.zip` 或 `aws.phar` 檔案，即可在幾分鐘內開始使用。

並不是所有服務皆可立即於開發套件中使用。若要了解 目前支援哪些服務 適用於 PHP 的 AWS SDK，請參閱服務名稱和 API 版本。

> **Note**
>
> 如果您要將程式碼從使用 SDK 第 2 版遷移到第 3 版，請務必從 第 2 版讀取升級 適用於 PHP 的 AWS SDK。

開始使用 SDK

如果您已準備好開始使用 SDK，請遵循 第 3 適用於 PHP 的 AWS SDK 版入門章節。它會引導您使用進行身分驗證 AWS、設定開發環境，以及使用 Amazon S3 建立第一個基本應用程式。

其他資源

- 常見問答集
- 詞彙表
- AWS SDKs和工具參考指南：包含設定、功能和其他在 AWS SDKs之間常見的基本概念。
- Guzzle 文件
- 使用 的程式碼範例 適用於 PHP 的 AWS SDK 可在 awsdocs/aws-doc-sdk-examples 儲存庫中找到。
- Gitter 上的 PHP SDK 社群。
- AWS re:Post.

GitHub:

- 的原始程式碼 適用於 PHP 的 AWS SDK 可在 aws/aws-sdk-php 儲存庫中取得。

- 幫助開發套件
- 回報錯誤或是請求功能

API 文件

在 https：//https://docs.aws.amazon.com/sdk-for-php/latest/reference/ 尋找開發套件的 API 文件。

開發套件主要版本的維護與支援

如需開發套件主要版本及其基礎相依性之維護與支援的相關資訊，請參閱《AWS 開發套件及工具參考指南》中的以下內容：

- AWS SDKs和工具維護政策
- AWS SDKs和工具版本支援矩陣

第 3 適用於 PHP 的 AWS SDK 版入門

本章旨在協助您開始使用 第 3 適用於 PHP 的 AWS SDK 版並執行 。

主題

- 第 3 適用於 PHP 的 AWS SDK 版的要求和建議
- 安裝第 3 適用於 PHP 的 AWS SDK 版
- AWS 使用第 3 適用於 PHP 的 AWS SDK 版向 驗證
- 使用第 3 適用於 PHP 的 AWS SDK 版建立簡單的應用程式
- AWS Cloud9 搭配第 3 適用於 PHP 的 AWS SDK 版使用

第 3 適用於 PHP 的 AWS SDK 版的要求和建議

為了獲得最佳結果 適用於 PHP 的 AWS SDK，請確保您的環境支援下列要求和建議。

要求

若要使用 適用於 PHP 的 AWS SDK，您必須使用 PHP 8.1 版或更新版本，並啟用 SimpleXML PHP 延伸。如果您需要簽署私有 Amazon CloudFront URLs，您也需要 OpenSSL PHP 延伸。

建議

除了最低要求外，我們建議您也需安裝、解除安裝並使用以下操作。

安裝 cURL 7.16.2 或更新版本	使用以 OpenSSL/NSS 和 zlib 編譯的最新版本 cURL。如果 cURL 未安裝在您的系統上，且您未幫用戶端設定自訂 http_handler，軟體開發套件將使用 PHP 串流包裝函數。
使用 OPCache	在共用記憶體中儲存預先編譯的指令碼位元碼來使用 OPcache 延伸改善 PHP 效能。這會消除 PHP 載入並解析每個請求之指令碼的需求。此延伸通常預設為啟用。

	執行 Amazon Linux 時，您需要安裝 php56-opcache 或 php55-opcache yum 套件才可使用 OPCache 延伸。
在生產環境中解除安裝 Xdebug	Xdebug 可協助判斷效能瓶頸。不過，如果效能對您的應用程式非常重要，請勿在生產環境中安裝 Xdebug 延伸模組。載入延伸會大幅降低開發套件的效能。
使用 Composer classmap 自動載入函式	自動載入函式會載入類別，因為 PHP 指令碼需要類別。Composer 會產生可自動載入應用程式 PHP 指令碼以及應用程式所需之所有其他 PHP 指令碼的自動載入函式，包括 適用於 PHP 的 AWS SDK。 對於生產環境，我們建議您使用 classmap 自動載入函式來提升自動載入函式效能。您可以傳遞 `-o` 或 `==optimize-autoloader` 選項到 Composer 的安裝命令來產生 classmap 自動載入函數。

相容性測試

執行位於 SDK 程式碼庫中的 `compatibility-test.php` 檔案，以確認您的系統可以執行 SDK。除符合軟體開發套件最低系統需求外，相容性測試將檢查選用設定，並提出可協助提升效能的建議。相容性測試輸出結果至命令列或 Web 瀏覽器。在瀏覽器中查看測試結果時，成功檢查時會出現綠色、警告為紫色、失敗為紅色。從命令列執行時，檢查的結果會出現在不同列。

使用開發套件回報問題時，分享相容性測試輸出可協助判斷潛在原因。

安裝第 3 適用於 PHP 的 AWS SDK 版

您可以安裝第 3 適用於 PHP 的 AWS SDK 版：

- 透過 Composer 做為相依項目
- 軟體開發套件預先封裝的 phar
- 軟體開發套件的 ZIP 檔案

安裝第 3 適用於 PHP 的 AWS SDK 版之前，請確定您的環境使用 PHP 8.1 版或更新版本。進一步了解環境需求和建議。

> **Note**
>
> 透過 .phar 和 .zip 方法安裝 SDK 需要分別安裝和啟用多位元組字串 PHP 延伸。

透過 Composer 安裝 適用於 PHP 的 AWS SDK 做為相依性

Composer 是安裝 的建議方法 適用於 PHP 的 AWS SDK。Composer 是適用於 PHP 的工具，可用來管理和安裝您專案的相依項目。

如需如何安裝 Composer 和設定自動載入，以及遵照其他最佳實務定義相依項目的詳細資訊，請參閱 getcomposer.org。

安裝 Composer

如果 Composer 尚未在您的專案中，請在 Download Composer 頁面上下載並安裝 Composer。

- 對於 Windows，請遵循 Windows Installer 指示。
- 對於 Linux，請遵循命令列安裝指示。

透過 Composer 新增 適用於 PHP 的 AWS SDK 做為相依性

如果您的系統已全域安裝 Composer，請在專案的基本目錄中執行下列命令，以安裝 適用於 PHP 的 AWS SDK 做為相依性：

```
$ composer require aws/aws-sdk-php
```

否則，請輸入此 Composer 命令來安裝最新版本的 適用於 PHP 的 AWS SDK 做為相依性。

```
$ php -d memory_limit=-1 composer.phar require aws/aws-sdk-php
```

將自動載入函式加入至 php 指令碼

安裝 Composer 會在您的環境中建立多個資料夾和檔案。您會使用的主要檔案是 `autoload.php`，它位在環境的 `vendor` 資料夾中。

若要在指令碼 適用於 PHP 的 AWS SDK 中利用 ，請在指令碼中包含自動載入器，如下所示。

```
<?php
   require '/path/to/vendor/autoload.php';
?>
```

使用封裝的 phar 安裝

的每個版本都 適用於 PHP 的 AWS SDK 包含預先封裝的 phar (PHP 封存），其中包含執行 SDK 所需的所有類別和相依性。此外， phar 會自動為 適用於 PHP 的 AWS SDK 及其所有相依性註冊類別自動載入器。

您可以下載封裝的 phar，並在指令碼中加入此檔案。

```
<?php
   require '/path/to/aws.phar';
?>
```

> **Note**
>
> 不建議使用 PHP 搭配 Suhosin 修補程式，但是在 Ubuntu 和 Debian 版本中，這是常見的做法。在這種情況下，您可能需要在 suhosin.ini 中允許使用 phar。若沒有執行此作業，則在您的程式碼中包含 phar 檔案，將會造成無提示的錯誤。若要修改 suhosin.ini，請加入下列程式行。
>
> ```
> suhosin.executor.include.whitelist = phar
> ```

使用 ZIP 檔案安裝

適用於 PHP 的 AWS SDK 包含 ZIP 檔案，其中包含執行 SDK 所需的所有類別和相依性。此外，該 ZIP 檔案包含了 適用於 PHP 的 AWS SDK 的類別自動載入函式，以及此函式的相依項目。

若要安裝軟體開發套件，請下載 .zip 檔案，然後將其解壓縮到您的專案中由您自選的位置。然後，在您的指令碼中加入自動載入函式，如下所示。

```
<?php
   require '/path/to/aws-autoloader.php';
```

```
?>
```

AWS 使用第 3 適用於 PHP 的 AWS SDK 版向 驗證

使用 進行開發 AWS 時，您必須建立程式碼向 進行身分驗證的方式 AWS 服務。您可以根據環境和您可用的存取權，以不同的方式設定 AWS 資源的程式設計 AWS 存取。

若要選擇您的身分驗證方法並針對 SDK 進行設定，請參閱 AWS SDKs和工具參考指南中的身分驗證和存取。

使用 IAM Identity Center 設定身分驗證以進行本機開發

我們建議在本機開發且未獲得其雇主身分驗證方法的新使用者進行設定 AWS IAM Identity Center。此方法包括安裝 AWS CLI 以簡化組態，以及定期登入 AWS 存取入口網站。如果您選擇此方法，您的環境應該在您完成 AWS SDKs和工具參考指南中的 IAM Identity Center 身分驗證程序之後包含下列元素：

- 在執行應用程式之前 AWS CLI，您用來啟動 AWS 存取入口網站工作階段的 。
- 共用 AWSconfig檔案，其[default]設定檔具有一組組態值，可由 SDK 參考。若要尋找此檔案的位置，請參閱 AWS SDK 和工具參考指南中的共用檔案位置。
- 共用config檔案包含 region設定。這會設定軟體開發套件用於請求 AWS 區域 的預設值。此區域用於未明確設定 region 屬性的 SDK 服務請求。
- 開發套件使用描述檔的 SSO 字符提供者組態，在傳送請求至 之前取得登入資料 AWS。此sso_role_name值是連接至 IAM Identity Center 許可集的 IAM 角色，允許存取您應用程式中 AWS 服務 使用的 。

  下列範例config檔案顯示使用 SSO 字符提供者組態設定的預設設定檔。設定檔sso_session的設定是指具名sso-session區段。sso-session 區段包含啟動 AWS 存取入口網站工作階段的設定。

  ```
  [default]
  sso_session = my-sso
  sso_account_id = 111122223333
  sso_role_name = SampleRole
  region = us-east-1
  output = json

  [sso-session my-sso]
  sso_region = us-east-1
  ```

```
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

適用於 PHP 的 AWS SDK 不需要將其他套件（例如 SSO和 SSOOIDC) 新增至您的應用程式，即可使用 IAM Identity Center 身分驗證。

啟動 AWS 存取入口網站工作階段

在執行存取的應用程式之前 AWS 服務，您需要 SDK 的作用中 AWS 存取入口網站工作階段，才能使用 IAM Identity Center 身分驗證來解析登入資料。視您設定的工作階段長度而定，您的存取最終會過期，且 SDK 將遇到身分驗證錯誤。若要登入 AWS 存取入口網站，請在 中執行下列命令 AWS CLI。

```
aws sso login
```

如果您遵循指引並設定預設設定檔，則不需要使用 `--profile`選項呼叫 命令。如果您的 SSO 權杖提供者組態使用已命名的設定檔，則命令為 `aws sso login --profile named-profile`。

若要選擇性地測試您是否已經有作用中的工作階段，請執行下列 AWS CLI 命令。

```
aws sts get-caller-identity
```

如果您的工作階段處於作用中狀態，對此命令的回應會報告共用`config`檔案中設定的 IAM Identity Center 帳戶和許可集。

> **Note**
>
> 如果您已有作用中的 AWS 存取入口網站工作階段並執行 `aws sso login`，則不需要提供登入資料。
>
> 登入程序可能會提示您允許 AWS CLI 存取您的資料。由於 AWS CLI 建置在適用於 Python 的 SDK 之上，因此許可訊息可能包含`botocore`名稱的變化。

進一步了解身分驗證

- 如需使用 IAM Identity Center 進行身分驗證的詳細資訊，請參閱 AWS SDKs和工具參考指南中的了解 IAM Identity Center 身分驗證
- 如需了解有關最佳實務的資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

- 若要建立短期 AWS 登入資料，請參閱《IAM 使用者指南》中的暫時安全登入資料。
- 若要了解 適用於 PHP 的 AWS SDK 可使用的其他登入資料提供者，請參閱 AWS SDKs和工具參考指南中的標準化登入資料提供者。

使用第 3 適用於 PHP 的 AWS SDK 版建立簡單的應用程式

使用 向 Amazon S3 打招呼 適用於 PHP 的 AWS SDK。下列範例顯示 Amazon S3 儲存貯體的清單。

先決條件

- 下載並安裝 SDK
- 使用 之前 適用於 PHP 的 AWS SDK，您必須使用 設定身分驗證 AWS。如需設定身分驗證的資訊，請參閱 AWS 使用第 3 適用於 PHP 的 AWS SDK 版向 驗證

在您的程式碼中包含 SDK

無論您使用何種技術來安裝開發套件，您只需使用單一 `require` 陳述式在程式碼中加入開發套件。請參閱下表中來找到最適用於您的安裝技術之 PHP 程式碼。以系統上的實際路徑來替換任何 `/path/to/` 的執行個體。

安裝技術	需要陳述式
使用 Composer	`require '/path/to/vendor/autoload.php';`
使用 phar	`require '/path/to/aws.phar';`
使用 ZIP	`require '/path/to/aws-autoloader.php';`

在本主題中，我們採用 Composer 安裝方法。若您正在使用不同的安裝方法，您可以重新參考本節來尋找要使用的正確 `require` 程式碼。

撰寫程式碼

確定您可以進行身分驗證。

將下列程式碼複製並貼到新的來源檔案中。儲存並命名檔案 `hello-s3.php`。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

/**
 * List your Amazon S3 buckets.
 */

//Create a S3Client
// snippet-start:[s3.php.list_buckets.main]
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Listing all S3 Bucket
$buckets = $s3Client->listBuckets();
foreach ($buckets['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}
```

執行程式

開啟命令提示以執行 PHP 程式。執行 PHP 程式的典型命令語法為：

```
php [source filename] [arguments...]
```

此範例程式碼不使用引數。若要執行此程式碼，請在命令提示中輸入以下內容：

```
$ php hello-s3.php
```

後續步驟

若要測試許多其他 Amazon S3 操作，請查看 GitHub 上的AWS 程式碼範例儲存庫。

AWS Cloud9 搭配第 3 適用於 PHP 的 AWS SDK 版使用

> **Note**
>
> AWS Cloud9 不再提供給新客戶。的現有客戶 AWS Cloud9 可以繼續正常使用服務。進一步了解。

AWS Cloud9 是以 Web 為基礎的整合開發環境 (IDE)，其中包含您用來在雲端中編寫、建置、執行、測試、偵錯和發行軟體的工具集合。您可以使用 AWS Cloud9 搭配 ，使用 瀏覽器 適用於 PHP 的 AWS SDK 撰寫和執行 PHP 程式碼。 AWS Cloud9 包含程式碼編輯器和終端機等工具。由於 AWS Cloud9 IDE 是以雲端為基礎，因此您可以使用網際網路連線的機器，從辦公室、家中或任何地方處理專案。如需 的一般資訊 AWS Cloud9，請參閱AWS Cloud9 《 使用者指南》。

請依照下列指示 AWS Cloud9 ，使用 設定 適用於 PHP 的 AWS SDK：

- 步驟 1：設定您的 AWS 帳戶 以使用 AWS Cloud9
- 步驟 2：設定您的 AWS Cloud9 開發環境
- 步驟 3：設定 適用於 PHP 的 AWS SDK
- 步驟 4：下載範例程式碼
- 步驟 5：執行範例程式碼

步驟 1：設定您的 AWS 帳戶 以使用 AWS Cloud9

若要使用 AWS Cloud9，請從 登入 AWS Cloud9 主控台 AWS Management Console。

> **Note**
>
> 如果您使用 AWS IAM Identity Center 進行身分驗證，您可能需要`iam:ListInstanceProfilesForRole`將必要的 許可新增至 IAM 主控台中的使用者連接政策。

若要在 AWS 帳戶中設定 IAM 實體以存取 AWS Cloud9 和登入 AWS Cloud9 主控台，請參閱AWS Cloud9 《 使用者指南》中的 的 團隊設定 AWS Cloud9。

步驟 2：設定您的 AWS Cloud9 開發環境

登入 AWS Cloud9 主控台之後，請使用 主控台來建立 AWS Cloud9 開發環境。建立環境後， AWS Cloud9 開啟該環境的 IDE。

如需詳細資訊，請參閱AWS Cloud9 《 使用者指南》中的在 中建立環境 AWS Cloud9。

> **Note**
>
> 第一次由主控台建立您的環境時，建議您選擇 Create a new instance for environment (EC2) (為環境建立新的執行個體) 選項。此選項 AWS Cloud9 會告知 建立環境、啟動 Amazon EC2 執行個體，然後將新執行個體連線至新環境。這是開始使用 的最快方法 AWS Cloud9。

如果 IDE 中尚未開啟終端機，請開啟終端機。從 IDE 的選單列，選擇 Window, New Terminal (視窗、新增終端機)。您可以使用終端機視窗來安裝工具和建置您的應用程式。

步驟 3：設定 適用於 PHP 的 AWS SDK

AWS Cloud9 開啟開發環境的 IDE 後，請使用終端機視窗在您的 適用於 PHP 的 AWS SDK 環境中設定 。

Composer 是安裝 的建議方法 適用於 PHP 的 AWS SDK。Composer 是適用於 PHP 的工具，可用來管理和安裝您專案的相依項目。

如需如何安裝 Composer 和設定自動載入，以及遵照其他最佳實務定義相依項目的詳細資訊，請參閱 getcomposer.org。

安裝 Composer

如果 Composer 尚未在您的專案中，請在 Download Composer 頁面上下載並安裝 Composer。

- 對於 Windows，請遵循 Windows Installer 指示。
- 對於 Linux，請遵循命令列安裝說明。

透過 Composer 新增 適用於 PHP 的 AWS SDK 做為相依性

如果您的系統已全域安裝 Composer，請在專案的基本目錄中執行下列命令，以安裝 適用於 PHP 的 AWS SDK 做為相依性：

```
$ composer require aws/aws-sdk-php
```

否則，請輸入此 Composer 命令來安裝最新版本的 適用於 PHP 的 AWS SDK 做為相依性。

```
$ php -d memory_limit=-1 composer.phar require aws/aws-sdk-php
```

將自動載入函式加入至 php 指令碼

安裝 Composer 會在您的環境中建立多個資料夾和檔案。您會使用的主要檔案是 `autoload.php`，它位在環境的 `vendor` 資料夾中。

若要在指令碼 適用於 PHP 的 AWS SDK 中利用 ，請在指令碼中包含自動載入器，如下所示。

```
<?php
   require '/path/to/vendor/autoload.php';
?>
```

步驟 4：下載範例程式碼

使用終端機視窗將 的範例程式碼下載 適用於 PHP 的 AWS SDK 到 AWS Cloud9 開發環境中。

若要將官方 AWS SDK 文件中使用的所有程式碼範例下載到您環境的根目錄，請執行下列命令：

```
$ git clone https://github.com/awsdocs/aws-doc-sdk-examples.git
```

的程式碼範例 適用於 PHP 的 AWS SDK 位於 `ENVIRONMENT_NAME/aws-doc-sdk-examples/php`目錄，其中 `ENVIRONMENT_NAME`是您開發環境的名稱。

若要遵循使用 Amazon S3 範例，建議您從程式碼範例 開始`ENVIRONMENT_NAME/aws-doc-sdk-examples/php/example_code/s3/ListBuckets.php`。此範例會列出您的 Amazon S3 儲存貯體。使用終端機視窗導覽至 `s3` 目錄並列出檔案。

```
$ cd aws-doc-sdk-examples/php/example_code/s3
$ ls
```

若要在 中開啟檔案 AWS Cloud9，您可以`ListBuckets.php`直接在終端機視窗中按一下 。

如需進一步了解程式碼範例的支援，請參閱適用於 PHP 的 AWS SDK 程式碼範例。

步驟 5：執行範例程式碼

若要在 AWS Cloud9 開發環境中執行程式碼，請選擇頂端選單列中的執行按鈕。 AWS Cloud9 會自動偵測`.`php副檔名，並使用 PHP （內建 Web 伺服器） 執行器執行程式碼。不過，在此範例中，我們實際上想要 PHP (**`cli`**) 選項。如需在 中執行程式碼的詳細資訊 AWS Cloud9，請參閱AWS Cloud9 《使用者指南》中的執行程式碼。

在以下螢幕擷取畫面中，請注意這些基本區域：

- 1：執行。執行按鈕位於頂端選單列。這會開啟結果的新標籤。

> **Note**
>
> 您也可以手動建立新的執行組態。在選單列上，選擇 Run (執行)、Run Configurations (執行組態)、New Run Configuration (新增執行組態)。

- 2： Command. 會將路徑和檔案名稱 AWS Cloud9 填入您執行的檔案的命令文字方塊。如果您的程式碼預期任何命令列參數都會傳入，則可以像透過終端機視窗執行程式碼一樣，將這些參數新增至命令列。
- 3：Runner. AWS Cloud9 偵測您的副檔名是 ，`.`php並選取 PHP （內建 Web 伺服器） Runner 來執行您的程式碼。選取 PHP (**`cli`**) 以改為執行此範例。

從執行中的程式碼產生的任何輸出會顯示在 索引標籤中。

在第 3 適用於 PHP 的 AWS SDK 版中設定服務用戶端

若要以程式設計方式存取 AWS 服務，第 3 適用於 PHP 的 AWS SDK 版會為每個 使用用戶端物件 AWS 服務。例如，如果您的應用程式需要存取 Amazon EC2，您可以建立 Amazon EC2 用戶端物件，這是 Ec2Client 類別的執行個體，以與該服務界面。然後，您可以使用 服務用戶端向該用戶端提出請求 AWS 服務。

設定 SDK 行為的方法有很多，但最終一切都與服務用戶端的行為有關。在程式碼建立使用該組態的服務用戶端之前，任何組態都不會生效。

您提供的組態範例如下：

- 當您呼叫 服務 AWS 時，程式碼如何向 進行身分驗證
- AWS 區域 您希望服務用戶端使用的
- 服務呼叫的重試和逾時設定
- HTTP 代理組態

如需許多 AWS SDKs 通用的設定、功能和其他基本概念，請參閱 AWS SDKs 和工具參考指南。

在外部設定第 3 適用於 PHP 的 AWS SDK 版的服務用戶端

許多組態設定可以在程式碼之外處理。大多數組態設定可以設定為環境變數或單獨的共用 AWS `config`檔案中。 AWS 共用`config`檔案可以維護個別的設定集，稱為設定檔，為不同的環境或測試提供不同的組態。如需共用 AWS `config`和 `credentials` 檔案的完整討論，請參閱 AWS SDKs和工具參考指南。

大多數環境變數和共用`config`檔案設定會在 AWS SDKs 和工具之間標準化和共用，以支援不同程式設計語言和應用程式的一致功能。

若要查看開發套件可從環境變數或組態檔案解析的所有設定，請參閱開發套件和工具參考指南中的設定參考。 AWS SDKs

用戶端組態的組態提供者鏈結

軟體開發套件會檢查多個位置（或來源）來尋找組態值。

1. 在程式碼中或服務用戶端本身上設定的任何明確設定，都優先於任何其他設定。

2. 環境變數
   - 如需設定環境變數的詳細資訊，請參閱 AWS SDKs和工具參考指南中的環境變數。
   - 請注意，您可以為殼層設定不同範圍的環境變數：全系統、全使用者和特定終端機工作階段。
3. 共用 `config`和 `credentials` 檔案
   - 如需設定這些檔案的詳細資訊，請參閱 SDK `config`和工具參考指南中的共用 和 `credentials` 檔案。 AWS SDKs
4. 開發套件原始碼本身提供的任何預設值都會最後使用。
   - 有些屬性，例如區域，沒有預設值。您必須在程式碼、環境設定或共用`config`檔案中明確指定它們。如果 SDK 無法解析所需的組態，API 請求可能會在執行時間失敗。

除了此一般組態鏈之外，第 3 適用於 PHP 的 AWS SDK 版也使用專門的提供者鏈，包括登入資料提供者鏈和AWS 區域 解析鏈。這些特殊化鏈結新增了額外的供應商，這些供應商會將 SDK 正在執行的環境納入考量。例如，在容器或 EC2 執行個體中。

建立使用外部設定設定的 服務用戶端

您需要在應用程式中建立服務用戶端，才能與 通訊 AWS 服務。服務用戶端是您與 的基本連線 AWS 服務，可處理所有複雜的通訊詳細資訊，讓您不必擔心。它們會自動處理安全、錯誤處理和重試等重要任務，讓您專注於建置應用程式，而不是處理技術複雜性。

使用沒有參數的建構函數來建構服務用戶端

如果您需要的所有組態設定都來自外部來源，您可以使用空的建構函數來建立服務用戶端：

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;

$s3 = new S3Client([]);
```

先前的程式碼片段會建立`S3Client`執行個體。在建立期間，軟體開發套件會查看組態提供者鏈中的設定。一旦軟體開發套件找到設定值，就會使用該值。

預設 AWS 區域 提供者鏈和預設登入資料提供者鏈也會用於建立程序。在鏈結的某個位置，軟體開發套件必須解析 AWS 區域 以使用和尋找設定，讓它能夠擷取簽署請求的登入資料。如果 SDK 找不到這些值，則用戶端建立會失敗。

適用於 PHP 的 AWS SDK 第 3 版環境變數

除了 AWS SDKs支援的跨磁碟設定之外，第 3 適用於 PHP 的 AWS SDK 版適用於下列環境變數：

AWS_SDK_LOAD_NONDEFAULT_CONFIG

設定時，此環境變數會指示 SDK 從 AWS `config`檔案 (`~/.aws/config`) 載入登入資料，以及`credentials`檔案 (`~/.aws/credential`)。

AWS_SDK_UA_APP_ID

設定自訂應用程式識別符，該識別符包含在 SDK 所提出請求的 User-Agent 標頭中。

AWS_SUPPRESS_PHP_DEPRECATION_WARNING

設為 true 時， 會隱藏軟體開發套件可能產生的 PHP 棄用警告。

在 第 3 適用於 PHP 的 AWS SDK 版的程式碼中設定服務用戶端

除了在外部設定服務用戶端之外，您也可以以程式設計方式在程式碼中設定它們。

透過在程式碼中設定服務用戶端，您可以精細控制許多可用的選項。您也可以在外部設定的大多數組態，以程式碼設定。

程式碼中的基本組態

您可以透過將選項的關聯陣列傳遞給用戶端的建構函式，在程式碼中建立和設定服務用戶端。在下列範例中，關聯陣列只包含用戶端使用的「區域」選項：

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;

//Create an S3Client
$s3 = new S3Client([
    'region' => 'eu-south-2'
]);
```

您可以在組態選項主題中找到選用的「版本」參數的相關資訊。

請注意，我們並未明確提供登入資料給用戶端。這是因為 SDK 使用預設登入資料提供者鏈結來尋找登入資料資訊。

中會詳細說明所有一般用戶端組態選項第 3 適用於 PHP 的 AWS SDK 版的用戶端建構函式選項。提供給用戶端的選項陣列可能會因您正在建立的用戶端而不同。這些自訂用戶端組態選項的說明如各用戶端的 API 文件所述。

使用 Sdk類別

Aws\Sdk 類別做為用戶端 factory，用於管理跨多個用戶端的共用組態選項。可以提供給特定用戶端建構函數的許多選項也可以提供給 Aws\Sdk類別。這些選項接著會套用到每個用戶端建構函式。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

範例程式碼

```
// The same options that can be provided to a specific client constructor can also be
 supplied to the Aws\Sdk class.
// Use the us-west-2 region and latest version of each client.
$sharedConfig = [
    'region' => 'us-west-2'
];
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk($sharedConfig);
// Create an Amazon S3 client using the shared configuration data.
$client = $sdk->createS3();
```

在所有用戶端共用的選項將放入根層級的金鑰值組。服務特定的組態資料可以在與服務命名空間（例如，「S3」、「DynamoDb」等）相同的索引鍵的關聯陣列中提供。

```
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2',
    'DynamoDb' => [
        'region' => 'eu-central-1'
    ]
]);

// Creating an Amazon DynamoDb client will use the "eu-central-1" AWS Region.
```

```
$client = $sdk->createDynamoDb();
```

服務特定的組態值為服務特定值與根層級值的整合 (即服務特定值以淺層合併至根層級值)。

> **Note**
>
> 如果您正在您的應用程式中使用多個用戶端執行個體，我們強烈建議您使用 Sdk 類別。Sdk 類別自動為各開發套件用戶端使用相同的 HTTP 用戶端，讓不同服務的開發套件用戶端執行非鎖定 HTTP 請求。如果軟體開發套件用戶端不使用相同的 HTTP 用戶端，則由軟體開發套件用戶端傳送的 HTTP 請求可能會封鎖服務之間的 promise 協同。

第 3 適用於 PHP 的 AWS SDK 版的用戶端建構函式選項

用戶端建構函數選項可在用戶端建構函數中提供，或提供給 `Aws\Sdk`類別。提供給特定類型用戶端的選項陣列可能不同，取決於您所建立的用戶端。這些自訂用戶端組態選項的說明如各用戶端的 API 文件所述。

如果您未明確提供用戶端所需的用戶端建構函數選項，適用於 PHP 的 SDK 會從環境變數或 AWS 組態檔案尋找值。所有用戶端都需要登入資料提供者值和 AWS 區域 值，因此您必須將這些值設定為建構函數選項或外部。

依預設，正在檢查的組態檔案會位於您主目錄中的 `.aws/config`，通常為 `~/.aws/config`。但是，您可以使用環境變數 `AWS_CONFIG_FILE` 來設訂預設組態檔案的位置。例如，如果您使用 限制檔案存取特定目錄，這可能會很有用`open_basedir`。

如需共用 AWS 和`credentials`檔案的位置`config`和格式的詳細資訊，請參閱 AWS SDKs和工具參考指南中的組態。

如需您可以在組態檔案或環境變數中 AWS 設定的所有全域組態設定的詳細資訊，請參閱 SDK 和工具參考指南中的組態和身分驗證設定參考。 AWS SDKs

組態選項

- api_provider
- 登入資料
- 偵錯
- 統計資料
- 端點

- endpoint_provider
- endpoint_discovery
- 處理常式
- http
- http_handler
- profile
- region
- retries
- 結構描述
- 服務
- signature_provider
- signature_version
- ua_append
- use_aws_shared_config_files
- validate
- version

下列範例示範如何將選項傳遞至 Amazon S3 用戶端建構函式。

```
use Aws\S3\S3Client;

$options = [
    'region'            => 'us-west-2',
    'version'           => '2006-03-01',
    'signature_version' => 'v4'
];

$s3Client = new S3Client($options);
```

如需建構用戶端的詳細資訊the section called “建立基本服務用戶端”，請參閱。

api_provider

Type

```
callable
```

PHP 可呼叫接受類型、服務和版本參數，而且會傳回一系列對應的組態資料。類型值可以是 `api`、`waiter` 或 `paginator` 其中之一。

在預設情況下，開發套件使用 `Aws\Api\FileSystemApiProvider` 執行個體，它會從開發套件的 `src/data` 資料夾載入 API 檔案。

登入資料

Type

> `array|Aws\CacheInterface|Aws\Credentials\CredentialsInterface|bool|callable`

傳遞 `Aws\Credentials\CredentialsInterface` 物件以使用特定的登入資料執行個體。以下指定應使用 IAM Identity Center 憑證提供者。此提供者也稱為 SSO 登入資料提供者。

```
$credentials = Aws\Credentials\CredentialProvider::sso('profile default');

$s3 = new Aws\S3\S3Client([
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

如果您使用具名設定檔，請在上一個範例中將設定檔的名稱替換為「`default`」。若要進一步了解如何設定具名設定檔，請參閱 SDK [config和工具參考指南中的共用和credentials檔案](#)。 AWS SDKs

如果您未指定要使用的登入資料提供者，並依賴登入資料提供者鏈結，則驗證失敗所產生的錯誤訊息通常是通用的。它從正在檢查有效登入資料的來源清單中的最後一個提供者產生，這可能不是您嘗試使用的提供者。當您指定要使用的登入資料提供者時，任何產生的錯誤訊息都會更實用且更相關，因為它只由該提供者產生。若要進一步了解檢查登入資料的來源鏈，請參閱 AWS SDKs和工具參考指南中的[登入資料提供者鏈](#)。

傳遞 `false` 以使用 null 登入資料，而且不簽署請求。

```
$s3 = new Aws\S3\S3Client([
    'region'      => 'us-west-2',
    'credentials' => false
]);
```

傳遞可呼叫的登入資料供應商函數以使用函數建立登入資料。

```
use Aws\Credentials\CredentialProvider;

// Only load credentials from environment variables
$provider = CredentialProvider::env();

$s3 = new Aws\S3\S3Client([
    'region'      => 'us-west-2',
    'credentials' => $provider
]);
```

傳遞 `Aws\CacheInterface` 執行個體以快取預設供應商鏈跨多個程序傳回的值。

```
use Aws\Credentials\CredentialProvider;
use Aws\PsrCacheAdapter;
use Symfony\Component\Cache\Adapter\FilesystemAdapter;


$cache = new PsrCacheAdapter(new FilesystemAdapter);
$provider = CredentialProvider::defaultProvider();
$cachedProvider = CredentialProvider::cache($provider, $cache);

$s3 = new Aws\S3\S3Client([
    'region' => 'us-west-2',
    'credentials' => $cachedProvider
]);
```

您可以在第 3 適用於 PHP 的 AWS SDK 版的登入資料指南中找到有關提供登入資料給用戶端的詳細資訊。

> **Note**
>
> 登入資料會在被使用時延遲載入和驗證。

偵錯

Type

`bool|array`

輸出各個傳輸的偵錯資訊。偵錯資訊包含有關交易在準備及透過網路傳送時每個狀態變更的資訊。偵錯輸出也包含有關用戶端使用的特定 HTTP 處理常式的資訊 (例如，偵錯 cURL 輸出)。

設定為 `true` 可在傳送請求時顯示偵錯資訊。

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'debug'   => true
]);

// Perform an operation to see the debug output
$s3->listBuckets();
```

或者，您可以使用以下金鑰提供關聯陣列。

logfn (可呼叫)

以日誌訊息叫用的函數。根據預設，會使用 PHP 的 `echo` 函數。

stream_size (int)

當串流的大小大於此數字時，將不會記錄串流資料。設定為 `0` 將不記錄任何串流資料。

scrub_auth (bool)

設定為 `false`以停用從記錄的訊息中清除身分驗證資料 （表示您的 AWS 存取金鑰 ID 和簽章將傳遞至 `logfn`)。

http (bool)

設為 `false` 以停用較低層級 HTTP 處理器的「偵錯」功能 (例如，verbose cURL 輸出)。

auth_headers (陣列)

設定為您要替換的標頭金鑰值映射，對應到您希望取代它們的值。除非 `scrub_auth` 設定為 `true`，否則將不會使用這些值。

auth_strings (陣列)

設定為規則表達式金鑰值映射以對應到它們的替換。如果 `scrub_auth` 設定為 `true`，這些值將由身分驗證資料清除程式使用。

```
$s3 = new Aws\S3\S3Client([
```

```
    'region'  => 'us-west-2',
    'debug'   => [
        'logfn'        => function ($msg) { echo $msg . "\n"; },
        'stream_size'  => 0,
        'scrub_auth'   => true,
        'http'         => true,
        'auth_headers' => [
            'X-My-Secret-Header' => '[REDACTED]',
        ],
        'auth_strings' => [
            '/SuperSecret=[A-Za-z0-9]{20}/i' => 'SuperSecret=[REDACTED]',
        ],
    ]
]);

// Perform an operation to see the debug output
$s3->listBuckets();
```

> **Note**
>
> 此選項也會輸出`http`偵錯選項所產生的基礎 HTTP 處理常式資訊。偵錯輸出在診斷 適用於 PHP 的 AWS SDK的問題時非常有用。請在開啟開發套件的問題時，提供隔離故障案例的偵錯輸出。

統計資料

Type

`bool|array`

將傳輸統計資料繫結至開發套件操作傳回的錯誤和結果。

設定為 `true` 以收集已傳送之請求的傳輸統計資料。

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'stats'   => true
]);

// Perform an operation
```

```
$result = $s3->listBuckets();
// Inspect the stats
$stats = $result['@metadata']['transferStats'];
```

或者，您可以使用以下金鑰提供關聯陣列。

重試 (bool)

設定為 `false`以停用嘗試重試的報告。在預設情況下，會收集並傳回重試統計資料。

http (bool)

設定為 `true` 以啟用從較低層級的 HTTP 配接器收集統計資料 (例如，GuzzleHttpTransferStats 中傳回的值)。HTTP 處理常式必須支援 __on_transfer_stats 選項才能使其生效。HTTP 統計資料會做為關聯陣列的已編製索引陣列傳回；每個關聯陣列皆包含針對用戶端 HTTP 處理器請求所傳回的傳輸統計資料。預設為停用。

如果重試請求，將會傳回每個請求的傳輸統計資料，其中 `$result['@metadata']['transferStats']['http'][0]` 會包含第一個請求的統計資料，`$result['@metadata']['transferStats']['http'][1]` 則會包含第二個請求的統計資料，以此類推。

計時器 (bool)

設定為 `true` 以啟用命令計時器，它會報告用於操作的總計時鐘時間，以秒為單位。預設為停用。

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'stats'   => [
        'retries'      => true,
        'timer'        => false,
        'http'         => true,
    ]
]);

// Perform an operation
$result = $s3->listBuckets();
// Inspect the HTTP transfer stats
$stats = $result['@metadata']['transferStats']['http'];
// Inspect the number of retries attempted
$stats = $result['@metadata']['transferStats']['retries_attempted'];
// Inspect the total backoff delay inserted between retries
$stats = $result['@metadata']['transferStats']['total_retry_delay'];
```

端點

Type

`string`

Web 服務的完整 URI 名稱。對於使用帳戶特定端點的服務，例如 AWS Elemental MediaConvert ，這是必要的。對於這些服務，請使用 `describeEndpoints`方法請求此端點。

只有在連線至自訂端點（例如 Amazon S3 或 Amazon DynamoDB Local 的本機版本）時，才需要此操作。

以下是連線至 Amazon DynamoDB Local 的範例：

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version'  => '2012-08-10',
    'region'   => 'us-east-1',
    'endpoint' => 'http://localhost:8000'
]);
```

如需可用AWS 區域和端點的清單，請參閱 AWS 區域和端點。

endpoint_provider

Type

`Aws\EndpointV2\EndpointProviderV2|callable`

EndpointProviderV2 或 PHP 可呼叫的選用執行個體，可接受選項雜湊，包括「服務」和「區域」金鑰。其會傳回 NULL 或端點資料的雜湊，其中需要 “endpoint” 鍵。

以下是如何建立最小端點提供者的範例。

```
$provider = function (array $params) {
    if ($params['service'] == 'foo') {
        return ['endpoint' => $params['region'] . '.example.com'];
    }
    // Return null when the provider cannot handle the parameters
    return null;
});
```

endpoint_discovery

Type

```
array|Aws\CacheInterface|Aws\EndpointDiscovery\ConfigurationInterface|
callable
```

端點探索會識別支援端點探索之服務 API 的正確端點並連接至其中。如需支援但不需要端點探索的服務，請在用戶端建立期間啟用 `endpoint_discovery`。如果服務不支援端點探索，此組態會遭到忽略。

`Aws\EndpointDiscovery\ConfigurationInterface`

選用組態供應商，可自動連接到服務 API 的適當端點來進行該服務指定的操作。

`Aws\EndpointDiscovery\Configuration` 物件接受兩個選項，包括布林值 “enabled”，此值表示已啟用端點探索啟用；以及整數 “cache_limit”，此值表示在端點快取中的金鑰數上限。

對於每個建立的用戶端，請傳遞 `Aws\EndpointDiscovery\Configuration` 物件以使用端點探索的特定組態。

```
use Aws\EndpointDiscovery\Configuration;
use Aws\S3\S3Client;

$enabled = true;
$cache_limit = 1000;

$config = new Aws\EndpointDiscovery\Configuration (
    $enabled,
    $cache_limit
);

$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-2',
    'endpoint_discovery' => $config,

]);
```

傳遞 `Aws\CacheInterface` 執行個體以快取端點探索跨多個程序傳回的值。

```
use Aws\DoctrineCacheAdapter;
```

```
use Aws\S3\S3Client;
use Doctrine\Common\Cache\ApcuCache;

$s3 = new S3Client([
    'region'      => 'us-west-2',
    'endpoint_discovery' => new DoctrineCacheAdapter(new ApcuCache),
]);
```

將陣列傳遞至端點探索。

```
use Aws\S3\S3Client;

$s3 = new S3Client([
    'region'      => 'us-west-2',
    'endpoint_discovery' => [
        'enabled' => true,
        'cache_limit' => 1000
    ],
]);
```

處理常式

Type

`callable`

接受命令物件和請求物件的處理常式，而且會傳回承諾 (`GuzzleHttp\Promise\PromiseInterface`)，其中包含 `Aws\ResultInterface` 物件 `Aws\Exception\AwsException`。處理常式不會接受下一個處理常式，因為其為終端機且預期要完成命令。如果沒有提供處理常式，將使用預設的 Guzzle 處理常式。

您可以使用 `Aws\MockHandler` 以傳回模擬結果或擲回模擬例外狀況。您將結果或例外狀況排入佇列，MockHandler 將以 FIFO 順序將它們移出佇列。

```
use Aws\Result;
use Aws\MockHandler;
use Aws\DynamoDb\DynamoDbClient;
use Aws\CommandInterface;
use Psr\Http\Message\RequestInterface;
use Aws\Exception\AwsException;
```

```
$mock = new MockHandler();

// Return a mocked result
$mock->append(new Result(['foo' => 'bar']));

// You can provide a function to invoke; here we throw a mock exception
$mock->append(function (CommandInterface $cmd, RequestInterface $req) {
    return new AwsException('Mock exception', $cmd);
});

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region' => 'us-east-1',
    'handler' => $mock
]);

// Result object response will contain ['foo' => 'bar']
$result = $client->listTables();

// This will throw the exception that was enqueued
$client->listTables();
```

http

Type

`array`

設定為一系列的 HTTP 選項並套用到開發套件建立的 HTTP 請求和傳輸。

開發套件支援以下組態選項：

cert

Type

`string|array`

指定 PEM 格式的用戶端憑證。

- 設定為字串，僅用於憑證檔案的路徑。

```
use Aws\S3\S3Client;

$client = new S3Client([
    'region'  => 'us-west-2',
    'http'    => ['cert' => '/path/to/cert.pem']
]);
```

- 設定為陣列，其中包含路徑和密碼。

```
use Aws\S3\S3Client;

$client = new S3Client([
    'region'  => 'us-west-2',
    'http'    => [
        'cert' => ['/path/to/cert.pem', 'password']
    ]
]);
```

connect_timeout

浮點數，描述在嘗試連接到伺服器時等待的秒數。使用 `0` 以無限期等待 (預設的行為)。

```
use Aws\DynamoDb\DynamoDbClient;

// Timeout after attempting to connect for 5 seconds
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'connect_timeout' => 5
    ]
]);
```

偵錯

Type

`bool|resource`

指示基礎 HTTP 處理常式輸出偵錯資訊。不同 HTTP 處理常式提供的偵錯資訊會有不同。

- 傳遞 `true` 將偵錯輸出寫入至 STDOUT。
- 傳遞 `resource` 所傳回的 `fopen` 以便將偵錯輸出寫入至特定 PHP 串流資源。

decode_content

Type

`bool`

指示基礎 HTTP 處理常式擴大壓縮回應的主體。在未啟用時，可使用 `GuzzleHttp\Psr7\InflateStream` 擴大壓縮回應的主體。

> **Note**
>
> 在軟體開發套件的預設 HTTP 處理器中，根據預設會啟用內容解碼。由於回溯相容性因素，此預設無法變更。如果您在 Amazon S3 中存放壓縮檔案，建議您在 S3 用戶端層級停用內容解碼。

```
use Aws\S3\S3Client;
use GuzzleHttp\Psr7\InflateStream;

$client = new S3Client([
    'region'  => 'us-west-2',
    'http'    => ['decode_content' => false],
]);

$result = $client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'massize_gzipped_file.tgz'
]);

$compressedBody = $result['Body']; // This content is still gzipped
$inflatedBody = new InflateStream($result['Body']); // This is now readable
```

延遲

Type

`int`

在傳送請求之前的延遲毫秒數。這通常用於重試請求之前的延遲。

expect

Type

```
bool|string
```

系統會將此選項傳遞至基礎 HTTP 處理常式。在預設情況下，當請求內文超過 1 MB 時會設定 Expect: 100-Continue 標頭。`true` 或 `false` 會啟用或停用所有請求的標頭。如果使用的是整數，只有內文超過此設定的請求會使用該標頭。使用方式為整數時，如果內文大小不明，則會傳送 Expect 標頭。

> **Warning**
>
> 停用 Expect 標頭可避免該服務傳回身分驗證或其他錯誤。此選項的設定請務必謹慎進行。

進度

Type

```
callable
```

定義傳輸進度進行時叫用的函數。此函數接受以下引數：

1. 預期要下載的位元組總數。
2. 目前已下載的位元組數。
3. 預期要上傳的位元組數。
4. 目前已上傳的位元組數。

```
use Aws\S3\S3Client;

$client = new S3Client([
    'region'  => 'us-west-2'
]);

// Apply the http option to a specific command using the "@http"
// command parameter
```

```
$result = $client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'large.mov',
    '@http' => [
        'progress' => function ($expectedDl, $dl, $expectedUl, $ul) {
            printf(
                "%s of %s downloaded, %s of %s uploaded.\n",
                $expectedDl,
                $dl,
                $expectedUl,
                $ul
            );
        }
    ]
]);
```

proxy

Type

`string|array`

您可以使用 `proxy`選項，透過代理連線到 AWS 服務。

- 提供字串值以連接到所有 URI 類型的代理。代理字串值可以包含結構描述、使用者名稱和密碼。例如 `"http://username:password@192.168.16.1:10"`。
- 提供代理設定的關聯陣列，其中的鍵是 URI 的結構描述，而其值是指定 URI 的代理 (亦即，您可以為 ”http” 和 “https” 端點提供不同的代理)。

```
use Aws\DynamoDb\DynamoDbClient;

// Send requests through a single proxy
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);

// Send requests through a different proxy per scheme
$client = new DynamoDbClient([
```

```
    'region'  => 'us-west-2',
    'http'    => [
        'proxy' => [
            'http' => 'tcp://192.168.16.1:10',
            'https' => 'tcp://192.168.16.1:11',
        ]
    ]
]);
```

您可以使用 HTTP_PROXY 環境變數來設定 “http” 特定通訊協定的代理，以及使用 HTTPS_PROXY 環境變數來設定 “https” 特定的代理。

接收

Type

`resource|string|Psr\Http\Message\StreamInterface`

sink 選項控制將操作的回應資料下載至何處。

- 提供 resource 傳回的 fopen 以便將回應內文下載至 PHP 串流。
- 提供磁碟上的檔案路徑做為 string 值，以便將回應內文下載至磁碟上的特定檔案。
- 提供 Psr\Http\Message\StreamInterface 以便將回應內文下載至特定 PSR 串流物件。

> **Note**
>
> 依據預設，開發套件會將回應內文下載至 PHP 臨時串流。這表示資料會保持在記憶體中，直到內文的大小達到 2 MB，此時資料將寫入磁碟上的臨時檔案。

同步

Type

`bool`

synchronous 選項會通知您要封鎖結果的基礎 HTTP 處理常式。

串流

Type

`bool`

設定為 `true` 以告知您想要 Web 服務串流回應內文的基礎 HTTP 處理常式，而不是一開始就下載它。例如，此選項在 Amazon S3 串流包裝函式類別中依賴，以確保資料串流。

timeout

Type

`float`

浮點數，描述請求的逾時 (以秒為單位)。使用 `0` 以無限期等待 (預設的行為)。

```
use Aws\DynamoDb\DynamoDbClient;

// Timeout after 5 seconds
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'timeout' => 5
    ]
]);
```

驗證

Type

`bool|string`

您可以使用 `verify` `http` 選項自訂開發套件的對等 SSL/TLS 憑證驗證行為。

- 設定為 `true` 以啟用 SSL/TLS 對等憑證驗證，並使用作業系統提供的預設 CA 套件。
- 設定為 `false` 以停用對等憑證驗證。(這是不安全的！)
- 設定為字串以提供 CA 憑證套件的路徑，以使用自訂 CA 套件來啟用驗證。

如果您的系統找不到 CA 套件，您會接收到錯誤，請將 CA 套件的路徑提供給軟體開發套件。如果您不需要特定 CA 套件，Mozilla 會提供常用的 CA 套件，您可以在這裡下載 (這是由 cURL 的維護者所維護)。一旦您的磁碟上有 CA 套件之後，即可設定 `openssl.cafile` PHP .ini 設定以指向該檔案的路徑，如此可讓您省略 `verify` 請求選項。您可以在 cURL 網站找到 SSL 憑證的更多詳細資訊。

```
use Aws\DynamoDb\DynamoDbClient;

// Use a custom CA bundle
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'verify' => '/path/to/my/cert.pem'
    ]
]);

// Disable SSL/TLS verification
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'verify' => false
    ]
]);
```

http_handler

Type

`callable`

`http_handler` 選項用於整合開發套件與其他 HTTP 用戶端。`http_handler` 選項是函數，可接受 `Psr\Http\Message\RequestInterface` 物件和套用至命令的多種 `http` 選項，並傳回 `GuzzleHttp\Promise\PromiseInterface` 物件履行的 `Psr\Http\Message\ResponseInterface` 物件或遭到拒絕並附帶以下例外狀況資料：

- `exception` - (`\Exception`) 遇到的例外狀況。
- `response` - (`Psr\Http\Message\ResponseInterface`) 接收到的回應 (如果有的話)。
- `connection_error` - (bool) 設定為 `true` 將錯誤標記為連線錯誤。如有需要，將此值設定為 `true` 也可讓開發套件自動重試操作。

開發套件藉由將提供的 `http_handler` 與 `handler` 物件包裝在一起，自動將指定的 `http_handler` 轉換為正常的 `Aws\WrappedHttpHandler` 選項。

根據預設，軟體開發套件會使用 Guzzle 做為其 HTTP 處理器。您可以在這裡提供不同的 HTTP 處理器，或者為 Guzzle 客戶端提供您自己的自訂定義選項。

設定 TLS 版本

其中一種使用案例是使用 Curl 設定 Guzzle 使用的 TLS 版本 (假設 Curl 已安裝在您的環境中)。請注意 Curl 的版本限制條件，了解其支援的 TLS 版本。根據預設會使用最新版本。如果明確設定 TLS 版本，而遠端伺服器不支援此版本，則會產生錯誤，而不是使用較早的 TLS 版本。

您可以藉由將 debug 用戶端選項設為 true 並檢查 SSL 連線輸出，來判斷用於指定用戶端操作的 TLS 版本。該行可能看起來如下：`SSL connection using TLSv1.2`

使用 Guzzle 6 設定 TLS 1.2 的範例：

```
use Aws\DynamoDb\DynamoDbClient;
use Aws\Handler\GuzzleV6\GuzzleHandler;
use GuzzleHttp\Client;

$handler = new GuzzleHandler(
    new Client([
        'curl' => [
            CURLOPT_SSLVERSION => CURL_SSLVERSION_TLSv1_2
        ]
    ])
);

$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http_handler' => $handler
]);
```

> **Note**
>
> `http_handler` 選項會取代任何提供的 `handler` 選項。

profile

Type

string

當登入資料是從 HOME 目錄中的 AWS 登入資料檔案建立時，「設定檔」選項指定要使用的設定檔（通常為 `~/.aws/credentials`)。此設定將覆寫 `AWS_PROFILE` 環境變數。

> **Note**
>
> 當您指定「設定檔」選項時，會忽略該`"credentials"`選項，並忽略 AWS 組態檔案中的登入資料相關設定（通常為 `~/.aws/config`)。

```
// Use the "production" profile from your credentials file
$ec2 = new Aws\Ec2\Ec2Client([
    'version' => '2014-10-01',
    'region'  => 'us-west-2',
    'profile' => 'production'
]);
```

如需設定登入資料和 .ini 檔案格式的詳細資訊，請參閱第 3 適用於 PHP 的 AWS SDK 版的登入資料。

region

Type

string

必要

true

AWS 要連線的區域。如需可用區域的清單，請參閱區域AWS 和端點。

```
// Set the Region to the EU (Frankfurt) Region
$s3 = new Aws\S3\S3Client([
    'region'  => 'eu-central-1',
    'version' => '2006-03-01'
```

```
]);
```

retries

Type

`int|array|Aws\CacheInterface|Aws\Retry\ConfigurationInterface|callable`

預設

`int(3)`

設定用戶端的重試模式和允許重試次數上限。傳遞 `0` 以停用重試。

三種重試模式為：

- `legacy` - 預設舊版重試實作
- `standard` - 新增重試配額系統，以防止不太可能成功的重試
- `adaptive` - 在標準模式上建立，新增一個用戶端速率限制器。請注意，此模式為實驗性質。

重試的組態包含模式和每個請求所使用的嘗試次數上限。組態可以按照下列優先順序在幾個不同的位置進行設定。

優先順序

重試組態的優先順序如下 (1 覆寫 2-3 等等)：

1. 用戶端組態選項
2. 環境變數
3. AWS 共用組態檔案

環境變數

- `AWS_RETRY_MODE` - 設定為 `legacy`、`standard` 或 `adaptive`。
- `AWS_MAX_ATTEMPTS` - 設定為每個請求的最大嘗試次數的整數值

共用組態檔金鑰

- `retry_mode` - 設定為 `legacy`、`standard` 或 `adaptive`。

- `max_attempts` - 設定為每個請求的最大嘗試次數的整數值

用戶端組態

下列範例會停用 Amazon DynamoDB 用戶端的重試。

```
// Disable retries by setting "retries" to 0
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'retries' => 0
]);
```

下面的例子傳遞一個整數，這將預設為 `legacy` 模式與傳遞的重試次數

```
// Disable retries by setting "retries" to 0
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'retries' => 6
]);
```

該 **`Aws\Retry\Configuration`** 物件會接受兩個參數，重試模式

和每個請求的嘗試次數上限的整數。這個範例會傳入一個

`Aws\Retry\Configuration` 物件進行重試組態。

```
use Aws\EndpointDiscovery\Configuration;
use Aws\S3\S3Client;

$enabled = true;
$cache_limit = 1000;

$config = new Aws\Retry\Configuration('adaptive', 10);

$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-2',
    'retries' => $config,
]);
```

這個例子在一個陣列中傳遞重試配置。

```
use Aws\S3\S3Client;

$s3 = new S3Client([
    'region' => 'us-west-2',
    'retries' => [
        'mode' => 'standard',
        'max_attempts' => 7
    ],
]);
```

此範例會傳遞 `Aws\CacheInterface` 的執行個體，以快取預設重試組態提供者所傳回的值。

```
use Aws\DoctrineCacheAdapter;
use Aws\S3\S3Client;
use Doctrine\Common\Cache\ApcuCache;

$s3 = new S3Client([
    'region' => 'us-west-2',
    'endpoint_discovery' => new DoctrineCacheAdapter(new ApcuCache),
]);
```

結構描述

Type

`string`

預設

`string(5) "https"`

當連線時要使用的 URI 結構描述。依據預設，軟體開發套件會使用 “https” 端點 (即使用 SSL/TLS 連線)。您可以藉由將 `scheme` 設為 “http”，來嘗試透過未加密的 “http” 端點連線到服務。

```
$s3 = new Aws\S3\S3Client([
    'version' => '2006-03-01',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

如需端點清單，以及服務是否支援 `http`機制，請參閱AWS 區域和端點。

服務

Type

`string`

必要

true

要使用的名稱服務。依據預設，使用開發套件提供的用戶端 (亦即 `Aws\S3\S3Client`) 時，將提供此值。在測試尚未發佈至開發套件，但您已在磁碟上提供的服務時，此選項將很有用。

signature_provider

Type

`callable`

可呼叫，可接受簽章版本名稱 （例如 `v4`)、服務名稱和 AWS 區域，並傳回`Aws\Signature\SignatureInterface`物件，或`NULL`如果提供者能夠為指定參數建立簽署者。此供應商用於建立由用戶端使用的簽署者。

`Aws\Signature\SignatureProvider` 類別中的開發套件提供各種函數，可用於建立自訂的簽章供應商。

signature_version

Type

`string`

字串，表示要搭配服務 (例如，`v4` 等) 使用的自訂簽章版本。如有需要，每個操作簽章版本皆可能覆寫此請求的簽章版本。

下列範例示範如何設定 Amazon S3 用戶端以使用簽章第 4 版：

```
// Set a preferred signature version
$s3 = new Aws\S3\S3Client([
```

```
    'version'           => '2006-03-01',
    'region'            => 'us-west-2',
    'signature_version' => 'v4'
]);
```

> **Note**
>
> 您的用戶端使用的 `signature_provider` 必須能夠建立您提供的 `signature_version` 選項。軟體開發套件根據預設使用的 `signature_provider` 可建立 “v4” 和 “anonymous” 簽章版本的簽章物件。

ua_append

Type

`string|string[]`

預設

`[]`

新增到使用者代理程式字串的字串或字串陣列已傳送至 HTTP 處理常式。

use_aws_shared_config_files

Type

`bool|array`

預設

`bool(true)`

設定為 false 以停用檢查「~/.aws/config」和「~/.aws/credentials」中的共用組態檔案。這將覆寫 AWS_CONFIG_FILE 環境變數。

validate

Type

`bool|array`

預設

`bool(true)`

設定為 `false` 以停用用戶端側參數驗證。您可能會發現關閉驗證會稍微提高用戶端效能，但差異極為輕微。

```
// Disable client-side validation
$s3 = new Aws\S3\S3Client([
    'version'  => '2006-03-01',
    'region'   => 'eu-west-1',
    'validate' => false
]);
```

設定為驗證選項的關聯陣列，以啟用特定驗證限制：

- `required` - 驗證必要的參數是否存在 (預設為開啟)。
- `min` - 驗證值的最小長度 (預設為開啟)。
- `max` - 驗證值的最大長度值。
- `pattern` - 驗證值符合規則表達式。

```
// Validate only that required values are present
$s3 = new Aws\S3\S3Client([
    'version'  => '2006-03-01',
    'region'   => 'eu-west-1',
    'validate' => ['required' => true]
]);
```

version

Type

`string`

必要

false

此選項指定要使用的 Web 服務版本（例如 `2006-03-01`)。

從 SDK 的 3.277.10 版開始，不需要「版本」選項。如果您未指定「版本」選項，軟體開發套件會使用最新版本的服務用戶端。

當您建立服務用戶端時，兩種情況都需要「版本」參數。

- 您使用的 PHP 開發套件版本早於 3.277.10。
- 您使用 3.277.10 版或更新版本，並想要為服務用戶端使用「最新」以外的版本。

例如，下列程式碼片段使用 開發套件的 3.279.7 版，但不是 的最新版本`Ec2Client`。

```
$ec2Client = new \Aws\Ec2\Ec2Client([
    'version' => '2015-10-01',
    'region' => 'us-west-2'
]);
```

指定版本限制可確保您的程式碼將不會受重大服務變更的影響。

您可以在每個用戶端的 API 文件頁面上找到可用的 API 版本清單。如果您無法載入特定的 API 版本，可能需要更新您的開發套件。

AWS 區域 設定第 3 適用於 PHP 的 AWS SDK 版的

SDK 用戶端會連接到 AWS 區域 您在建立用戶端時指定的特定 AWS 服務 中的 。此組態可讓您的應用程式與該地理區域 AWS 的資源互動。當您在不明確設定區域的情況下建立服務用戶端時，軟體開發套件會使用外部組態的預設區域。

區域解析鏈

第 3 適用於 PHP 的 AWS SDK 版使用下列順序來判斷服務用戶端使用的區域：

1. 程式碼中提供的區域 - 如果您在用戶端建構函數選項中明確設定區域，這優先於所有其他來源。

   ```
   $s3Client = new Aws\S3\S3Client([
       'region' => 'us-west-2'
   ]);
   ```

2. 環境變數 - 如果程式碼中未提供任何區域，軟體開發套件會依序檢查這些環境變數：
   - `AWS_REGION`
   - `AWS_DEFAULT_REGION`

```
# Example of setting Region through environment variables.
export AWS_REGION=us-east-1
```

3. AWS 組態檔案 - 如果未設定區域環境變數，軟體開發套件會檢查 AWS 組態檔案：
   a. SDK 會尋找 `~/.aws/config`（或`AWS_CONFIG_FILE`環境變數指定的位置）
   b. SDK 會檢查`AWS_PROFILE`環境變數所指定設定檔內的區域設定
   c. 如果未指定 `AWS_PROFILE` ，軟體開發套件會使用「預設」描述檔

   例如，假設我們具有下列組態檔案設定：

   ```
   # Example ~/.aws/config file.
   [default]
   region = eu-west-1

   [profile production]
   region = eu-central-1
   ```

   如果`AWS_PROFILE`環境變數設定為「生產」值，用戶端會使用 `eu-central-1 Region`。如果沒有`AWS_PROFILE`環境變數，用戶端會使用 `eu-west-1`區域。
4. 如果開發套件在上述任何來源中找不到區域值，則會擲回例外狀況，因為區域值是服務用戶端的必要設定。

最佳實務

使用第 3 適用於 PHP 的 AWS SDK 版中的區域時，請考慮下列最佳實務：

在生產程式碼中明確設定區域

對於生產應用程式，我們建議您在程式碼中明確設定區域，而不是依賴環境變數或 `config`。這可讓您的程式碼更可預測，並減少對外部組態的依賴。

使用環境變數進行開發和測試

對於開發和測試環境，使用環境變數可提供更多彈性，而無需變更程式碼。

使用多個環境的設定檔

如果您的應用程式需要使用多個 AWS 環境，請考慮在您的 AWS `config` 檔案中使用不同的設定檔，並視需要在它們之間切換。

使用第 3 適用於 PHP 的 AWS SDK 版登入資料提供者

如需 AWS SDKs 可用登入資料機制的參考資訊，請參閱 AWS SDKs和工具參考指南中的登入資料和存取權。

> **Important**
>
> 為了安全起見，強烈建議您不要使用根帳戶進行 AWS 存取。如需最新的安全建議，請務必參閱《IAM 使用者指南》中的 IAM 中的安全最佳實務。

第 3 適用於 PHP 的 AWS SDK 版中登入資料提供者的角色是為開發套件的 AWS 服務 用戶端提供登入資料。SDK 會使用其來源的登入資料，透過密碼編譯方式簽署每個請求來驗證服務。登入資料通常由存取金鑰組成：存取金鑰 ID 和私密存取金鑰。

當您使用臨時登入資料時，例如當您設定 IAM Identity Center 身分驗證或設定執行時間以擔任 IAM 角色時，工作階段字符會新增至存取金鑰，以提供對 AWS 資源的限時存取。

第 3 適用於 PHP 的 AWS SDK 版中的登入資料提供者是什麼？

登入資料供應商是一個函數，會傳回一個 `GuzzleHttp\Promise\PromiseInterface`，其中包含 `Aws\Credentials\CredentialsInterface` 執行個體，或被退回並包含 `Aws\Exception\CredentialsException`。開發套件提供登入資料提供者函數的數個實作，或者您可以實作自己的自訂邏輯來建立登入資料或最佳化登入資料載入。

登入資料供應商會傳送至 `credentials` 用戶端建構函式選項。登入資料供應商為非同步，會在每次叫用 API 操作時，強制其進行延遲評估。因此，將登入資料提供者函數傳遞到軟體開發套件用戶端建構函數，不會立即驗證登入資料。如果登入資料提供者未傳回登入資料物件，API 操作將會遭到拒絕，並且其回應為 `Aws\Exception\CredentialsException`。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

// Use the ECS credential provider.
$provider = CredentialProvider::ecsCredentials();
// Be sure to memoize the credentials.
$memoizedProvider = CredentialProvider::memoize($provider);

// Pass the provider to the client
$client = new S3Client([
```

```
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $memoizedProvider
]);
```

了解第 3 適用於 PHP 的 AWS SDK 版中的預設登入資料提供者鏈結

預設登入資料提供者鏈結是由 SDK 調用的一系列內建登入資料提供者組成。它由 defaultProvider 登入資料提供者函數實作，沒有參數。找到有效的憑證後，系統就會停止搜尋。

會依下列順序 適用於 PHP 的 AWS SDK 執行登入資料提供者：

- **`env`** provider - 開發套件會搜尋AWS 已設定為環境變數的存取金鑰。
- **`assumeRoleWithWebIdentityCredentialProvider`** provider - SDK 會搜尋 IAM 角色和 Web 身分字符檔案設定。
- 在鏈結的這個時間點，軟體開發套件會尋找共用 AWS `config` 和 `credentials` 檔案中的組態。軟體開發套件會在「預設」設定檔下尋找組態，但如果已設定`AWS_PROFILE`環境變數，軟體開發套件會使用其具名設定檔值。
  - **`sso`** provider - SDK 會在共用`config`檔案中尋找 IAM Identity Center 組態設定。
  - **`process`** provider - SDK 會在共用`credentials`檔案中尋找 `credential_process`設定。
  - **`ini`** provider - SDK 會在共用`credentials`檔案中尋找 AWS 登入資料或 IAM 角色資訊。
  - **`process`** provider - SDK 會在共用`config`檔案中尋找 `credential_process`設定。
  - **`ini`** provider - SDK 會在共用`config`檔案中尋找 AWS 登入資料或 IAM 角色資訊。

- **`ecsCredentials`** provider - SDK 會尋找環境變數`AWS_CONTAINER_CREDENTIALS_FULL_URI`，`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`或提供取得臨時登入資料的資訊。
- **`instanceProfile`** provider - SDK 使用 EC2 執行個體中繼資料服務來取得執行個體描述檔中指定的 IAM 角色。開發套件會使用角色資訊來取得臨時登入資料。

> **Note**
>
> 預設供應商的結果會自動記憶。

您可以在 GitHub 原始程式碼中檢閱鏈結的程式碼。

第 3 適用於 PHP 的 AWS SDK 版中的內建登入資料提供者

開發套件提供數個內建登入資料提供者，您可以個別使用或合併在自訂登入資料提供者鏈中。

當您在服務用戶端建立期間指定登入資料提供者時，開發套件會嘗試僅使用指定的登入資料提供者載入登入資料。它不使用預設的登入資料提供者鏈結。如果您知道希望服務用戶端使用`instanceProfile`提供者，您可以透過在服務用戶端建構函數中指定`instanceProfile`提供者來使預設鏈結發生短路：

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::instanceProfile();
// Be sure to memoize the credentials
$memoizedProvider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'credentials' => $memoizedProvider  // The default credential provider chain is not
 used.
]);
```

> **Important**
>
> 每次執行 API 操作時，都會叫用登入資料供應商。如果載入登入資料是昂貴的任務 (例如，從磁碟或網路資源載入)，或如果登入資料並未由您的供應商快取，請考慮將登入資料供應商封裝在 `Aws\Credentials\CredentialProvider::memoize` 函數中。開發套件使用的預設登入資料供應商會自動記憶。

主題

- assumeRole 適用於 PHP 的 SDK 中的 提供者
- sso 適用於 PHP 的 開發套件中的 提供者
- defaultProvider 適用於 PHP 的 SDK 中的 提供者
- ecsCredentials 適用於 PHP 的 開發套件中的 提供者
- env 適用於 PHP 的 SDK 中的 提供者
- assumeRoleWithWebIdentityCredentialProvider 適用於 PHP 的 SDK 中的 提供者

- ini 適用於 PHP 的 SDK 中的 提供者
- process 適用於 PHP 的 SDK 中的 提供者
- instanceProfile 適用於 PHP 的 SDK 中的 提供者

`assumeRole` 適用於 PHP 的 SDK 中的 提供者

如果您使用 `Aws\Credentials\AssumeRoleCredentialProvider` 藉由採用角色來建立登入資料，您必須提供 `'client'` 資訊以及 `StsClient` 物件和 `'assume_role_params'` 詳細資訊，如下所示。

> **Note**
>
> 為了避免在每個 API 操作上不必要的擷取 AWS STS 登入資料，您可以使用 `memoize`函數來處理過期時自動重新整理登入資料。如需範例，請參閱下列程式碼。

```
use Aws\Credentials\CredentialProvider;
use Aws\Credentials\InstanceProfileProvider;
use Aws\Credentials\AssumeRoleCredentialProvider;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

// Passing Aws\Credentials\AssumeRoleCredentialProvider options directly
$profile = new InstanceProfileProvider();
$ARN = "arn:aws:iam::123456789012:role/xaccounts3access";
$sessionName = "s3-access-example";

$assumeRoleCredentials = new AssumeRoleCredentialProvider([
    'client' => new StsClient([
        'region' => 'us-east-2',
        'version' => '2011-06-15',
        'credentials' => $profile
    ]),
    'assume_role_params' => [
        'RoleArn' => $ARN,
        'RoleSessionName' => $sessionName,
    ],
]);

// To avoid unnecessarily fetching STS credentials on every API operation,
```

```
// the memoize function handles automatically refreshing the credentials when they
 expire
$provider = CredentialProvider::memoize($assumeRoleCredentials);

$client = new S3Client([
    'region'      => 'us-east-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

如需 `'assume_role_params'` 的詳細資訊，請參閱 AssumeRole。

`sso` 適用於 PHP 的 開發套件中的 提供者

`Aws\Credentials\CredentialProvider::sso` 是單一登入憑證提供者。此提供者也稱為 AWS IAM Identity Center 登入資料提供者。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$credentials = CredentialProvider::sso('profile default');

$s3 = new Aws\S3\S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

如果您使用具名設定檔，請在上一個範例中將設定檔的名稱替換為「`default`」。若要進一步了解如何設定具名設定檔，請參閱 SDK `config`和工具參考指南中的共用和`credentials`檔案。 AWS SDKs 或者，您可以使用 `AWS_PROFILE`環境變數來指定要使用的設定檔設定。

若要進一步了解 IAM Identity Center 提供者的運作方式，請參閱 AWS SDKs和工具參考指南中的了解 IAM Identity Center 身分驗證。

`defaultProvider` 適用於 PHP 的 SDK 中的 提供者

`Aws\Credentials\CredentialProvider::defaultProvider` 是預設登入資料提供者，也稱為預設登入資料提供者鏈結。如果在建立用戶端時省略 `credentials` 選項，將會使用此供應商。例如，如果您建立 S3Client，如以下程式碼片段所示，開發套件會使用預設提供者：

```
$client = new S3Client([
```

```
    'region' => 'us-west-2'
]);
```

如果您想要將參數提供給鏈結中的特定登入資料提供者，也可以在程式碼中使用 defaultProvider。例如，如果使用`ecsCredentials`提供者函數，以下範例會提供自訂連線逾時和重試設定。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::defaultProvider([
    'timeout' => '1.5',
    'retries' => 5
]);

$client = new S3Client([
    'region' => 'us-west-2',
    'credentials' => $provider
]);
```

`ecsCredentials` 適用於 PHP 的 開發套件中的 提供者

`Aws\Credentials\CredentialProvider::ecsCredentials` 嘗試以 `GET` 請求載入登入資料，其 URI 由容器中的環境變數 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 指定。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::ecsCredentials();
// Be sure to memoize the credentials
$memoizedProvider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $memoizedProvider
]);
```

`env` 適用於 PHP 的 SDK 中的 提供者

使用環境變數來包含您的登入資料，可防止您不小心共用 AWS 私密存取金鑰。我們建議您絕不在任何生產檔案中將 AWS 存取金鑰直接新增至用戶端。

若要向 Amazon Web Services 進行身分驗證，軟體開發套件會先檢查您環境變數中的登入資料。開發套件使用 `getenv()` 函數來尋找 `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY` 和 `AWS_SESSION_TOKEN` 環境變數。這類登入資料稱為環境登入資料。如需如何取得這些值的說明，請參閱 AWS SDKs和工具參考指南中的使用短期憑證進行身分驗證。

如果您在 上託管應用程式AWS Elastic Beanstalk，您可以透過 AWS Elastic Beanstalk 主控台設定 `AWS_SECRET_KEY`、 `AWS_ACCESS_KEY_ID`和 `AWS_SESSION_TOKEN`環境變數，讓 SDK 可以自動使用這些登入資料。

如需如何設定環境變數的詳細資訊，請參閱 AWS SDKs和工具參考指南中的環境變數支援。此外，如需 AWS SDKs支援的所有環境變數清單，請參閱環境變數清單。

您也可以在命令列中設定環境變數，如下所示。

Linux

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
   # The access key for your AWS ##.
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your AWS ##.
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your AWS ##.
   # The AWS_SECURITY_TOKEN environment variable can also be used, but is only
 supported for backward compatibility purposes.
   # AWS_SESSION_TOKEN is supported by multiple AWS SDKs other than PHP.
```

Windows

```
C:\> SET  AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
   # The access key for your AWS ##.
C:\> SET  AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your AWS ##.
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your AWS ##.
   # The AWS_SECURITY_TOKEN environment variable can also be used, but is only
 supported for backward compatibility purposes.
   # AWS_SESSION_TOKEN is supported by multiple AWS SDKs besides PHP.
```

`Aws\Credentials\CredentialProvider::env` 嘗試從環境變數載入登入資料。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;
```

```
$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => CredentialProvider::env()
]);
```

assumeRoleWithWebIdentityCredentialProvider 適用於 PHP 的 SDK 中的 提供者

Aws\Credentials\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider 嘗試透過擔任角色載入登入資料。如果環境變數 AWS_ROLE_ARN 和 AWS_WEB_IDENTITY_TOKEN_FILE 存在，供應商將嘗試使用磁碟上的字符擔任 AWS_ROLE_ARN 中指定的角色，而此字符位於 AWS_WEB_IDENTITY_TOKEN_FILE 中指定的完整路徑。如果使用環境變數，供應商將嘗試從 AWS_ROLE_SESSION_NAME 環境變數設定工作階段。

如果未設定環境變數，提供者將使用預設設定檔，或設定為 AWS_PROFILE 的設定檔。根據預設，供應商會從 ~/.aws/config 和 ~/.aws/credentials 讀取設定檔，而且可從 filename 組態選項中指定的設定檔讀取。供應商將擔任設定檔的 role_arn 中的角色，從 web_identity_token_file 中設定的完整路徑讀取字符。如果設定在設定檔上，將使用 role_session_name。

供應商會做為預設鏈的一部分進行呼叫，也可以直接呼叫。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::assumeRoleWithWebIdentityCredentialProvider();
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

在預設情況下，這位登入資料供應商會繼承將由 StsClient 用來擔任該角色的已設定區域。您可以選擇提供完整的 StsClient。在任何提供的 StsClient 上，登入資料應設定為 false。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$stsClient = new StsClient([
    'region'      => 'us-west-2',
    'version'     => 'latest',
    'credentials' => false
])

$provider = CredentialProvider::assumeRoleWithWebIdentityCredentialProvider([
    'stsClient' => $stsClient
]);
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

`ini` 適用於 PHP 的 SDK 中的 提供者

`Aws\Credentials\CredentialProvider::ini` 會嘗試從共用 `config`和 `credentials` 檔案載入登入資料。根據預設，軟體開發套件會嘗試從位於 的共用 AWS `credentials`檔案載入「預設」設定檔`~/.aws/credentials`。如果 SDK 找到`AWS_SDK_LOAD_NONDEFAULT_CONFIG`環境變數，也會在位於 的共用 AWS `config`檔案中檢查「預設」設定檔`~/.aws/config`。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::ini();
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
```

```
]);
```

您可以將引數提供給建立供應商的函數，以使用自訂描述檔或 .ini 檔案位置。

```
$profile = 'production';
$path = '/full/path/to/credentials.ini';

$provider = CredentialProvider::ini($profile, $path);
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

`process` 適用於 PHP 的 SDK 中的 提供者

`Aws\Credentials\CredentialProvider::process` 會嘗試透過執行共用 AWS 組態檔案中設定檔中指定的`credential_process`值來載入登入資料。

根據預設，軟體開發套件會先嘗試從位於 的共用 AWS `credentials`檔案載入「預設」描述檔`~/.aws/credentials`。如果在共用`credentials`檔案中找不到「預設」設定檔，軟體開發套件會在共用`config`檔案中尋找預設設定檔。以下是共用`credentials`檔案的組態範例。

```
[default]
credential_process = /path/to/file/credential_returning_executable.sh --custom-command
 custom_parameter
```

SDK 會使用 PHP 的 `shell_exec`函數完全呼叫 `credential_process`命令，然後從 stdout 讀取 JSON 資料。`credential_process` 必須以下列格式將登入資料寫入 stdout：

```
{
    "Version": 1,
    "AccessKeyId": "",
    "SecretAccessKey": "",
    "SessionToken": "",
    "Expiration": ""
}
```

`SessionToken` 和 `Expiration` 為選用。如果存在，則該登入資料會被視為臨時。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::process();
// Cache the results in a memoize function to avoid loading and parsing
// the ini file on every API operation
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

您可以將引數提供給建立供應商的函數，以使用自訂描述檔或 .ini 檔案位置。

```
$profile = 'production';
$path = '/full/path/to/credentials.ini';

$provider = CredentialProvider::process($profile, $path);
$provider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
    'version'     => '2006-03-01',
    'credentials' => $provider
]);
```

`instanceProfile` 適用於 PHP 的 SDK 中的 提供者

`Aws\Credentials\CredentialProvider::instanceProfile` 會嘗試載入 Amazon EC2 執行個體設定檔中指定之 IAM 角色的登入資料。

```
use Aws\Credentials\CredentialProvider;
use Aws\S3\S3Client;

$provider = CredentialProvider::instanceProfile();
// Be sure to memoize the credentials
$memoizedProvider = CredentialProvider::memoize($provider);

$client = new S3Client([
    'region'      => 'us-west-2',
```

```
    'version'     => '2006-03-01',
    'credentials' => $memoizedProvider
]);
```

根據預設，供應商會重試擷取登入資料，最多三次。您可以使用 `retries`選項設定重試次數，並將 選項設定為 以完全停用`0`，如下列程式碼所示。

```
use Aws\Credentials\CredentialProvider;

$provider = CredentialProvider::instanceProfile([
    'retries' => 0
]);
$memoizedProvider = CredentialProvider::memoize($provider);
```

如果環境變數`AWS_METADATA_SERVICE_NUM_ATTEMPTS`可用，其值會優先於先前顯示的「重試」選項。

> **Note**
>
> 您可以將`AWS_EC2_METADATA_DISABLED`環境變數設定為 ，以停用從 Amazon EC2 執行個體描述檔載入的嘗試`true`。

在適用於 PHP 的 SDK 中鏈結登入資料提供者

您可使用 `Aws\Credentials\CredentialProvider::chain()` 函數來鏈結登入資料供應商。此函數接受 variadic 數量的引數，每個引數皆為登入資料供應商函數。此函數會傳回一個新的函數，這個新函數由提供的函數組成，並且會逐一呼叫，直到其中一個提供者傳回已成功履行的 promise。

`defaultProvider` 使用此結構在失敗之前檢查多個供應商。`defaultProvider` 的來源示範如何使用 `chain` 函數。

```
// This function returns a provider
public static function defaultProvider(array $config = [])
{
    // This function is the provider, which is actually the composition
    // of multiple providers. Notice that we are also memoizing the result by
    // default.
    return self::memoize(
```

```
        self::chain(
            self::env(),
            self::ini(),
            self::instanceProfile($config)
        )
    );
}
```

建立自訂登入資料提供者以搭配適用於 PHP 的 SDK 使用

登入資料供應商是簡單的函數，被叫用時會傳回承諾 (`GuzzleHttp\Promise\PromiseInterface`)，其中包含 `Aws\Credentials\CredentialsInterface` 物件，或被退回並包含 `Aws\Exception\CredentialsException`。

建立供應商的最佳實務是建立函數，此函數被叫用時間和建立實際的登入資料供應商。例如，以下是 env 提供者的來源 (為了示範用途而有稍做修改)。請注意，這是一個函數，會傳回實際的供應商函數。這可讓您輕鬆地撰寫登入資料供應商，並傳遞它們做為值。

```
use GuzzleHttp\Promise;
use GuzzleHttp\Promise\RejectedPromise;

// This function CREATES a credential provider
public static function env()
{
    // This function IS the credential provider
    return function () {
        // Use credentials from environment variables, if available
        $key = getenv(self::ENV_KEY);
        $secret = getenv(self::ENV_SECRET);
        if ($key && $secret) {
            return Create::promise_for(
                new Credentials($key, $secret, getenv(self::ENV_SESSION))
            );
        }

        $msg = 'Could not find environment variable '
            . 'credentials in ' . self::ENV_KEY . '/' . self::ENV_SECRET;
        return new RejectedPromise(new CredentialsException($msg));
    };
}
```

在適用於 PHP 的 SDK 中記住登入資料

有時您可能需要建立能記憶先前傳回值的登入資料供應商。如果載入登入資料是昂貴的操作，或使用 `Aws\Sdk` 類別跨多個用戶端共用登入資料供應商時，這將會很有用。您可以藉由將登入資料供應商函數包裝在 `memoize` 函數中，將記憶功能新增至登入資料供應商。

```
use Aws\Credentials\CredentialProvider;

$provider = CredentialProvider::instanceProfile();
// Wrap the actual provider in a memoize function
$provider = CredentialProvider::memoize($provider);

// Pass the provider into the Sdk class and share the provider
// across multiple clients. Each time a new client is constructed,
// it will use the previously returned credentials as long as
// they haven't yet expired.
$sdk = new Aws\Sdk(['credentials' => $provider]);

$s3 = $sdk->getS3(['region' => 'us-west-2', 'version' => 'latest']);
$ec2 = $sdk->getEc2(['region' => 'us-west-2', 'version' => 'latest']);

assert($s3->getCredentials() === $ec2->getCredentials());
```

當記憶的登入資料過期時，記憶包裝函式會叫用包裝的供應商，以嘗試重新整理登入資料。

使用第 3 適用於 PHP 的 AWS SDK 版擔任 IAM 角色

針對 Amazon EC2 執行個體變數登入資料使用 IAM 角色

如果您在 Amazon EC2 執行個體上執行應用程式，提供憑證以呼叫 的偏好方式 AWS 是使用 IAM 角色來取得臨時安全憑證。

當您使用 IAM 角色時，不需要擔心應用程式的登入資料管理。它們允許執行個體從 Amazon EC2 執行個體的中繼資料伺服器擷取臨時憑證，以「擔任」角色。

暫時登入資料通常稱為執行個體描述檔登入資料，允許存取角色政策允許的動作和資源。Amazon EC2 會處理安全向 IAM 服務驗證執行個體以擔任角色的所有工作，並定期重新整理擷取的角色登入資料。這讓您幾乎無需執行任何工作，即可確保應用程式安全。如需接受臨時安全登入資料的 服務清單，請參閱《AWS IAM 使用者指南》中的使用 IAM 的 服務。

> **Note**
>
> 為避免每次都必須連接中繼資料服務，您可將 `Aws\CacheInterface` 執行個體做為 `'credentials'` 選項傳遞至用戶端建構函式。如此可讓開發套件改為使用快取的執行個體描述檔登入資料。如需詳細資訊，請參閱第 3 版的 適用於 PHP 的 AWS SDK 組態。

如需使用 SDKs 開發 Amazon EC2 應用程式的詳細資訊，請參閱 AWS SDKs和工具參考指南中的為 Amazon EC2 執行個體使用 IAM 角色。

建立 IAM 角色並將其指派給 Amazon EC2 執行個體

1. 建立 IAM 用戶端。

   匯入

   ```
   require 'vendor/autoload.php';

   use Aws\Iam\IamClient;
   ```

   範例程式碼

   ```
   $client = new IamClient([
       'region' => 'us-west-2',
       'version' => '2010-05-08'
   ]);
   ```

2. 建立具有您將使用之動作和資源許可的 IAM 角色。

   範例程式碼

   ```
   $result = $client->createRole([
       'AssumeRolePolicyDocument' => 'IAM JSON Policy', // REQUIRED
       'Description' => 'Description of Role',
       'RoleName' => 'RoleName', // REQUIRED
   ]);
   ```

3. 建立 IAM 執行個體描述檔，並從結果中存放 Amazon Resource Name (ARN)。

> **Note**
>
> 如果您使用 IAM 主控台而非 適用於 PHP 的 AWS SDK，主控台會自動建立執行個體描述檔，並為其提供與其對應之角色相同的名稱。
>
> 範例程式碼
>
> ```
> $IPN = 'InstanceProfileName';
>
> $result = $client->createInstanceProfile([
>  'InstanceProfileName' => $IPN ,
> ]);
>
> $ARN = $result['Arn'];
> $InstanceID =  $result['InstanceProfileId'];
> ```

4. 建立 Amazon EC2 用戶端。

   匯入

   ```
   require 'vendor/autoload.php';

   use Aws\Ec2\Ec2Client;
   ```

   範例程式碼

   ```
   $ec2Client = new Ec2Client([
       'region' => 'us-west-2',
       'version' => '2016-11-15',
   ]);
   ```

5. 將執行個體描述檔新增至執行中或已停止的 Amazon EC2 執行個體。使用 IAM 角色的執行個體描述檔名稱。

   範例程式碼

   ```
   $result = $ec2Client->associateIamInstanceProfile([
       'IamInstanceProfile' => [
           'Arn' => $ARN,
           'Name' => $IPN,
       ],
   ```

```
    'InstanceId' => $InstanceID
]);
```

如需更多詳細資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EC2 的 IAM 角色。

針對 Amazon ECS 任務使用 IAM 角色

Amazon Elastic Container Service (Amazon ECS) 中的任務可以擔任 IAM 角色來進行 AWS API 呼叫。這是管理應用程式要使用之登入資料的策略，類似於 Amazon EC2 執行個體描述檔如何提供登入資料給 Amazon EC2 執行個體。

您可以將使用臨時 AWS 憑證的 IAM 角色與 ECS 任務定義或 `RunTask` API 操作建立關聯，而不是建立和分發長期憑證至容器或使用 Amazon EC2 執行個體的 角色。

如需使用容器任務可擔任之 IAM 角色的詳細資訊，請參閱《Amazon ECS 開發人員指南》中的任務 IAM 角色主題。如需在任務定義`taskRoleArn`中以 形式使用任務 IAM 角色的範例，請參閱《Amazon ECS 開發人員指南》中的任務定義範例。

在另一個 中擔任 IAM 角色 AWS 帳戶

當您使用 AWS 帳戶 （帳戶 A) 並想要擔任另一個帳戶 （帳戶 B) 中的角色時，您必須先在帳戶 B 中建立 IAM 角色。此角色允許帳戶 （帳戶 A) 中的實體在帳戶 B 中執行特定動作。如需跨帳戶存取的詳細資訊，請參閱教學課程：使用 IAM 角色委派跨 AWS 帳戶存取。

在帳戶 B 中建立角色後，請記下角色 ARN。當您從帳戶 A 擔任角色時，您將使用此 ARN。您使用帳戶 A 中與您的實體相關聯的 AWS 登入資料來擔任角色。

使用 的登入資料建立 AWS STS 用戶端 AWS 帳戶。在下文中我們使用登入資料設定檔，但您可以使用任何方法。對於新建立的 AWS STS 用戶端，呼叫 assume-role 並提供自訂 sessionName。擷取結果中的新臨時登入資料。根據預設，登入資料會持續一小時。

範例程式碼

```
$stsClient = new Aws\Sts\StsClient([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2011-06-15'
]);

$ARN = "arn:aws:iam::123456789012:role/xaccounts3access";
```

```
$sessionName = "s3-access-example";

$result = $stsClient->AssumeRole([
      'RoleArn' => $ARN,
      'RoleSessionName' => $sessionName,
]);

 $s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' =>  [
        'key'    => $result['Credentials']['AccessKeyId'],
        'secret' => $result['Credentials']['SecretAccessKey'],
        'token'  => $result['Credentials']['SessionToken']
    ]
]);
```

如需詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的使用 IAM 角色或 AssumeRole。

使用具有 Web 身分的 IAM 角色

Web Identity Federation 可讓客戶在存取 AWS 資源時使用第三方身分提供者進行身分驗證。您必須先建立 IAM 角色並設定 Web 身分提供者 (IdP)，然後才能擔任具有 Web 身分的角色。如需詳細資訊，請參閱建立 Web 身分的角色或 OpenID Connect 聯合身分 (主控台)。

建立身分提供者並為您的 Web 身分建立角色之後，請使用 AWS STS 用戶端來驗證使用者。為您的身分提供 webIdentityToken 和 ProviderId，以及為使用者提供具有許可之 IAM 角色的角色 ARN。

範例程式碼

```
$stsClient = new Aws\Sts\StsClient([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2011-06-15'
]);

$ARN = "arn:aws:iam::123456789012:role/xaccounts3access";
$sessionName = "s3-access-example";
$duration = 3600;

$result = $stsClient->AssumeRoleWithWebIdentity([
      'WebIdentityToken' => "FACEBOOK_ACCESS_TOKEN",
```

```
      'ProviderId' => "graph.facebook.com",
      'RoleArn' => $ARN,
      'RoleSessionName' => $sessionName,
]);

 $s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' =>  [
        'key'    => $result['Credentials']['AccessKeyId'],
        'secret' => $result['Credentials']['SecretAccessKey'],
        'token'  => $result['Credentials']['SessionToken']
    ]
]);
```

如需詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 AssumeRoleWithWebIdentity—透過 Web 型身分提供者的聯合或 AssumeRoleWithWebIdentity。

使用設定檔擔任角色

在 中定義設定檔 `~/.aws/credentials`

您可以透過在 中定義設定檔， 適用於 PHP 的 AWS SDK 將 設定為使用 IAM 角色`~/.aws/credentials`。

使用您要擔任之角色`role_arn`的設定建立新的設定檔。同時包含另一個設定檔`source_profile`的設定，其登入資料具有擔任 IAM 角色的許可。如需這些組態設定的詳細資訊，請參閱 AWS SDKs和工具參考指南中的擔任角色登入資料。

例如，在下列 中`~/.aws/credentials`，`project1`設定檔會設定 `role_arn`，並將`default`設定檔指定為登入資料的來源，以確認與其相關聯的實體可以擔任該角色。

```
[project1]
role_arn = arn:aws:iam::123456789012:role/testing
source_profile = default
role_session_name = OPTIONAL_SESSION_NAME

[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
aws_session_token= YOUR_AWS_SESSION_TOKEN
```

如果您設定AWS_PROFILE環境變數，或在執行個體化服務用戶端時使用 profile 參數，project1則會使用default設定檔做為來源登入資料，擔任 中指定的角色。

下列程式碼片段顯示 S3Client建構函數中使用 profile 參數。S3Client 將具有與project1設定檔相關聯之角色相關聯的許可。

```
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-1',
    'version' => '2006-03-01',
    'profile' => 'project1'
]);
```

在 中定義設定檔 ~/.aws/config

~/.aws/config 檔案也可以包含您要擔任的設定檔。如果您設定環境變數 AWS_SDK_LOAD_NONDEFAULT_CONFIG，適用於 PHP 的 SDK 會從 config 檔案載入設定檔。設定 AWS_SDK_LOAD_NONDEFAULT_CONFIG 時，軟體開發套件會從 ~/.aws/config和 載入設定檔~/.aws/credentials。來自 的設定檔~/.aws/credentials最後載入，優先於~/.aws/config具有相同名稱的 設定檔。來自任一位置的設定檔都能做為 source_profile 或要擔任的設定檔。

下列範例使用 config 檔案中定義的 project1 設定檔和 credentials 檔案中的 default設定檔。AWS_SDK_LOAD_NONDEFAULT_CONFIG 也會設定 。

```
# Profile in ~/.aws/config.

[profile project1]
role_arn = arn:aws:iam::123456789012:role/testing
source_profile = default
role_session_name = OPTIONAL_SESSION_NAME
```

```
# Profile in ~/.aws/credentials.

[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
aws_session_token= YOUR_AWS_SESSION_TOKEN
```

當S3Client建構函式執行並顯示下列程式碼片段時，project1設定檔中定義的角色將使用與default設定檔相關聯的登入資料來擔任。

```
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-1',
    'version' => '2006-03-01',
    'profile' => 'project1'
]);
```

在適用於 PHP 的 SDK AWS STS 中使用 的臨時登入資料

AWS Security Token Service (AWS STS) 可讓您為 IAM 使用者或您透過聯合身分驗證的使用者請求有限權限、臨時憑證。如需進一步了解，請參閱《IAM 使用者指南》中的暫時安全登入資料。您可以使用臨時安全登入資料來存取大多數 AWS 服務。如需接受臨時安全登入資料的 服務清單，請參閱《AWS IAM 使用者指南》中的使用 IAM 的 服務。

臨時憑證的一個常見使用案例是透過第三方身分提供者驗證使用者，授予行動或用戶端應用程式存取 AWS 資源的權限（請參閱 Web Identity Federation)。

取得臨時登入資料

AWS STS 有數個 操作會傳回暫時登入資料，但`GetSessionToken`操作是最簡單的示範。下列程式碼片段會透過呼叫 PHP 開發套件 STS 用戶端的 `getSessionToken`方法來擷取臨時憑證。

```
$sdk = new Aws\Sdk([
    'region'   => 'us-east-1',
]);

$stsClient = $sdk->createSts();

$result = $stsClient->getSessionToken();
```

`GetSessionToken` 和其他 AWS STS 操作的結果一律包含 `'Credentials'`值。如果您列印 `$result`（例如使用 `print_r($result)`），它看起來如下所示。

```
Array
(
    ...
    [Credentials] => Array
    (
        [SessionToken] => '<base64 encoded session token value>'
        [SecretAccessKey] => '<temporary secret access key value>'
        [Expiration] => 2013-11-01T01:57:52Z
```

```
        [AccessKeyId] => '<temporary access key value>'
    )
    ...
)
```

提供臨時登入資料給 適用於 PHP 的 AWS SDK

您可以透過執行個體化 AWS 用戶端並 AWS STS 直接傳入從 接收的值，將暫時登入資料與其他用戶端搭配使用。

```
use Aws\S3\S3Client;

$result = $stsClient->getSessionToken();

$s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' => [
        'key'    => $result['Credentials']['AccessKeyId'],
        'secret' => $result['Credentials']['SecretAccessKey'],
        'token'  => $result['Credentials']['SessionToken']
    ]
]);
```

您也可以建構 `Aws\Credentials\Credentials` 物件，並在個體化用戶端時使用它。

```
use Aws\Credentials\Credentials;
use Aws\S3\S3Client;

$result = $stsClient->getSessionToken();

$credentials = new Credentials(
    $result['Credentials']['AccessKeyId'],
    $result['Credentials']['SecretAccessKey'],
    $result['Credentials']['SessionToken']
);

$s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

不過，提供臨時登入資料的最佳方式是使用隨附於 `StsClient` 的 `createCredentials()` helper 方法。此方法會從 AWS STS 結果中擷取資料，並為您建立 `Credentials` 物件。

```
$result = $stsClient->getSessionToken();
$credentials = $stsClient->createCredentials($result);

$s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' => $credentials
]);
```

如需有關為什麼您可能需要在應用程式或專案中使用暫時登入資料的詳細資訊，請參閱 AWS STS 文件中的授予暫時存取權的案例。

在適用於 PHP 的 SDK 中建立匿名用戶端

在某些情況下，您可能會希望建立未與任何登入資料關聯的用戶端。如此可讓您對服務進行匿名請求。

例如，您可以同時設定 Amazon S3 物件和 Amazon CloudSearch 網域，以允許匿名存取。

若要建立匿名用戶端，請將 `'credentials'` 選項設定為 `false`。

```
$s3Client = new S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
    'credentials' => false
]);

// Makes an anonymous request. The object would need to be publicly
// readable for this to succeed.
$result = $s3Client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'my-key',
]);
```

在適用於 PHP 的 SDK 中使用 AWS 通用執行期 (AWS CRT) 延伸模組

AWS CRT 程式庫為數個 AWS SDKs 提供具有良好效能和最少佔用空間的基本功能。本主題討論適用於 PHP 的 開發套件何時使用 AWS CRT，以及如何安裝 AWS CRT 延伸模組。

當您需要安裝 AWS CRT 延伸模組時

適用於 PHP 的 SDK 使用 AWS CRT 程式庫的授權和檢查總和功能。當您使用 時，需要 AWS CRT 延伸模組：

- Amazon S3 多區域存取點
- Amazon EventBridge 全域端點
- Amazon Simple Storage Service (Amazon S3) 中的 CRC-32C 檢查總和演算法

如果您使用上面列出的功能，且您的 PHP 環境中未安裝 AWS CRT 範圍，適用於 PHP 的 SDK 將報告錯誤訊息，並提醒您安裝 擴充功能。

安裝 AWS 通用執行期 (AWS CRT) 延伸模組

有關如何安裝 AWS CRT 擴充功能的說明，請參閱 aws-crt-php 的 GitHub 儲存庫主頁面。

使用第 3 適用於 PHP 的 AWS SDK 版

本章提供使用第 3 適用於 PHP 的 AWS SDK 版與 AWS 服務 有效互動的完整指導。您將學習提出服務請求、處理錯誤和利用進階 SDK 功能在 上建置強大 PHP 應用程式的基本技術 AWS。

無論您是建置簡單的應用程式還是複雜的系統，本章所述的技術都可協助您最佳化程式碼、改善錯誤處理，以及實作有效的模式，以便 AWS 服務 透過 PHP 開發套件使用 。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版提出 AWS 服務 請求
- 使用第 3 適用於 PHP 的 AWS SDK 版進行非同步程式設計
- 處理第 3 適用於 PHP 的 AWS SDK 版中的錯誤
- 第 3 適用於 PHP 的 AWS SDK 版中的處理常式和中介軟體
- 第 3 適用於 PHP 的 AWS SDK 版中的串流
- 第 3 適用於 PHP 的 AWS SDK 版中使用分頁結果
- 第 3 適用於 PHP 的 AWS SDK 版中的等待程式
- 第 3 適用於 PHP 的 AWS SDK 版中的 JMESPath 表達式

使用第 3 適用於 PHP 的 AWS SDK 版提出 AWS 服務 請求

SDK 請求工作流程概觀

使用 第 3 適用於 PHP 的 AWS SDK 版會遵循所有 的一致模式 AWS 服務。基本工作流程包含三個主要步驟：

1. 建立服務用戶端 - 為 AWS 服務 您要使用的 建立用戶端物件。
2. 執行操作 - 在用戶端上呼叫對應至服務 API 中操作的方法。
3. 程序結果 - 使用成功傳回的類似陣列的結果物件，或處理失敗時擲出的任何例外狀況。

以下各節會詳細說明這些步驟，從如何建立和設定服務用戶端開始。

建立基本服務用戶端

您可以藉由將選項的關聯陣列傳遞至用戶端的建構函數，來建立用戶端。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

範例程式碼

```
//Create an S3Client
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-2'  // Since version 3.277.10 of the SDK,
]);                          // the 'version' parameter defaults to 'latest'.
```

您可以在組態選項主題中找到選用的「版本」參數的相關資訊。

請注意，我們並未明確提供登入資料給用戶端。這是因為 SDK 使用預設登入資料提供者鏈結來尋找登入資料資訊。

中會詳細說明所有一般用戶端組態選項第 3 適用於 PHP 的 AWS SDK 版的用戶端建構函式選項。提供給用戶端的選項陣列可能會因您正在建立的用戶端而不同。這些自訂用戶端組態選項的說明如各用戶端的 API 文件所述。

雖然上述範例顯示基本用戶端建立，但您可以自訂服務用戶端以符合特定需求。如需透過程式碼設定服務用戶端的詳細資訊，請參閱 在 第 3 適用於 PHP 的 AWS SDK 版的程式碼中設定服務用戶端。如果您需要使用外部組態檔案或環境變數來設定服務用戶端，請參閱 在外部設定第 3 適用於 PHP 的 AWS SDK 版的服務用戶端。

提出要求

您可以呼叫用戶端物件上相同名稱的 方法來提出服務請求。例如，若要執行 Amazon S3 PutObject 操作，請呼叫 `Aws\S3\S3Client::putObject()`方法。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

範例程式碼

```
// Use the us-east-2 region and latest version of each client.
$sharedConfig = [
    'profile' => 'default',
    'region' => 'us-east-2'
];

// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk($sharedConfig);

// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();

// Send a PutObject request and get the result object.
$result = $s3Client->putObject([
    'Bucket' => 'my-bucket',
    'Key' => 'my-key',
    'Body' => 'this is the body!'
]);

// Download the contents of the object.
$result = $s3Client->getObject([
    'Bucket' => 'my-bucket',
    'Key' => 'my-key'
]);

// Print the body of the result by indexing into the result object.
echo $result['Body'];
```

用戶端可使用的操作以及輸入與輸出的結構皆根據服務描述檔在執行時間時定義。建立用戶端時，如果您未提供服務模型的`version`參數（例如「2006-03-01」或「最新」），用戶端會預設為最新版本。開發套件根據提供的版本找到對應的組態檔案。

如 `putObject()` 的操作方法皆接受陳述式，為代表操作參數的相關陣列。此陣列的結構 (以及結果物件的結構) 已為軟體開發套件 API 文件中的各項操作定義 (例如，請參閱 putObject 操作的 API 文件)。

HTTP 處理常式選項

您也可以使用特殊 `@http` 參數來微調基礎 HTTP 處理常式如何執行請求。您可以在 `@http` 參數中包含的選項與您在使用 “http” 用戶端選項執行個體化用戶端時可設定的選項相同。

```
// Send the request through a proxy
```

```
$result = $s3Client->putObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'my-key',
    'Body'   => 'this is the body!',
    '@http'  => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);
```

使用結果物件

執行成功操作會傳回 `Aws\Result` 物件。除了傳回原始 XML 或服務的 JSON 資料外，開發套件會強迫回應資料變為關聯性的陣列結構。將根據對於特定服務的知識以及基礎回應結構來標準化資料中的部分面向。

您可以從`AWS\Result`物件存取資料，例如關聯的 PHP 陣列。

匯入

```
require 'vendor/autoload.php';
use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

範例程式碼

```
// Use the us-east-2 region and latest version of each client.
$sharedConfig = [
    'profile' => 'default',
    'region' => 'us-east-2',
];

// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk($sharedConfig);

// Use an Aws\Sdk class to create the S3Client object.
$s3 = $sdk->createS3();
$result = $s3->listBuckets();
foreach ($result['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}

// Convert the result object to a PHP array
```

```
$array = $result->toArray();
```

結果物件的內容會根據執行的操作以及服務版本而定。每個 API 操作的結果結構將根據各操作記錄於 API 文件中。

開發套件與 JMESPath 整合，此 DSL 用於搜尋及操控 JSON 資料，即如同本文件用於操控 PHP 陣列。結果物件包含您可以用於更明顯宣告從結果中擷取資料的 `search()` 方法。

範例程式碼

```
$s3 = $sdk->createS3();
$result = $s3->listBuckets();
```

```
$names = $result->search('Buckets[].Name');
```

第 3 適用於 PHP 的 AWS SDK 版中的命令物件

適用於 PHP 的 AWS SDK 使用 命令模式來封裝參數和處理常式，稍後將用於傳輸 HTTP 請求。

命令的隱含使用

如果您檢查任何用戶端類別，則可以看到與 API 操作對應的方法實際並上不存在。它們使用 `__call()` 神奇方法來實作。這些虛擬方法實際上是封裝軟體開發套件使用命令物件的捷徑。

您通常不需要與命令物件直接互動。您呼叫像 `Aws\S3\S3Client::putObject()` 這樣的方法時，開發套件實際上會根據提供的參數建立一個 `Aws\CommandInterface` 對象，執行該命令並傳回一個填入的 `Aws\ResultInterface` 物件 (或者在錯誤時擲出例外)。呼叫用戶端的任何 `Async` 方法 (例如 `Aws\S3\S3Client::putObjectAsync()`) 時都會發生類似的流程：用戶端根據提供的參數建立命令、序列化 HTTP 請求、啟動請求並傳回 promise。

以下範例具有相同功能。

```
$s3Client = new Aws\S3\S3Client([
    'version' => '2006-03-01',
    'region'  => 'us-standard'
]);

$params = [
    'Bucket' => 'foo',
    'Key'    => 'baz',
```

```
    'Body'   => 'bar'
];

// Using operation methods creates a command implicitly
$result = $s3Client->putObject($params);

// Using commands explicitly
$command = $s3Client->getCommand('PutObject', $params);
$result = $s3Client->execute($command);
```

命令參數

所有命令都支援一些特殊參數，這些參數不是服務 API 的一部分，而會控制軟體開發套件的行為。

@http

使用此參數可以微調基礎 HTTP 處理器如何執行請求。您可以在 `@http` 參數中包含的選項與您在使用 "http" 用戶端選項執行個體化用戶端時可設定的選項相同。

```
// Configures the command to be delayed by 500 milliseconds
$command['@http'] = [
    'delay' => 500,
];
```

@retries

如同「重試」用戶端選項，`@retries` 控制著命令在遭視為失敗之前可以重試的次數。將它設為 `0` 以停用重試。

```
// Disable retries
$command['@retries'] = 0;
```

> **Note**
>
> 如果您已停用了用戶端上的重試，則無法在傳送給該用戶端的個別命令上將它們選擇性啟用。

建立命令物件

您可以使用用戶端的 `getCommand()` 方法建立命令。其不會立即執行或傳輸 HTTP 請求，而只會在傳遞給用戶端的 `execute()` 方法時才執行。這使您有機會在執行命令之前修改命令物件。

```
$command = $s3Client->getCommand('ListObjects');
$command['MaxKeys'] = 50;
$command['Prefix'] = 'foo/baz/';
$result = $s3Client->execute($command);

// You can also modify parameters
$command = $s3Client->getCommand('ListObjects', [
    'MaxKeys' => 50,
    'Prefix'  => 'foo/baz/',
]);
$command['MaxKeys'] = 100;
$result = $s3Client->execute($command);
```

命令 HandlerList

從用戶端建立命令時，會為其提供複製的用戶端 `Aws\HandlerList` 物件。該命令會獲得用戶端處理器清單的一個複製，允許命令使用不影響用戶端所執行其他命令的自訂中介軟體和處理器。

這表示您可以對每個命令 (例如 `Aws\MockHandler`) 使用不同的 HTTP 用戶端，並透過中介軟體為每個命令新增自訂行為。以下範例使用 `MockHandler` 建立模擬結果，而不傳送實際的 HTTP 請求。

```
use Aws\Result;
use Aws\MockHandler;

// Create a mock handler
$mock = new MockHandler();
// Enqueue a mock result to the handler
$mock->append(new Result(['foo' => 'bar']));
// Create a "ListObjects" command
$command = $s3Client->getCommand('ListObjects');
// Associate the mock handler with the command
$command->getHandlerList()->setHandler($mock);
// Executing the command will use the mock handler, which returns the
// mocked result object
$result = $client->execute($command);

echo $result['foo']; // Outputs 'bar'
```

除了變更命令使用的處理常式之外，還可以將自訂中介軟體注入命令。以下範例使用 `tap` 中介軟體，該中介軟體在處理常式清單中做為觀察者。

```
use Aws\CommandInterface;
```

```
use Aws\Middleware;
use Psr\Http\Message\RequestInterface;

$command = $s3Client->getCommand('ListObjects');
$list = $command->getHandlerList();

// Create a middleware that just dumps the command and request that is
// about to be sent
$middleware = Middleware::tap(
    function (CommandInterface $command, RequestInterface $request) {
        var_dump($command->toArray());
        var_dump($request);
    }
);

// Append the middleware to the "sign" step of the handler list. The sign
// step is the last step before transferring an HTTP request.
$list->append('sign', $middleware);

// Now transfer the command and see the var_dump data
$s3Client->execute($command);
```

CommandPool

Aws\CommandPool 使您能夠使用產生 Aws\CommandInterface 物件的反覆運算器並同時執行命令。CommandPool 確保在反覆集區中的命令時，同時執行固定數量的命令 (命令完成時執行更多命令，以確保集區大小恆定)。

這是一個非常簡單的範例，只需用 CommandPool 傳送一些命令即可。

```
use Aws\S3\S3Client;
use Aws\CommandPool;

// Create the client
$client = new S3Client([
    'region'  => 'us-standard',
    'version' => '2006-03-01'
]);

$bucket = 'example';
$commands = [
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'a']),
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'b']),
```

```
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'c'])
];

$pool = new CommandPool($client, $commands);

// Initiate the pool transfers
$promise = $pool->promise();

// Force the pool to complete synchronously
$promise->wait();
```

這個範例對於 CommandPool 來說非常不足。讓我們來試試更複雜的範例。假設您想要將磁碟上的檔案上傳至 Amazon S3 儲存貯體。如果要從磁碟中取得檔案清單，我們可以使用 PHP 的 DirectoryIterator。此反覆運算器會產生 SplFileInfo 物件。CommandPool 接受一個產生 Aws\CommandInterface 物件的反覆運算器，所以我們會對應 SplFileInfo 物件來傳回 Aws\CommandInterface 物件。

```
<?php
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
use Aws\CommandPool;
use Aws\CommandInterface;
use Aws\ResultInterface;
use GuzzleHttp\Promise\PromiseInterface;

// Create the client
$client = new S3Client([
    'region'  => 'us-standard',
    'version' => '2006-03-01'
]);

$fromDir = '/path/to/dir';
$toBucket = 'amzn-s3-demo-bucket';

// Create an iterator that yields files from a directory
$files = new DirectoryIterator($fromDir);

// Create a generator that converts the SplFileInfo objects into
// Aws\CommandInterface objects. This generator accepts the iterator that
// yields files and the name of the bucket to upload the files to.
$commandGenerator = function (\Iterator $files, $bucket) use ($client) {
```

```
    foreach ($files as $file) {
        // Skip "." and ".." files
        if ($file->isDot()) {
            continue;
        }
        $filename = $file->getPath() . '/' . $file->getFilename();
        // Yield a command that is executed by the pool
        yield $client->getCommand('PutObject', [
            'Bucket' => $bucket,
            'Key'    => $file->getBaseName(),
            'Body'   => fopen($filename, 'r')
        ]);
    }
};

// Now create the generator using the files iterator
$commands = $commandGenerator($files, $toBucket);

// Create a pool and provide an optional array of configuration
$pool = new CommandPool($client, $commands, [
    // Only send 5 files at a time (this is set to 25 by default)
    'concurrency' => 5,
    // Invoke this function before executing each command
    'before' => function (CommandInterface $cmd, $iterKey) {
        echo "About to send {$iterKey}: "
            . print_r($cmd->toArray(), true) . "\n";
    },
    // Invoke this function for each successful transfer
    'fulfilled' => function (
        ResultInterface $result,
        $iterKey,
        PromiseInterface $aggregatePromise
    ) {
        echo "Completed {$iterKey}: {$result}\n";
    },
    // Invoke this function for each failed transfer
    'rejected' => function (
        AwsException $reason,
        $iterKey,
        PromiseInterface $aggregatePromise
    ) {
        echo "Failed {$iterKey}: {$reason}\n";
    },
]);
```

```
// Initiate the pool transfers
$promise = $pool->promise();

// Force the pool to complete synchronously
$promise->wait();

// Or you can chain the calls off of the pool
$promise->then(function() { echo "Done\n"; });
```

`CommandPool` 組態

`Aws\CommandPool` 建構函式接受各種組態選項。

並行 (可呼叫|int)

同時執行的最大命令數。提供一個動態調整集區規模的函數。此函數提供目前等待中的請求數量，並且預計會傳回一個代表新集區規模限制的整數。

之前 (可呼叫)

在傳送每個命令之前呼叫的函數。`before` 函數接受該命令之反覆運算器的命令和金鑰。傳送命令之前，可以視需要在 `before` 函數中改變命令。

已履行 (可呼叫)

promise 履行時呼叫的函數。該函數提供結果物件，產生結果的反覆運算器 ID，以及如果需要將集區短路時可以解決或拒絕的彙總 promise。

拒絕 (可呼叫)

promise 被拒絕時呼叫的函數。該函數提供 `Aws\Exception` 物件，產生例外的反覆運算器 ID，以及如果需要將集區短路時可以解決或拒絕的彙總 promise。

命令之間的手動垃圾收集

如果您達到大型命令集區的記憶體限制，這可能是因為在達到記憶體限制時，PHP 廢棄項目收集器尚未收集軟體開發套件產生的循環參考。在命令之間手動叫用集合演算法會允許在達到該限制前收集循環。以下範例會在傳送每個命令前，使用回呼建立會叫用集合演算法的 `CommandPool`。請注意，叫用廢棄項目收集器不會伴隨效能成本，最佳使用方式將取決於您的使用案例和環境。

```
$pool = new CommandPool($client, $commands, [
    'concurrency' => 25,
```

```
    'before' => function (CommandInterface $cmd, $iterKey) {
        gc_collect_cycles();
    }
]);
```

使用第 3 適用於 PHP 的 AWS SDK 版進行非同步程式設計

您可以使用開發套件的非同步功能來同時傳送命令。您可以在操作名稱中加入 `Async` 前綴來以非同步方式傳送請求。這將啟動請求並傳回承諾。

承諾以成功或以例外狀況遭拒的結果物件履行。這可讓您建立多個承諾，並在基礎 HTTP 處理常式傳輸請求時讓它們同時傳送 HTTP 請求。

匯入

```
require 'vendor/autoload.php';
use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

範例程式碼

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2'
]);
// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();
//Listing all S3 Bucket
$CompleteSynchronously = $s3Client->listBucketsAsync();
// Block until the result is ready.
$CompleteSynchronously = $CompleteSynchronously->wait();
```

您可以透過使用承諾的 `wait` 方法同步強制承諾完成。強制 promise 完成的同時也會根據預設「取消包裝」promise 的狀態，表示其會傳回 promise 的結果或者拋出遇到的例外狀況。在承諾上呼叫 `wait()` 時，直到 HTTP 請求完成且發布結果或丟出例外狀況前，處理程序皆會封鎖。

以事件迴圈程式庫來使用軟體開發套件時，請不要封鎖結果。相反地，請使用結果的 `then()` 方法來存取在操作完成時解決或拒絕的承諾。

匯入

```
require 'vendor/autoload.php';
use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

範例程式碼

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2'
]);
// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();
```

```
$promise = $s3Client->listBucketsAsync();
$promise
    ->then(function ($result) {
        echo 'Got a result: ' . var_export($result, true);
    })
    ->otherwise(function ($reason) {
        echo 'Encountered an error: ' . $reason->getMessage();
    });
```

第 3 適用於 PHP 的 AWS SDK 版中的 Promise

適用於 PHP 的 AWS SDK 使用 promise 來允許非同步工作流程，而此非同步性允許同時傳送 HTTP 請求。由開發套件使用的 promise 規格是 Promises/A+。

什麼是承諾？

Promise 代表非同步操作的最終結果。與 promise 主要互動的方式，是透過其 `then` 方法。此方法註冊回呼以接收 promise 的最終值或 promise 無法履行的理由。

適用於 PHP 的 AWS SDK 倚賴 guzzlehttp/promises Composer 套件進行承諾實作。Guzzle promises 支援封鎖和非封鎖的工作流程，並可與任何非封鎖事件迴圈一起使用。

> **Note**
>
> HTTP 請求會使用 適用於 PHP 的 AWS SDK 單一執行緒在 中同時傳送，其中使用非封鎖呼叫來傳輸一或多個 HTTP 請求，同時回應狀態變更（例如履行或拒絕承諾）。

開發套件中的 Promise

Promise 用於整個軟體開發套件。例如，promise 用於由開發套件所提供的大多數高階抽象概念：分頁程式、等待程式、命令集區、分段上傳、S3 目錄/儲存貯體傳輸等。

當您呼叫用任何 `Async` 非同步尾碼方法時，開發套件提供的所有用戶端都會傳回 promise。例如，下列程式碼示範如何建立取得 Amazon DynamoDB`DescribeTable` 操作結果的承諾。

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'region'  => 'us-west-2',
    'version' => 'latest',
]);

// This will create a promise that will eventually contain a result
$promise = $client->describeTableAsync(['TableName' => 'mytable']);
```

請注意，您可以呼叫 `describeTable` 或 `describeTableAsync`。這些方法是由 API 模型和與用戶端關聯的 `__call` 編號支援的用戶端神奇 `version` 方法。透過呼叫不帶 `describeTable` 尾碼的 `Async` 方法，用戶端將在傳送 HTTP 請求時封鎖，並返回 `Aws\ResultInterface` 物件或擲出 `Aws\Exception\AwsException`。透過用 `Async` (即 `describeTableAsync`) 添加操作名稱尾碼，用戶端將建立一個最終由 `Aws\ResultInterface` 物件履行或被 `Aws\Exception\AwsException` 拒絕的 promise。

> **Important**
>
> 當 promise 傳回時，結果可能已經到達 (例如，當使用模擬處理常式時)，或者 HTTP 請求可能沒有啟動。

您可以使用 `then` 方法以 promise 註冊回呼。該方法接受兩個回呼，`$onFulfilled` 和 `$onRejected`，兩者都是非必要的。如果 promise 已履行，則會呼叫 `$onFulfilled` 回呼，如果 promise 被拒絕 (代表失敗)，則會呼叫 `$onRejected` 回呼。

```
$promise->then(
    function ($value) {
        echo "The promise was fulfilled with {$value}";
    },
    function ($reason) {
        echo "The promise was rejected with {$reason}";
    }
```

```
);
```

同時執行命令

多個 promise 可以組合在一起，以便同時執行。這可以透過將軟體開發套件與非封鎖事件迴路整合，或透過建立多個 promise 並等待它們同時完成來實現。

```
use GuzzleHttp\Promise\Utils;

$sdk = new Aws\Sdk([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

$s3 = $sdk->createS3();
$ddb = $sdk->createDynamoDb();

$promises = [
    'buckets' => $s3->listBucketsAsync(),
    'tables'  => $ddb->listTablesAsync(),
];

// Wait for both promises to complete.
$results = Utils::unwrap($promises);

// Notice that this method will maintain the input array keys.
var_dump($results['buckets']->toArray());
var_dump($results['tables']->toArray());
```

> **Note**
>
> CommandPool 提供更強大的機制以同時執行多個 API 操作。

鏈結承諾

Promise 的好處之一是它們可以組合，可讓您建立轉換管道。Promise 是由鏈結 `then` 回呼與後續 `then` 回呼構成。`then` 方法傳回的值是根據提供的回呼結果履行或拒絕的 promise。

```
$promise = $client->describeTableAsync(['TableName' => 'mytable']);

$promise
```

```
    ->then(
        function ($value) {
            $value['AddedAttribute'] = 'foo';
            return $value;
        },
        function ($reason) use ($client) {
            // The call failed. You can recover from the error here and
            // return a value that will be provided to the next successful
            // then() callback. Let's retry the call.
            return $client->describeTableAsync(['TableName' => 'mytable']);
        }
    )->then(
        function ($value) {
            // This is only invoked when the previous then callback is
            // fulfilled. If the previous callback returned a promise, then
            // this callback is invoked only after that promise is
            // fulfilled.
            echo $value['AddedAttribute']; // outputs "foo"
        },
        function ($reason) {
            // The previous callback was rejected (failed).
        }
    );
```

> **Note**
>
> Promise 回呼的傳回值是提供給下游 promise 的 `$value` 引數。如果您要為下游 provide 鏈提供值，則必須在回呼函數中傳回一個值。

拒絕轉送

promise 被拒絕時，您可以註冊一個回呼以呼叫。如果在任何回呼中擲出例外，promise 將被例外拒絕，且鏈中的下一個 promise 將被拒絕並產生例外。如果您從 `$onRejected` 回呼成功傳回一個值，則 promise 鏈中的下一個 promise 將使用來自 `$onRejected` 回呼傳回的值以履行。

等待承諾

您可以透過使用 promise 的 `wait` 方法同步強制 promise 完成。

```
$promise = $client->listTablesAsync();
$result = $promise->wait();
```

如果在呼叫 promise 的 `wait` 函數時遇到例外，那麼 promise 會被例外拒絕，且會擲出該例外。

```
use Aws\Exception\AwsException;

$promise = $client->listTablesAsync();

try {
    $result = $promise->wait();
} catch (AwsException $e) {
    // Handle the error
}
```

在已履行的 promise 上呼叫 `wait` 不會觸發等待函數。它只會傳回之前傳送的值。

```
$promise = $client->listTablesAsync();
$result = $promise->wait();
assert($result ### $promise->wait());
```

在已拒絕的 promise 上呼叫 `wait` 會擲出例外。如果拒絕原因是 `\Exception` 的一個執行個體，則擲出原因。否則，會擲出 `GuzzleHttp\Promise\RejectionException`，並且可以透過呼叫例外的 `getReason` 方法來獲取原因。

> **Note**
>
> 中的 API 操作呼叫 適用於 PHP 的 AWS SDK 會遭到 `Aws\Exception\AwsException`類別的子類別拒絕。但是，交付給 `then` 方法的原因可能不同，因為增加了變更拒絕原因的自訂中介軟體。

取消承諾

可以使用 promise 的 `cancel()` 方法取消 promise。如果 promise 已經解決，呼叫 `cancel()` 將不會生效。取消一個 promise 會取消該 promise，和任何等待該 promise 交付的 promise。一個遭到取消的 promise 會被 `GuzzleHttp\Promise\RejectionException` 拒絕。

結合承諾

您可以將 promise 結合到彙整 promise 中以建構更複雜的工作流程。`guzzlehttp/promise` 套件包含各種可用於結合 promise 的函數。

您可以在 namespace-GuzzleHttp.Promise 上找到所有 promise 集合函數的 API 文件。

each 和 each_limit

當您有`Aws\CommandInterface`命令的任務佇列可與固定集區大小同時執行時，請使用 CommandPool（命令可以位於記憶體中或由延遲迭代器產生）。`CommandPool` 確保同時傳送固定數量的命令，直到提供的反覆運算器耗盡為止。

`CommandPool` 僅適用於由同一用戶端執行的命令。您可以使用 `GuzzleHttp\Promise\each_limit` 函數，使用固定的集區大小同時執行不同用戶端的傳送命令。

```
use GuzzleHttp\Promise;

$sdk = new Aws\Sdk([
    'version' => 'latest',
    'region'  => 'us-west-2'
]);

$s3 = $sdk->createS3();
$ddb = $sdk->createDynamoDb();

// Create a generator that yields promises
$promiseGenerator = function () use ($s3, $ddb) {
    yield $s3->listBucketsAsync();
    yield $ddb->listTablesAsync();
    // yield other promises as needed...
};

// Execute the tasks yielded by the generator concurrently while limiting the
// maximum number of concurrent promises to 5
$promise = Promise\each_limit($promiseGenerator(), 5);

// Waiting on an EachPromise will wait on the entire task queue to complete
$promise->wait();
```

Promise coroutines

Guzzle promise 程式庫一個更強大的功能，是它允許您使用 promise coroutine，讓編寫非同步工作流看起來更像是編寫傳統的同步工作流。實際上， 適用於 PHP 的 AWS SDK 在大多數高層級抽象中使用 coroutine promise。

假設在儲存貯體可用時，您希望建立多個儲存貯體並上傳檔案到儲存貯體中，且想同時完成這些事，以便能夠盡快發生。您可以透過使用 `all()` promise 函數將多個 coroutine promise 組合在一起，輕鬆完成此任務。

```
use GuzzleHttp\Promise;

$uploadFn = function ($bucket) use ($s3Client) {
    return Promise\coroutine(function () use ($bucket, $s3Client) {
        // You can capture the result by yielding inside of parens
        $result = (yield $s3Client->createBucket(['Bucket' => $bucket]));
        // Wait on the bucket to be available
        $waiter = $s3Client->getWaiter('BucketExists', ['Bucket' => $bucket]);
        // Wait until the bucket exists
        yield $waiter->promise();
        // Upload a file to the bucket
        yield $s3Client->putObjectAsync([
            'Bucket' => $bucket,
            'Key'    => '_placeholder',
            'Body'   => 'Hi!'
        ]);
    });
};

// Create the following buckets
$buckets = ['foo', 'baz', 'bar'];
$promises = [];

// Build an array of promises
foreach ($buckets as $bucket) {
    $promises[] = $uploadFn($bucket);
}

// Aggregate the promises into a single "all" promise
$aggregate = Promise\all($promises);

// You can then() off of this promise or synchronously wait
$aggregate->wait();
```

處理第 3 適用於 PHP 的 AWS SDK 版中的錯誤

同步錯誤處理

執行操作時若發生錯誤，將顯示例外情況。因此，如果您需要處理程式碼中的錯誤，請在操作周圍使用 `try/catch` 區塊。開發套件發生錯誤時將丟出服務專屬的例外狀況。

下列為使用 Aws\S3\S3Client 的範例。如果發生錯誤，丟出的例外狀況將會是類型 Aws\S3\Exception\S3Exception。所有開發套件丟出的服務專屬的例外狀況皆從 Aws\Exception\AwsException 延伸而來。此類別包含關於失敗的實用資訊，包括請求 ID、錯誤碼和錯誤類型。請注意，對於某些支援它的服務，回應資料會強制轉換為關聯陣列結構 (類似於 Aws\Result 物件)，此資料可以如何一般 PHP 關聯陣列一樣進行存取。toArray() 方法將傳回任何這類資料，如果存在的話。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\S3\Exception\S3Exception;
```

範例程式碼

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2'
]);

// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();

try {
    $s3Client->createBucket(['Bucket' => 'my-bucket']);
} catch (S3Exception $e) {
    // Catch an S3 specific exception.
    echo $e->getMessage();
} catch (AwsException $e) {
    // This catches the more generic AwsException. You can grab information
    // from the exception using methods of the exception object.
    echo $e->getAwsRequestId() . "\n";
    echo $e->getAwsErrorType() . "\n";
    echo $e->getAwsErrorCode() . "\n";

    // This dumps any modeled response data, if supported by the service
    // Specific members can be accessed directly (e.g. $e['MemberName'])
    var_dump($e->toArray());
}
```

非同步錯誤處理

傳送非同步請求時不會丟出例外狀況。但是，您必須使用傳回承諾的 `then()` 或 `otherwise()` 方法接來收結果或錯誤。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\S3\Exception\S3Exception;
```

範例程式碼

```
//Asynchronous Error Handling
$promise = $s3Client->createBucketAsync(['Bucket' => 'my-bucket']);
$promise->otherwise(function ($reason) {
    var_dump($reason);
});

// This does the same thing as the "otherwise" function.
$promise->then(null, function ($reason) {
    var_dump($reason);
});
```

您可以「取消包裝」promise，並拋出例外狀況。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\S3\Exception\S3Exception;
```

範例程式碼

```
$promise = $s3Client->createBucketAsync(['Bucket' => 'my-bucket']);
```

```
//throw exception
try {
    $result = $promise->wait();
} catch (S3Exception $e) {
    echo $e->getMessage();
}
```

第 3 適用於 PHP 的 AWS SDK 版中的處理常式和中介軟體

擴展 的主要機制 適用於 PHP 的 AWS SDK 是透過處理常式和中介軟體。每種開發套件的用戶端類別均擁有 `Aws\HandlerList` 執行個體，其可透過用戶端的 `getHandlerList()` 方法存取。您可以擷取並修改用戶端的 `HandlerList`，藉此新增或移除用戶端行為。

處理常式

透過處理常式函數，使用者可以將命令與請求實際轉換為結果；且處理常式通常會傳送 HTTP 請求。為了增強自身行為，處理常式可以由中介軟體組成。處理常式是一個函數，它接受 `Aws\CommandInterface` 和 `Psr\Http\Message\RequestInterface` 並回傳一個以 `Aws\ResultInterface` 履行或以 `Aws\Exception\AwsException` 理由拒絕的 promise。

處理器會針對每次呼叫傳回相同的模擬結果，如下所示。

```
use Aws\CommandInterface;
use Aws\Result;
use Psr\Http\Message\RequestInterface;
use GuzzleHttp\Promise;

$myHandler = function (CommandInterface $cmd, RequestInterface $request) {
    $result = new Result(['foo' => 'bar']);
    return Promise\promise_for($result);
};
```

接著，您可以在用戶端的建構函式中提供 `handler` 選項，搭配開發套件用戶端來使用此處理常式。

```
// Set the handler of the client in the constructor
$s3 = new Aws\S3\S3Client([
```

```
    'region'  => 'us-east-1',
    'version' => '2006-03-01',
    'handler' => $myHandler
]);
```

您也可以在建構完成後，使用 `setHandler` 的 `Aws\ClientInterface` 方法，變更用戶端的處理常式。

```
// Set the handler of the client after it is constructed
$s3->getHandlerList()->setHandler($myHandler);
```

> **Note**
>
> 若要在建構多區域用戶端之後變更其處理常式，請使用 的 `useCustomHandler`方法`Aws\MultiRegionClient`。
>
> ```
> $multiRegionClient->useCustomHandler($myHandler);
> ```

模擬處理常式

我們建議您透過 `MockHandler` 來編寫使用開發套件的測試。您可以使用 `Aws\MockHandler` 以傳回模擬結果或擲回模擬例外狀況。您能將結果或例外狀況排入佇列，而 MockHandler 則會以 FIFO 順序將這些內容移出佇列。

```
use Aws\Result;
use Aws\MockHandler;
use Aws\DynamoDb\DynamoDbClient;
use Aws\CommandInterface;
use Psr\Http\Message\RequestInterface;
use Aws\Exception\AwsException;

$mock = new MockHandler();

// Return a mocked result
$mock->append(new Result(['foo' => 'bar']));

// You can provide a function to invoke; here we throw a mock exception
$mock->append(function (CommandInterface $cmd, RequestInterface $req) {
    return new AwsException('Mock exception', $cmd);
```

```
});

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'version' => 'latest',
    'handler' => $mock
]);

// Result object response will contain ['foo' => 'bar']
$result = $client->listTables();

// This will throw the exception that was enqueued
$client->listTables();
```

中介軟體

中介軟體是一種特殊類型的高階函數，其可增強傳輸命令和委派給「下一個」處理器的行為。中介軟體函數可接受 `Aws\CommandInterface` 和 `Psr\Http\Message\RequestInterface`，並回傳以 `Aws\ResultInterface` 履行或以 `Aws\Exception\AwsException` 理由拒絕的 promise。

中介軟體屬於高階函數，且可修改通過中介軟體的命令、請求或結果。中介軟體所採用的格式如下所示。

```
use Aws\CommandInterface;
use Psr\Http\Message\RequestInterface;

$middleware = function () {
    return function (callable $handler) use ($fn) {
        return function (
            CommandInterface $command,
            RequestInterface $request = null
        ) use ($handler, $fn) {
            // Do something before calling the next handler
            // ...
            $promise = $fn($command, $request);
            // Do something in the promise after calling the next handler
            // ...
            return $promise;
        };
    };
};
```

中介軟體會收到待執行的命令，以及選用的請求物件。而中介軟體可選擇增強請求與命令，或選擇保持原樣。然後，中介軟體會呼叫鏈結中的下一個處理常式，也可能會縮短下一個處理常式並傳回 promise。而透過呼叫下一個處理常式所建立的 promise，可以使用 promise 的 `then` 方法來進行增強，以便先修改最終結果或錯誤，再傳回 promise 並備份堆疊的中介軟體。

HandlerList

本開發套件會透過 `Aws\HandlerList`，善加管理執行命令時所使用的中介軟體與處理常式。每種開發套件的用戶端均擁有 `HandlerList`，且系統會複製此 `HandlerList`，並將其新增至用戶端所建立的每個命令。您可以在用戶端的 `HandlerList` 中新增一個中介軟體，藉此連接中介軟體與預設處理器，以用於用戶端所建立的每個命令。若要從特定命令中新增和移除中介軟體，則可以修改特定命令所擁有的 `HandlerList`。

`HandlerList` 代表中介軟體堆疊，用以包裝處理常式。為了協助您管理中介軟體清單並安排包裝處理常式的順序，`HandlerList` 會將堆疊的中介軟體分解為指定步驟，而這些步驟即為傳輸命令生命週期的一部分：

1. `init` - 新增預設參數
2. `validate` - 驗證必要參數
3. `build` - 將待傳送的 HTTP 請求序列化
4. `sign` - 簽署序列化的 HTTP 請求
5. <handler> (並非步驟，但會執行實際傳輸)

init

此生命週期步驟表示系統會將命令初始化，但尚未將請求序列化。這個步驟通常會用來將預設參數新增至命令。

您可以透過 `init` 與 `appendInit` 方法，將中介軟體新增至 `prependInit` 步驟；`appendInit` 會將中介軟體新增至 `prepend` 清單的結尾處，而 `prependInit` 則會將中介軟體新增至 `prepend` 清單的起始處。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});
```

```
// Append to the end of the step with a custom name
$client->getHandlerList()->appendInit($middleware, 'custom-name');
// Prepend to the beginning of the step
$client->getHandlerList()->prependInit($middleware, 'custom-name');
```

validate

此生命週期步驟旨在驗證命令的輸入參數。

您可以透過 `validate` 與 `appendValidate` 方法，將中介軟體新增至 `prependValidate` 步驟；`appendValidate` 會將中介軟體新增至 `validate` 清單的結尾處，而 `prependValidate` 則會將中介軟體新增至 `validate` 清單的起始處。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});

// Append to the end of the step with a custom name
$client->getHandlerList()->appendValidate($middleware, 'custom-name');
// Prepend to the beginning of the step
$client->getHandlerList()->prependValidate($middleware, 'custom-name');
```

build

此生命週期步驟會針對正在執行的命令，將 HTTP 請求序列化。下游生命週期事件將收到命令與 PSR-7 HTTP 請求。

您可以透過 `build` 與 `appendBuild` 方法，將中介軟體新增至 `prependBuild` 步驟；`appendBuild` 會將中介軟體新增至 `build` 清單的結尾處，而 `prependBuild` 則會將中介軟體新增至 `build` 清單的起始處。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});

// Append to the end of the step with a custom name
$client->getHandlerList()->appendBuild($middleware, 'custom-name');
// Prepend to the beginning of the step
```

```
$client->getHandlerList()->prependBuild($middleware, 'custom-name');
```

sign

在透過線路傳送 HTTP 請求前，通常會使用此生命週期步驟來簽署該請求。一般而言，您應該避免在簽署 HTTP 請求後進行修改，以防止發生簽章錯誤。

此步驟為處理常式傳輸 HTTP 請求之前，所要執行的最後一個 `HandlerList` 步驟。

您可以透過 `sign` 與 `appendSign` 方法，將中介軟體新增至 `prependSign` 步驟；`appendSign` 會將中介軟體新增至 `sign` 清單的結尾處，而 `prependSign` 則會將中介軟體新增至 `sign` 清單的起始處。

```
use Aws\Middleware;

$middleware = Middleware::tap(function ($cmd, $req) {
    // Observe the step
});

// Append to the end of the step with a custom name
$client->getHandlerList()->appendSign($middleware, 'custom-name');
// Prepend to the beginning of the step
$client->getHandlerList()->prependSign($middleware, 'custom-name');
```

可用的中介軟體

本開發套件提供多種中介軟體，以供您增強用戶端行為或查看命令的執行狀況。

mapCommand

如果您在將命令序列化為 HTTP 請求前，需要修改該命令，則 `Aws\Middleware::mapCommand` 中介軟體相當實用。例如，`mapCommand` 可用來執行驗證或新增預設參數。而 `mapCommand` 函數所接受的呼叫，會應允 `Aws\CommandInterface` 物件並傳回 `Aws\CommandInterface` 物件。

```
use Aws\Middleware;
use Aws\CommandInterface;

// Here we've omitted the require Bucket parameter. We'll add it in the
// custom middleware.
$command = $s3Client->getCommand('HeadObject', ['Key' => 'test']);
```

```
// Apply a custom middleware named "add-param" to the "init" lifecycle step
$command->getHandlerList()->appendInit(
    Middleware::mapCommand(function (CommandInterface $command) {
        $command['Bucket'] = 'amzn-s3-demo-bucket';
        // Be sure to return the command!
        return $command;
    }),
    'add-param'
);
```

mapRequest

如果您已經將請求序列化但尚未進行傳送，則 `Aws\Middleware::mapRequest` 中介軟體有助於您修改該請求。例如，此中介軟體可用來將自訂 HTTP 標頭新增至請求。而 `mapRequest` 函數所接受的呼叫，會應允 `Psr\Http\Message\RequestInterface` 引數並傳回 `Psr\Http\Message\RequestInterface` 物件。

```
use Aws\Middleware;
use Psr\Http\Message\RequestInterface;

// Create a command so that we can access the handler list
$command = $s3Client->getCommand('HeadObject', [
    'Key'    => 'test',
    'Bucket' => 'amzn-s3-demo-bucket'
]);

// Apply a custom middleware named "add-header" to the "build" lifecycle step
$command->getHandlerList()->appendBuild(
    Middleware::mapRequest(function (RequestInterface $request) {
        // Return a new request with the added header
        return $request->withHeader('X-Foo-Baz', 'Bar');
    }),
    'add-header'
);
```

現在當您執行命令時，系統會一併傳送該命令與自訂標頭。

> **Important**
>
> 請注意，系統會在結束 `build` 步驟時，將中介軟體附加至處理常式清單。如此一來，便可確保系統在呼叫此中介軟體前，已經成功建立請求。

mapResult

如果您需要修改命令執行的結果，則 `Aws\Middleware::mapResult` 中介軟體相當實用。`mapResult` 函數所接受的呼叫，會應允 `Aws\ResultInterface` 引數並傳回 `Aws\ResultInterface` 物件。

```
use Aws\Middleware;
use Aws\ResultInterface;

$command = $s3Client->getCommand('HeadObject', [
    'Key'    => 'test',
    'Bucket' => 'amzn-s3-demo-bucket'
]);

$command->getHandlerList()->appendSign(
    Middleware::mapResult(function (ResultInterface $result) {
        // Add a custom value to the result
        $result['foo'] = 'bar';
        return $result;
    })
);
```

現在當您執行命令時，系統傳回的結果將會包含 `foo` 屬性。

歷程記錄

`history` 中介軟體有助於您測試開發套件是否成功執行預期的命令、成功傳送預期的 HTTP 請求，以及成功接收預期的結果。此中介軟體基本上與 web 瀏覽器的歷史記錄功能類似。

```
use Aws\History;
use Aws\Middleware;

$ddb = new Aws\DynamoDb\DynamoDbClient([
    'version' => 'latest',
    'region'  => 'us-west-2'
]);

// Create a history container to store the history data
$history = new History();

// Add the history middleware that uses the history container
$ddb->getHandlerList()->appendSign(Middleware::history($history));
```

依據預設，在系統清除項目前，Aws\History 歷史記錄容器可以存放 10 個項目。如果您要自訂項目數，則可以將要保留的項目數傳入建構函式。

```
// Create a history container that stores 20 entries
$history = new History(20);
```

當您執行通過 history 中介軟體的請求後，即可檢查歷史記錄容器。

```
// The object is countable, returning the number of entries in the container
count($history);

// The object is iterable, yielding each entry in the container
foreach ($history as $entry) {
    // You can access the command that was executed
    var_dump($entry['command']);
    // The request that was serialized and sent
    var_dump($entry['request']);
    // The result that was received (if successful)
    var_dump($entry['result']);
    // The exception that was received (if a failure occurred)
    var_dump($entry['exception']);
}

// You can get the last Aws\CommandInterface that was executed. This method
// will throw an exception if no commands have been executed.
$command = $history->getLastCommand();

// You can get the last request that was serialized. This method will throw an
 exception
// if no requests have been serialized.
$request = $history->getLastRequest();

// You can get the last return value (an Aws\ResultInterface or Exception).
// The method will throw an exception if no value has been returned for the last
// executed operation (e.g., an async request has not completed).
$result = $history->getLastReturn();

// You can clear out the entries using clear
$history->clear();
```

tap

您可以將 tap 中介軟體做為觀察程式使用。當您透過中介軟體鏈結來傳送命令時，便可以使用此中介軟體呼叫函數。tap 函數所接受的呼叫，會應允 Aws\CommandInterface，以及系統所執行的選用 Psr\Http\Message\RequestInterface。

```
use Aws\Middleware;

$s3 = new Aws\S3\S3Client([
    'region'  => 'us-east-1',
    'version' => '2006-03-01'
]);

$handlerList = $s3->getHandlerList();

// Create a tap middleware that observes the command at a specific step
$handlerList->appendInit(
    Middleware::tap(function (CommandInterface $cmd, RequestInterface $req = null) {
        echo 'About to send: ' . $cmd->getName() . "\n";
        if ($req) {
            echo 'HTTP method: ' . $request->getMethod() . "\n";
        }
    }
);
```

建立自訂處理常式

處理常式即為一個函數，可接受 Aws\CommandInterface 物件與 Psr\Http\Message\RequestInterface 物件，且會回傳以 GuzzleHttp\Promise\PromiseInterface 履行或以 Aws\ResultInterface 拒絕的 Aws\Exception\AwsException。

縱使開發套件提供多種 @http 選項，但處理常式僅需了解下列選項的使用方式：

- connect_timeout
- debug
- decode_content (選用)
- 延遲
- progress (選用)
- proxy
- sink

- synchronous (選用)
- stream (選用)
- timeout
- 驗證
- http_stats_receiver (選用) - 如果使用 stats 組態參數進行請求，即透過 HTTP 傳輸統計資料的關聯陣列來呼叫此函數。

除非將選項指定為選用，否則處理常式必須能夠處理選項，或必須傳回遭拒絕的 promise。

除了處理特定的 `@http` 選項之外，處理器還必須新增採用下列格式的 `User-Agent` 標頭；您可以使用 `Aws\Sdk::VERSION` 來取代以下格式中的 “3.X”，並用您的處理器特定的 User-Agent 字串來取代 “HandlerSpecificData/version ...”。

```
User-Agent: aws-sdk-php/3.X HandlerSpecificData/version ...
```

第 3 適用於 PHP 的 AWS SDK 版中的串流

作為 PSR-7 HTTP 訊息標準整合的一部分， 會在內部 適用於 PHP 的 AWS SDK 使用 PSR-7 StreamInterface 作為其 PHP 串流的抽象。任何在輸入欄位中定義為 blob 的命令 (例如：BodyS3::PutObject 命令上的 參數)，皆可透過字串、PHP 串流資源，或是 `Psr\Http\Message\StreamInterface` 執行個體滿足自身需求。

> **Warning**
>
> 軟體開發套件擁有任何原始 PHP 串流資源的所有權，而系統會將這類資源以輸入參數的形式提供給命令使用。此軟體開發套件會代您耗用並關閉該串流。
> 如果您需要在開發套件操作與程式碼之間共享串流，請先將該串流包裝在 `GuzzleHttp\Psr7\Stream` 執行個體中，隨後再將其納為命令參數。軟體開發套件會使用串流，因此程式碼需要考量串流的內部游標移動。Guzzle 串流會在 PHP 的廢棄項目收集器銷毀 `fclose` 時，立即在基礎串流資源上呼叫該程式碼，所以您不需要自行關閉串流。

串流裝飾項目

透過 Guzzle 提供的各種串流裝飾項目，您可以控制開發套件、Guzzle 與串流資源的互動方式，而這些串流資源會以輸入參數的形式提供給命令使用。您可以善用這些裝飾項目，修改處理常式在指定串流中進行讀取和尋找的方式。以下提供局部清單；如需更完整的資訊，請參閱 GuzzleHttpPsr7 儲存庫。

AppendStream

GuzzleHttp\Psr7\AppendStream

逐一讀取多個串流。

```
use GuzzleHttp\Psr7;

$a = Psr7\stream_for('abc, ');
$b = Psr7\stream_for('123.');
$composed = new Psr7\AppendStream([$a, $b]);

$composed->addStream(Psr7\stream_for(' Above all listen to me'));

echo $composed(); // abc, 123. Above all listen to me.
```

CachingStream

GuzzleHttp\Psr7\CachingStream

允許使用者在無法尋找的串流上，尋找先前的讀取位元組。舉例來說，重新導向會造成系統必須倒轉串流，若是因此而導致無法尋找的實體主體傳輸失敗，則此裝飾項目相當實用。系統會將讀取自遠端串流的資料暫存於 PHP 臨時串流中，以便先在記憶體內快取先前的讀取位元組，接著再從磁碟上進行快取。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for(fopen('http://www.google.com', 'r'));
$stream = new Psr7\CachingStream($original);

$stream->read(1024);
echo $stream->tell();
// 1024

$stream->seek(0);
echo $stream->tell();
// 0
```

InflateStream

GuzzleHttp\Psr7\InflateStream

透過 PHP 的 zlib.inflate 篩選條件來解壓縮或壓縮 gzip 內容。

此串流裝飾項目會略過指定串流的前十個位元組，藉此移除 gzip 標頭，並將供選用的串流轉換成 PHP 串流資源，最後再附加 zlib.inflate 篩選。然後，系統會將該串流轉換回 Guzzle 串流資源，以便做為 Guzzle 串流使用。

LazyOpenStream

GuzzleHttp\Psr7\LazyOpenStream

延遲讀取或編寫檔案，該檔案只有在串流上執行 I/O 操作後才會開啟。

```
use GuzzleHttp\Psr7;

$stream = new Psr7\LazyOpenStream('/path/to/file', 'r');
// The file has not yet been opened...

echo $stream->read(10);
// The file is opened and read from only when needed.
```

LimitStream

GuzzleHttp\Psr7\LimitStream

用來讀取現有串流物件的子集或切片。這對於將大型檔案分成較小的部分以區塊（例如 Amazon S3 分段上傳 API) 傳送非常有用。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for(fopen('/tmp/test.txt', 'r+'));
echo $original->getSize();
// >>> 1048576

// Limit the size of the body to 1024 bytes and start reading from byte 2048
$stream = new Psr7\LimitStream($original, 1024, 2048);
echo $stream->getSize();
// >>> 1024
echo $stream->tell();
// >>> 0
```

NoSeekStream

GuzzleHttp\Psr7\NoSeekStream

包裝串流，且無法進行尋找。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for('foo');
$noSeek = new Psr7\NoSeekStream($original);

echo $noSeek->read(3);
// foo
var_export($noSeek->isSeekable());
// false
$noSeek->seek(0);
var_export($noSeek->read(3));
// NULL
```

PumpStream

GuzzleHttp\Psr7\PumpStream

提供唯讀串流，進而從 PHP 可呼叫功能中擷取資料。

當系統呼叫供選用的可呼叫功能時，PumpStream 會將請求讀取的資料量傳遞至可呼叫功能。可呼叫功能可以選擇忽略此值，並傳回低於或高於請求的位元組。透過供選用可呼叫功能所傳回的任何額外資料，皆會暫存於內部，直到 PumpStream 的 read() 函數耗盡為止。如果沒有更多要讀取的資料，則供選用的可呼叫更能必須傳回 false。

實作串流裝飾項目

建立串流裝飾項目的操作十分簡易，這都要歸功於 GuzzleHttp\Psr7\StreamDecoratorTrait。此特徵所提供的方式，旨在透過基礎串流代理來實作 `Psr\Http\Message\StreamInterface`。因此，您只需 `use StreamDecoratorTrait` 並實作自訂方法即可。

例如，假設我們想要在每次從串流讀取最後一個位元組時呼叫特定的函數。即可透過覆寫 `read()` 方法來進行實作。

```
use Psr\Http\Message\StreamInterface;
use GuzzleHttp\Psr7\StreamDecoratorTrait;

class EofCallbackStream implements StreamInterface
{
    use StreamDecoratorTrait;
```

```
    private $callback;

    public function __construct(StreamInterface $stream, callable $cb)
    {
        $this->stream = $stream;
        $this->callback = $cb;
    }

    public function read($length)
    {
        $result = $this->stream->read($length);

        // Invoke the callback when EOF is hit
        if ($this->eof()) {
            call_user_func($this->callback);
        }

        return $result;
    }
}
```

您可以將此裝飾項目新增至任何現有串流，而使用方法則如下所示。

```
use GuzzleHttp\Psr7;

$original = Psr7\stream_for('foo');

$eofStream = new EofCallbackStream($original, function () {
    echo 'EOF!';
});

$eofStream->read(2);
$eofStream->read(1);
// echoes "EOF!"
$eofStream->seek(0);
$eofStream->read(3);
// echoes "EOF!"
```

第 3 適用於 PHP 的 AWS SDK 版中使用分頁結果

有些 AWS 服務操作會分頁，並以截斷的結果回應。例如，Amazon S3ListObjects 操作一次最多只會傳回 1，000 個物件。類似這些 (名稱前方通常會加上「列出」或「描述」) 的操作需要以字符 (或標記) 參數進行後續請求，以擷取整組結果。

分頁程式是 的一項功能 適用於 PHP 的 AWS SDK ，可做為此程序的抽象概念，讓開發人員更輕鬆地使用分頁 APIs。分頁程式基本上是結果的疊代運算。它們是透過用戶端的 getPaginator() 方法建立的。當您呼叫 getPaginator() 時，必須提供操作的名稱和操作的引數 (其方式與您執行操作時相同)。您可以使用 foreach 逐一查看分頁程式物件，以取得個別的 Aws\Result 物件。

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'amzn-s3-demo-bucket'
]);

foreach ($results as $result) {
    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
}
```

分頁程式物件

getPaginator() 方法傳回的物件是 Aws\ResultPaginator 類別的執行個體。此類別會實作 PHP 原生 iterator 界面，使其能與 foreach 搭配運作。其亦可用於疊代運算函數如 iterator_to_array，以及整合 SPL 疊代運算如 LimitIterator 物件。

分頁程式物件一次只能保留一個結果的「頁面」，並且會延遲執行。這表示它們只會執行所需的請求，以產生結果的目前頁面。例如，Amazon S3ListObjects 操作一次最多只會傳回 1，000 個物件，因此如果您的儲存貯體有 ~10，000 個物件，分頁器就需要執行總共 10 個請求。當您逐一查看結果時，第一個請求會在您開始查看時執行，第二個請求在第二次查看時執行，以此類推。

從結果列舉資料

分頁程式物件有一個名為 search() 的方法，這可讓您為一組結果中的資料建立疊代運算。當您呼叫 search() 時，請提供 JMESPath 表達式以指定要擷取的資料。呼叫 search() 將傳回疊代運算，在各個結果頁面上產生表達式的結果。它會在您逐一查看傳回的疊代運算時，進行延遲的評估。

以下範例等同於上述程式碼範例，但使用 ResultPaginator::search() 方法更簡潔。

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'amzn-s3-demo-bucket'
]);

foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

JMESPath 表達式可讓您執行相當複雜的作業。例如，如果您要列印所有物件金鑰和常見前綴 (例如，執行儲存貯體的 `ls`)，您可以執行以下動作。

```
// List all prefixes ("directories") and objects ("files") in the bucket
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket'    => 'amzn-s3-demo-bucket',
    'Delimiter' => '/'
]);

$expression = '[CommonPrefixes[].Prefix, Contents[].Key][]';
foreach ($results->search($expression) as $item) {
    echo $item . "\n";
}
```

非同步分頁

您可以提供 `each()` 的 `Aws\ResultPaginator` 方法的回呼，以異步方式逐一查看分頁程式的結果。它會針對分頁程式產生的每個值叫用回呼。

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'amzn-s3-demo-bucket'
]);

$promise = $results->each(function ($result) {
    echo 'Got ' . var_export($result, true) . "\n";
});
```

> **Note**
>
> 使用 `each()` 方法可讓您對 API 操作的結果進行分頁，同時以非同步方式傳送其他請求。

底層以 coroutine 為基礎的承諾將會產生來自回呼的非 null 傳回值。這表示您可以從回呼傳回承諾，此回呼必須先解決再能繼續逐一查看其餘項目，基本上就是合併其他承諾至逐一查看。回呼傳回的最後一個非 null 值，是履行承諾至任何下游承諾的結果。如果上個傳回值是一個承諾，該承諾的解析度是履行或拒絕下游承諾的結果。

```
// Delete all keys that end with "Foo"
$promise = $results->each(function ($result) use ($s3Client) {
    if (substr($result['Key'], -3) === 'Foo') {
        // Merge this promise into the iterator
        return $s3Client->deleteAsync([
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'Foo'
        ]);
    }
});

$promise
    ->then(function ($result) {
        // Result would be the last result to the deleteAsync operation
    })
    ->otherwise(function ($reason) {
        // Reason would be an exception that was encountered either in the
        // call to deleteAsync or calls performed while iterating
    });

// Forcing a synchronous wait will also wait on all of the deleteAsync calls
$promise->wait();
```

第 3 適用於 PHP 的 AWS SDK 版中的等待程式

等待程式提供抽象化方法來等待，直到資源透過輪詢資源進入特定狀態，藉此更便於實現最終一致的系統。您可以檢視單一版本服務用戶端的 API 文件，來尋找用戶端支援的等待程式清單。若要前往該處，請前往 API 文件中的用戶端頁面，並導覽至特定版本號碼（以日期表示），然後向下捲動至「等待」區段。此連結會將您帶到 S3 的等待程式區段。

在下列範例中，Amazon S3 用戶端用於建立儲存貯體。然後將使用等待程式來等待，直到儲存貯體存在。

```
// Create a bucket
$s3Client->createBucket(['Bucket' => 'amzn-s3-demo-bucket']);
```

```
// Wait until the created bucket is available
$s3Client->waitUntil('BucketExists', ['Bucket' => 'amzn-s3-demo-bucket']);
```

如果等待程式需要輪詢儲存貯體太多次，它將丟出一個 `\RuntimeException` 例外狀況。

等待程式組態

等待程式由關聯的組態選項陣列所驅動。特定等待程式使用的所有選項都有預設值，但是可能遭到覆寫以支援不同的等待策略。

您可以傳遞 `@waiter` 選項的關聯陣列到用戶端 `waitUntil()` 和 `getWaiter()` 方法的 `$args` 引數，來修改等待程式組態選項。

```
// Providing custom waiter configuration options to a waiter
$s3Client->waitUntil('BucketExists', [
    'Bucket'  => 'amzn-s3-demo-bucket',
    '@waiter' => [
        'delay'       => 3,
        'maxAttempts' => 10
    ]
]);
```

延遲 (int)

輪詢嘗試間的延遲秒數。每個等待程式都有預設的 `delay` 組態值，但是您可能需要修改此設定來符合不同的使用案例。

maxAttempts (int)

在等待程式失敗前可發出的最高輪詢嘗試數量。此選項可確保您不會無限期等待資源。每個等待程式都有預設的 `maxAttempts` 組態值，但是您可能需要修改此設定來符合不同的使用案例。

initDelay (int)

在第一個輪詢嘗試前的等待時間 (以秒為單位)。在等待已知會需要時間進入期望狀態的資源時，這非常有幫助。

之前 (可呼叫)

每次嘗試前叫用的 PHP 可呼叫功能。可呼叫操作使用將執行的 `Aws\CommandInterface` 命令以及目前已執行的嘗試數量來叫用。`before` 可呼叫操作的使用可能會在執行前或提供進度資訊前修改命令。

```
use Aws\CommandInterface;
```

```
$s3Client->waitUntil('BucketExists', [
    'Bucket'  => 'amzn-s3-demo-bucket',
    '@waiter' => [
        'before' => function (CommandInterface $command, $attempts) {
            printf(
                "About to send %s. Attempt %d\n",
                $command->getName(),
                $attempts
            );
        }
    ]
]);
```

非同步等待

除了同步等待外，您可以叫用等待程式以在傳送其他請求或等待一次有多個資源時以非同步方式等待。

您可以使用用戶端的 `getWaiter($name, array $args = [])` 方法從用戶端擷取等待程式，以存取等待程式 promise。使用等待程式的 `promise()` 方法來啟動等待程式。等待程式承諾將以最後在等待程式中執行的 `Aws\CommandInterface` 來履行，並在發生錯誤時以 `RuntimeException` 拒絕。

```
use Aws\CommandInterface;

$waiterName = 'BucketExists';
$waiterOptions = ['Bucket' => 'amzn-s3-demo-bucket'];

// Create a waiter promise
$waiter = $s3Client->getWaiter($waiterName, $waiterOptions);

// Initiate the waiter and retrieve a promise
$promise = $waiter->promise();

// Call methods when the promise is resolved.
$promise
    ->then(function () {
        echo "Waiter completed\n";
    })
    ->otherwise(function (\Exception $e) {
        echo "Waiter failed: " . $e . "\n";
    });
```

```
// Block until the waiter completes or fails. Note that this might throw
// a \RuntimeException if the waiter fails.
$promise->wait();
```

公開以承諾為基礎的等待程式 API 可產生一些強大且相對低成本的使用案例。例如，如果您想在多個資源上等待，並且使用第一個成功解決的等待程式來執行一些動作時該怎麼做？

```
use Aws\CommandInterface;

// Create an array of waiter promises
$promises = [
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'a'])->promise(),
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'b'])->promise(),
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'c'])->promise()
];

// Initiate a race between the waiters, fulfilling the promise with the
// first waiter to complete (or the first bucket to become available)
$any = Promise\any($promises)
    ->then(function (CommandInterface $command) {
        // This is invoked with the command that succeeded in polling the
        // resource. Here we can know which bucket won the race.
        echo "The {$command['Bucket']} waiter completed first!\n";
    });

// Force the promise to complete
$any->wait();
```

第 3 適用於 PHP 的 AWS SDK 版中的 JMESPath 表達式

JMESPath 可讓您宣告式指定如何從 JSON 文件擷取元素。對 jmespath.php 適用於 PHP 的 AWS SDK 具有相依性，可支援第 3 版中的 適用於 PHP 的 AWS SDK Paginator 和第 3 適用於 PHP 的 AWS SDK 版中的 Waiters 等高階抽象概念，但也會公開 JMESPath 搜尋 `Aws\ResultInterface`和 `Aws\ResultPaginator`。

您可以透過瀏覽器試用線上 JMESPath 範例對 JMESPath 進行研究。您可以從 JMESPath 規格中進一步了解此語言，包括可用的表達式和函數。

AWS CLI 支援 JMESPath。您為 CLI 輸出編寫的表達式與為 適用於 PHP 的 AWS SDK編寫的表達式 100% 相容。

從結果擷取資料

此 `Aws\ResultInterface` 界面擁有 `search($expression)` 方法，可從根據 JMESPath 表達式的結果模型擷取資料。使用 JMESPath 表達式查詢結果物件中的資料可以幫助刪除範本條件程式碼，並且更加簡潔地表達正在擷取的資料。

為了示範其運作方式，我們將從以下的預設 JSON 輸出開始，其中說明兩個連接至個別 Amazon EC2 執行個體的 Amazon Elastic Block Store (Amazon EBS) 磁碟區。

```
$result = $ec2Client->describeVolumes();
// Output the result data as JSON (just so we can clearly visualize it)
echo json_encode($result->toArray(), JSON_PRETTY_PRINT);
```

```
{
    "Volumes": [
        {
            "AvailabilityZone": "us-west-2a",
            "Attachments": [
                {
                    "AttachTime": "2013-09-17T00:55:03.000Z",
                    "InstanceId": "i-a071c394",
                    "VolumeId": "vol-e11a5288",
                    "State": "attached",
                    "DeleteOnTermination": true,
                    "Device": "/dev/sda1"
                }
            ],
            "VolumeType": "standard",
            "VolumeId": "vol-e11a5288",
            "State": "in-use",
            "SnapshotId": "snap-f23ec1c8",
            "CreateTime": "2013-09-17T00:55:03.000Z",
            "Size": 30
        },
        {
            "AvailabilityZone": "us-west-2a",
            "Attachments": [
                {
                    "AttachTime": "2013-09-18T20:26:16.000Z",
                    "InstanceId": "i-4b41a37c",
                    "VolumeId": "vol-2e410a47",
                    "State": "attached",
```

```
                    "DeleteOnTermination": true,
                    "Device": "/dev/sda1"
                }
            ],
            "VolumeType": "standard",
            "VolumeId": "vol-2e410a47",
            "State": "in-use",
            "SnapshotId": "snap-708e8348",
            "CreateTime": "2013-09-18T20:26:15.000Z",
            "Size": 8
        }
    ],
    "@metadata": {
        "statusCode": 200,
        "effectiveUri": "https:\/\/ec2.us-west-2.amazonaws.com",
        "headers": {
            "content-type": "text\/xml;charset=UTF-8",
            "transfer-encoding": "chunked",
            "vary": "Accept-Encoding",
            "date": "Wed, 06 May 2015 18:01:14 GMT",
            "server": "AmazonEC2"
        }
    }
}
```

首先，我們可以使用下列命令，讓磁碟區清單只擷取第一個磁碟區。

```
$firstVolume = $result->search('Volumes[0]');
```

現在，我們使用 `wildcard-index` 表達式 `[*]` 在整個清單重複使用，也可以擷取並重新命名三個元素：`VolumeId` 已重新命名為 `ID`、`AvailabilityZone` 重新命名為 `AZ` 而 `Size` 則保持為 `Size`。我們可以使用放置於 `multi-hash` 表達式後的 `wildcard-index` 表達式擷取和重新命名這些元素。

```
$data = $result->search('Volumes[*].{ID: VolumeId, AZ: AvailabilityZone, Size: Size}');
```

這給我們如下所示的一系列 PHP 資料：

```
array(2) {
  [0] =>
  array(3) {
    'AZ' =>
    string(10) "us-west-2a"
```

```
    'ID' =>
    string(12) "vol-e11a5288"
    'Size' =>
    int(30)
  }
  [1] =>
  array(3) {
    'AZ' =>
    string(10) "us-west-2a"
    'ID' =>
    string(12) "vol-2e410a47"
    'Size' =>
    int(8)
  }
}
```

在 multi-hash 表示法，您也可以使用鏈結金鑰如 key1.key2[0].key3 來擷取出深度嵌套在結構中的元素。下列範例以 Attachments[0].InstanceId 金鑰做示範，別名簡稱為 InstanceId。(在大多數情況下，JMESPath 表達式會忽略空格。)

```
$expr = 'Volumes[*].{ID: VolumeId,
                     InstanceId: Attachments[0].InstanceId,
                     AZ: AvailabilityZone,
                     Size: Size}';

$data = $result->search($expr);
var_dump($data);
```

上述表達式會輸出以下資料：

```
array(2) {
  [0] =>
  array(4) {
    'ID' =>
    string(12) "vol-e11a5288"
    'InstanceId' =>
    string(10) "i-a071c394"
    'AZ' =>
    string(10) "us-west-2a"
    'Size' =>
    int(30)
  }
```

```
  [1] =>
  array(4) {
    'ID' =>
    string(12) "vol-2e410a47"
    'InstanceId' =>
    string(10) "i-4b41a37c"
    'AZ' =>
    string(10) "us-west-2a"
    'Size' =>
    int(8)
  }
}
```

您也可以使用 `multi-list` 表達式：`[key1, key2]` 來篩選多個元素。這樣做會針對每個物件將篩選出的所有屬性格式化為單一排序列表，而不管類型如何。

```
$expr = 'Volumes[*].[VolumeId, Attachments[0].InstanceId, AvailabilityZone, Size]';
$data = $result->search($expr);
var_dump($data);
```

執行之前的搜尋會產生以下資料：

```
array(2) {
  [0] =>
  array(4) {
    [0] =>
    string(12) "vol-e11a5288"
    [1] =>
    string(10) "i-a071c394"
    [2] =>
    string(10) "us-west-2a"
    [3] =>
    int(30)
  }
  [1] =>
  array(4) {
    [0] =>
    string(12) "vol-2e410a47"
    [1] =>
    string(10) "i-4b41a37c"
    [2] =>
    string(10) "us-west-2a"
    [3] =>
```

```
    int(8)
  }
}
```

使用 `filter` 表達式以特定欄位值篩選結果。下列範例查詢只會輸出 `us-west-2a` 可用區域內的磁碟區。

```
$data = $result->search("Volumes[?AvailabilityZone ## 'us-west-2a']");
```

JMESPath 也支援函數表達式。假設您想要執行與上述相同的查詢，但改為擷取磁碟區所在的 AWS 區域以「us-」開頭的所有磁碟區。以下表達式使用 `starts_with` 函數，以 `us-` 的字串常值來傳遞。這個函數的結果接著會與 `true` 的 JSON 常值進行比較，只傳遞透過篩選條件投射傳回的 `true` 篩選條件述詞結果。

```
$data = $result->search('Volumes[?starts_with(AvailabilityZone, 'us-') ## `true`]');
```

從分頁程式擷取資料

如同您在第 3 版指南中的 適用於 PHP 的 AWS SDK 分頁程式所知道，`Aws\ResultPaginator`物件用於從可分頁的 API 操作產生結果。 適用於 PHP 的 AWS SDK 可讓您從`Aws\ResultPaginator`物件中擷取篩選的資料並反覆運算，基本上，在 JMESPath 表達式的結果為映射函數的迭代器上實作平面映射。

假設您想要建立一個只從大於 1 MB 儲存貯體產生物件的 `iterator`。這可透過首先建立一個 `ListObjects` 分頁程式，然後將一個 `search()` 函數套用於分頁程式，在分頁資料上建立一個平面映射的疊代運算。

```
$result = $s3Client->getPaginator('ListObjects', ['Bucket' => 't1234']);
$filtered = $result->search('Contents[?Size > `1048576`]');

// The result yielded as $data will be each individual match from
// Contents in which the Size attribute is > 1048576
foreach ($filtered as $data) {
    var_dump($data);
}
```

AWS 服務 從第 3 適用於 PHP 的 AWS SDK 版呼叫

下列各節包含範例、教學課程、任務和指南，說明如何使用 適用於 PHP 的 AWS SDK 來使用 AWS 服務。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版的功能和選項
- 具有 指引的程式碼範例 適用於 PHP 的 AWS SDK

使用第 3 適用於 PHP 的 AWS SDK 版的功能和選項

第 3 適用於 PHP 的 AWS SDK 版支援使用 AWS 服務 APIs的其他功能和選項。本主題中的各節涵蓋這些服務的這些選項。

主題

- 搭配第 3 適用於 PHP 的 AWS SDK 版使用 DynamoDB 工作階段處理常式
- 第 3 版中的 Amazon S3 功能和選項 適用於 PHP 的 AWS SDK

搭配第 3 適用於 PHP 的 AWS SDK 版使用 DynamoDB 工作階段處理常式

DynamoDB 工作階段處理常式是 PHP 的自訂工作階段處理常式，可讓開發人員使用 Amazon DynamoDB 做為工作階段存放區。使用 DynamoDB 儲存工作階段，透過將工作階段移出本機檔案系統並移至共用位置，來緩解分散式 Web 應用程式中工作階段處理時所發生的問題。DynamoDB 快速、可擴展、易於設定並自動處理資料的複寫。

DynamoDB 工作階段處理常式使用 `session_set_save_handler()`函數將 DynamoDB 操作掛鉤至 PHP 的原生工作階段函數，以允許真正捨棄取代。這包括功能的支援 (例如，工作階段鎖定和廢棄項目收集等功能)，這是 PHP 預設工作階段處理器的一部分。

如需 DynamoDB 服務的詳細資訊，請參閱 Amazon DynamoDB 首頁。

基本使用

步驟 1：註冊處理常式

首先，將處理常式執行個體化，並進行登錄。

```
use Aws\DynamoDb\SessionHandler;
```

```
$dynamoDb = new Aws\DynamoDb\DynamoDbClient([
    'region'=>'us-east-1'  // Since version 3.277.10 of the SDK,
]);                        // the 'version' parameter defaults to 'latest'.

$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name' => 'sessions'
]);

$sessionHandler->register();
```

步驟 2. 建立資料表以存放您的工作階段

您必須先建立資料表來儲存工作階段，然後才能實際地使用工作階段處理常式。您可以使用適用於 AWS Amazon DynamoDB 的主控台，或使用 來提前執行此操作 適用於 PHP 的 AWS SDK。

建立此表格時，請使用 'id' 做為主索引鍵的名稱。同時建議您使用 'expires' 屬性來設定存活期屬性，以受益於工作階段的自動廢棄項目集合。

步驟 3。像平常一樣使用 PHP 工作階段

工作階段處理器註冊完畢後，若資料表存在，您可以使用 `$_SESSION` 超全域 (superglobal) 寫入和讀取該工作階段，其方式與您一般對 PHP 預設工作階段處理器執行的作業相同。DynamoDB 工作階段處理常式會封裝和抽象與 DynamoDB 的互動，並可讓您直接使用 PHP 的原生工作階段函數和界面。

```
// Start the session
session_start();

// Alter the session data
$_SESSION['user.name'] = 'jeremy';
$_SESSION['user.role'] = 'admin';

// Close the session (optional, but recommended)
session_write_close();
```

組態

您可以使用下列選項來設定工作階段處理常式的動作。所有選項皆為選用，但務必了解選項的預設值。

`table_name`

要存放工作階段的 DynamoDB 資料表名稱。預設值為 `'sessions'`。

hash_key

DynamoDB 工作階段資料表中的雜湊索引鍵名稱。預設值為 `'id'`。

data_attribute

DynamoDB 工作階段資料表中存放工作階段資料的屬性名稱。預設值為 `'data'`。

data_attribute_type

DynamoDB 工作階段資料表中存放工作階段資料的屬性類型。此值預設為 `'string'`，但可以選擇性地設定為 `'binary'`。

session_lifetime

在應該進行垃圾收集之前，非作用中工作階段的存留期。若未提供此值，將會使用的實際生命週期為 `ini_get('session.gc_maxlifetime')`。

session_lifetime_attribute

DynamoDB 工作階段資料表中存放工作階段過期時間的屬性名稱。預設值為 `'expires'`。

consistent_read

工作階段處理常式是否應針對 `GetItem` 操作，使用一致性讀取。預設值為 `true`。

locking

是否使用工作階段鎖定。預設值為 `false`。

batch_config

用來在收集垃圾時進行批次刪除的設定。這些選項會直接傳入 DynamoDB WriteRequestBatch 物件。透過 `SessionHandler::garbageCollect()`，以手動方式觸發廢棄項目收集作業。

max_lock_wait_time

在放棄之前，工作階段處理常式應等待以取得鎖定的時間長度上限 (以秒為單位)。預設值為 `10`，而且只用於工作階段鎖定。

min_lock_retry_microtime

工作階段處理常式應等待以嘗試取得鎖定的最短間隔時間 (以微秒為單位)。預設值為 `10000`，而且只用於工作階段鎖定。

max_lock_retry_microtime

工作階段處理常式應等待以嘗試取得鎖定的最長間隔時間 (以微秒為單位)。預設值為 `50000`，而且只用於工作階段鎖定。

若要設定工作階段處理常式，在進行處理常式的執行個體化時，指定設定選項。下列的程式碼提供了範例，其中指定了所有的設定選項。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'                    => 'sessions',
    'hash_key'                      => 'id',
    'data_attribute'                => 'data',
    'data_attribute_type'           => 'string',
    'session_lifetime'              => 3600,
    'session_lifetime_attribute'    => 'expires',
    'consistent_read'               => true,
    'locking'                       => false,
    'batch_config'                  => [],
    'max_lock_wait_time'            => 10,
    'min_lock_retry_microtime'      => 5000,
    'max_lock_retry_microtime'      => 50000,
]);
```

定價

除了資料儲存和資料傳輸費用之外，使用 DynamoDB 的相關費用是根據資料表的佈建輸送量來計算（請參閱 Amazon DynamoDB 定價詳細資訊)。傳輸量是以寫入容量和讀取容量的單位來測量。Amazon DynamoDB 首頁顯示：

讀取容量的單位，代表針對大小為 4 KB 的項目，每秒一次強式一致性讀取 (或是每秒 2 次最終一致讀取)。寫入容量的單位，代表針對大小為 1 KB 的項目，每秒一次寫入。

最後，您的工作階段資料表所需的傳輸量和成本，將會與您所預期的傳輸流量和工作階段大小相關。下表說明每個工作階段函數在 DynamoDB 資料表上執行的讀取和寫入操作量。

透過 `session_start()` 讀取	• 1 項讀取操作 (如果 `consistent_read` 為 `false`，則只有 0.5)。 • (條件式) 1 項寫入操作，以在工作階段過期時，將工作階段刪除。
透過 `session_start()` 讀取 (使用工作階段鎖定)	• 至少 1 項寫入操作。 • (條件式) 每次嘗試取得工作階段的鎖定時，進行額外的寫入操作。根據設定的鎖定等待時間和重試選項。

	• (條件式) 1 項寫入操作，以在工作階段過期時，將工作階段刪除。
透過 `session_write_close()` 寫入	• 1 項寫入操作。
透過 `session_destroy()` 刪除	• 1 項寫入操作。
垃圾收集	• 針對資料表中每 4 KB 的資料進行 0.5 次的讀取操作，以掃描過期的工作階段。 • 針對每個過期項目進行 1 次寫入操作，以刪除項目。

工作階段鎖定

DynamoDB 工作階段處理常式支援漸進式工作階段鎖定，以模擬 PHP 預設工作階段處理常式的行為。根據預設，DynamoDB 工作階段處理常式會關閉此功能，因為它可能會成為效能瓶頸並增加成本，尤其是當應用程式在使用 Ajax 請求或 iframe 時存取工作階段時。請先仔細考慮您的應用程式是否需要工作階段鎖定功能，然後再啟用。

若要啟用工作階段鎖定，請在您進行 `'locking'` 的執行個體化時，將 `true` 選項設定為 `SessionHandler`。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name' => 'sessions',
    'locking'    => true,
]);
```

垃圾回收

使用 ‘expires’ 屬性在 DynamoDB 資料表中設定 TTL 屬性。這會自動對您的工作階段進行廢棄項目收集，您就不需自行手段執行。

或者，DynamoDB 工作階段處理常式支援使用一系列 `Scan`和 `BatchWriteItem`操作收集工作階段垃圾。由於 `Scan` 操作運作方式的本質，和需要找出所有過期的工作階段並加以刪除，垃圾收集程序可能會需要用到許多的佈建傳輸量。

因此，我們不支援自動化的垃圾收集。更理想的做法是將垃圾收集作業排定於離峰時間進行，如此當耗用的傳輸量突增時，就不會干擾到其他的應用程式。例如，您可以設定夜間的 cron 工作，來觸發指令碼，以執行垃圾收集作業。此指令碼需要執行類似下列的動作。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'   => 'sessions',
    'batch_config' => [
        'batch_size' => 25,
        'before' => function ($command) {
            echo "About to delete a batch of expired sessions.\n";
        }
    ]
]);

$sessionHandler->garbageCollect();
```

您也可以使用 `'before'` 中的 `'batch_config'` 選項，來讓垃圾收集程序所執行的 `BatchWriteItem` 操作產生延遲。這將增加垃圾回收完成所需的時間，但可協助您分散 DynamoDB 工作階段處理常式提出的請求，以協助您在垃圾回收期間保持接近或低於佈建的輸送容量。

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'   => 'sessions',
    'batch_config' => [
        'before' => function ($command) {
            $command['@http']['delay'] = 5000;
        }
    ]
]);

$sessionHandler->garbageCollect();
```

最佳實務

1. 在 AWS 地理位置最接近或與您應用程式伺服器位於相同區域的 區域中建立工作階段資料表。這可確保應用程式與 DynamoDB 資料庫之間的最低延遲。
2. 請謹慎選擇您的工作階段資料表的佈建輸送容量。請考量至您應用程式的預期傳輸流量，以及工作階段的預期大小。或者，為您的資料表使用「隨需」讀取/寫入容量模式。
3. 透過 AWS 管理主控台或使用 Amazon CloudWatch 監控消耗的輸送量，並視需要調整輸送量設定，以滿足應用程式的需求。
4. 讓您工作階段的大小保持小型 (最好不到 1 KB)。小型工作階段的效能更為理想，而且需要佈建的輸送容量也較低。
5. 除非您的應用程式要求，否則請勿使用工作階段鎖定。

6. 請透過 cron 任務或其他的排程機制，來進行廢棄項目收集的排程，而不要使用 PHP 內建的工作階段廢棄項目收集觸發，以在離峰時間執行。善用 `'batch_config'` 選項。

所需的 IAM 許可

若要使用 DynamoDB SessionHhandler，您設定的登入資料必須具有許可，才能使用您在上一個步驟中建立的 DynamoDB 資料表。下列 IAM 政策包含您需要的最低許可。若要使用此政策，請將資源值取代為您先前建立之資料表的 Amazon Resource Name (ARN)。如需建立和連接 IAM 政策的詳細資訊，請參閱《IAM 使用者指南》中的管理 IAM 政策。

第 3 版中的 Amazon S3 功能和選項 適用於 PHP 的 AWS SDK

本主題討論第 3 適用於 PHP 的 AWS SDK 版為使用 Amazon S3 所提供的其他功能和選項。

主題

- 第 3 版中的 Amazon S3 多區域用戶端 適用於 PHP 的 AWS SDK
- 第 3 版中的 Amazon S3 串流包裝函式 適用於 PHP 的 AWS SDK
- 第 3 版中的 Amazon S3 Transfer Manager 適用於 PHP 的 AWS SDK
- 第 3 版中的 Amazon S3 用戶端加密 適用於 PHP 的 AWS SDK
- 使用檢查總和保護資料完整性

第 3 版中的 Amazon S3 多區域用戶端 適用於 PHP 的 AWS SDK

第 3 適用於 PHP 的 AWS SDK 版提供一般多區域用戶端，可與任何 服務搭配使用。這可讓使用者提供`@region`輸入參數給任何命令，以指定 AWS 要傳送命令的區域。此外，開發套件為 Amazon S3 提供多區域用戶端，以智慧方式回應特定的 Amazon S3 錯誤，並相應地重新路由命令。這讓使用者能夠使用相同的用戶端來與多個區域溝通。對於 Amazon S3 串流包裝函式第 3 適用於 PHP 的 AWS SDK 版的使用者來說，此功能特別有用，其儲存貯體位於多個區域。

基本使用

無論使用標準 Amazon S3 S3 用戶端的基本使用模式都相同。命令層級的唯一用量差異是可以使用`@region`輸入參數指定 AWS 區域。

```
// Create a multi-region S3 client
$s3Client = (new \Aws\Sdk)->createMultiRegionS3(['version' => 'latest']);
```

```
// You can also use the client constructor
$s3Client = new \Aws\S3\S3MultiRegionClient([
    'version' => 'latest',
    // Any Region specified while creating the client will be used as the
    // default Region
    'region' => 'us-west-2',
]);

// Get the contents of a bucket
$objects = $s3Client->listObjects(['Bucket' => $bucketName]);

// If you would like to specify the Region to which to send a command, do so
// by providing an @region parameter
$objects = $s3Client->listObjects([
    'Bucket' => $bucketName,
    '@region' => 'eu-west-1',
]);
```

> **Important**
>
> 使用多區域 Amazon S3 用戶端時，您不會遇到任何永久重新導向例外狀況。當命令傳送到錯誤的區域`Aws\S3\Exception\PermanentRedirectException`時，標準 Amazon S3 用戶端會擲回 的執行個體。多區域用戶端將重新配發命令到正確區域。

儲存貯體區域快取

Amazon S3 多區域用戶端會維護指定儲存貯體所在 AWS 區域的內部快取。在預設情況下，每個用戶端都有自己的記憶體內快取。若要在用戶端或程序之間共用快取，請向您的多區域用戶端提供 `Aws\CacheInterface` 的執行個體做為 `bucket_region_cache` 選項。

```
use Aws\DoctrineCacheAdapter;
use Aws\Sdk;
use Doctrine\Common\Cache\ApcuCache;

$sdk = new Aws\Sdk([
    'version' => 'latest',
    'region' => 'us-west-2',
    'S3' => [
        'bucket_region_cache' => new DoctrineCacheAdapter(new ApcuCache),
    ],
```

```
]);
```

第 3 版中的 Amazon S3 串流包裝函式 適用於 PHP 的 AWS SDK

Amazon S3 串流包裝函式可讓您使用內建 PHP 函數，例如 `file_get_contents`、`fopen`、、`copy``rename`、`unlink`、 和 ，從 Amazon S3 存放`mkdir`和擷取資料`rmdir`。

您需要註冊 Amazon S3 串流包裝函式才能使用它。

```
$client = new Aws\S3\S3Client([/** options **/]);

// Register the stream wrapper from an S3Client object
$client->registerStreamWrapper();
```

這可讓您使用`s3://`通訊協定存取存放在 Amazon S3 中的儲存貯體和物件。Amazon S3 串流包裝函式接受字串，其中包含儲存貯體名稱，後面接著正斜線和選用的物件索引鍵或字首：`s3:// <bucket>[/<key-or-prefix>]`。

> **Note**
>
> 串流包裝函式是專為使用物件和儲存貯體而設計 (您在這些物件和儲存貯體上擁有最低限度的讀取權限)。這代表您的使用者應具有權限，可執行任何儲存貯體上的 `ListBucket`，以及使用者需要與其進行互動的任何物件上的 `GetObject`。對於您沒有此許可層級的使用案例，我們建議您直接使用 Amazon S3 用戶端操作。

下載資料

您可以使用 `file_get_contents` 來擷取物件的內容。不過，請謹慎使用；因為此函式會將物件的整個內容載入到記憶體。

```
// Download the body of the "key" object in the "bucket" bucket
$data = file_get_contents('s3://bucket/key');
```

使用較大的檔案`fopen()`時，或如果您需要從 Amazon S3 串流資料，請使用 。

```
// Open a stream in read-only mode
if ($stream = fopen('s3://bucket/key', 'r')) {
    // While the stream is still open
```

```
    while (!feof($stream)) {
        // Read 1,024 bytes from the stream
        echo fread($stream, 1024);
    }
    // Be sure to close the stream resource when you're done with it
    fclose($stream);
}
```

> **Note**
>
> 只有在呼叫 `fflush` 時，才會傳回檔案寫入錯誤。呼叫未排清的 `fclose` 時，不會傳回這些錯誤。如果 `fclose` 關閉了串流，則其傳回值將會是 `true`，無論是否為了回應其內部 `fflush` 而產生了任何的錯誤。根據 PHP 實作的方式，在呼叫 `file_put_contents` 時，也不會傳回這些錯誤。

開啟可搜尋的串流

以 ”r” 模式開啟的串流，只允許從串流讀取資料，而且根據預設無法搜尋。這樣資料才能以真正串流的方式從 Amazon S3 下載，其中先前讀取的位元組不需要緩衝到記憶體中。如果您需要使串流可供搜尋，可以將 `seekable` 傳入函式的串流細節內容選項。

```
$context = stream_context_create([
    's3' => ['seekable' => true]
]);

if ($stream = fopen('s3://bucket/key', 'r', false, $context)) {
    // Read bytes from the stream
    fread($stream, 1024);
    // Seek back to the beginning of the stream
    fseek($stream, 0);
    // Read the same bytes that were previously read
    fread($stream, 1024);
    fclose($stream);
}
```

開啟可搜尋的串流，能夠讓您搜尋先前已讀取的位元組內容。您無法往前跳到尚未從遠端伺服器讀取的位元組。若要能夠叫用先前讀取的資料，可使用串流修飾器，來將資料置於 PHP 臨時串流中以進行緩衝。當已快取的資料量超過 2 MB 時，臨時串流中的資料會從記憶體傳送到磁碟。使用串流內容設定從 Amazon S3 `seekable` 下載大型檔案時，請記住這一點。

上傳資料

您可以使用 將資料上傳至 Amazon S3`file_put_contents()`。

```
file_put_contents('s3://bucket/key', 'Hello!');
```

您可以在串流資料時，使用 `fopen()` 和 “w”、”x” 或 “a” 串流存取模式，來上傳較大型的檔案。Amazon S3 串流包裝函式不支援同時讀取和寫入串流（例如「r+」、「w+」等）。這是因為 HTTP 通訊協定不允許同時讀取和寫入。

```
$stream = fopen('s3://bucket/key', 'w');
fwrite($stream, 'Hello!');
fclose($stream);
```

> **Note**
>
> 在傳送請求的承載之前，Amazon S3 需要指定 Content-Length 標頭。因此，要在 `PutObject` 操作中上傳的資料，會利用 PHP 臨時串流來進行內部緩衝，直到串流已排清或關閉。

> **Note**
>
> 只有在呼叫 `fflush` 時，才會傳回檔案寫入錯誤。呼叫未排清的 `fclose` 時，不會傳回這些錯誤。如果 `fclose` 關閉了串流，則其傳回值將會是 `true`，無論是否為了回應其內部 `fflush` 而產生了任何的錯誤。根據 PHP 實作的方式，在呼叫 `file_put_contents` 時，也不會傳回這些錯誤。

fopen 模式

PHP 的 fopen() 函數需要指定 `$mode` 選項。此模式選項會指定資料是否可以讀取或寫入串流，以及開啟串流時該檔案是否必須存在。

對於以 Amazon S3 物件為目標的串流，Amazon S3 串流包裝函式支援下列模式。

r	物件必須已存在的唯讀串流。

w	僅限寫入的串流。如果物件已存在，則會予以覆寫。
a	僅限寫入的串流。如果物件已存在，則會將其下載至暫時串流，而任何對串流的寫入都會附加至任何先前上傳的資料。
x	僅限寫入的串流。如果物件已存在，就會引發錯誤。

其他物件函數

串流包裝函式允許許多不同的內建 PHP 函數使用自訂系統，例如 Amazon S3。以下是 Amazon S3 串流包裝函式可讓您使用存放在 Amazon S3 中的物件執行的一些函數。

unlink()	從儲存貯體刪除物件。
	`// Delete an object from a bucket`
	`unlink('s3://bucket/key');`
	您可以傳入 `DeleteObject` 操作可使用的任何選項，來修改刪除物件的方式 (例如，指定特定的物件版本)。
	`// Delete a specific version of an object from a bucket`
	`unlink('s3://bucket/key', stream_context_create([`
	`'s3' => ['VersionId' => '123']`
	`]);`

filesize()	取得物件的大小。
	`// Get the Content-Length of an object`
	`$size = filesize('s3://bucket/key', );`

is_file()	檢查 URL 是否為檔案。

	`if (is_file('s3://bucket/key')) {`
	`  echo 'It is a file!';`
	`}`

file_exists()	檢查物件是否存在。
	`if (file_exists('s3://bucket/key'))`
	` {`
	`  echo 'It exists!';`
	`}`

filetype()	檢查 URL 是否對應到檔案或儲存貯體 (dir)。
file()	透過行陣列來載入物件的內容。您可以傳入 `GetObject` 操作可使用的任何選項，來修改檔案下載的方式。
filemtime()	取得上次修改物件的日期。
rename()	藉由先複製物件然後再刪除原始物件，來將物件重新命名。您可以將 `CopyObject` 和 `DeleteObject` 操作可用的選項傳到串流細節內容參數，來修改複製和刪除物件的方式。

> **Note**
>
> 雖然 `copy`通常適用於 Amazon S3 串流包裝函式，但由於 PHP 中的`copy`函數內部，某些錯誤可能無法正確報告。建議您改用 AwsS3ObjectCopier 的執行個體。

使用儲存貯體和資料夾

使用 `mkdir()` 處理儲存貯體

您可以建立和瀏覽 Amazon S3 儲存貯體，類似於 PHP 允許您在檔案系統上建立和周遊目錄的方式。

以下是建立儲存貯體的範例。

```
mkdir('s3://amzn-s3-demo-bucket');
```

> **Note**
>
> 在 2023 年 4 月，Amazon S3 會自動為所有新建立的儲存貯體啟用 S3 封鎖公開存取和停用存取控制清單。此變更也會影響 `StreamWrappermkdir`函數搭配許可和 ACLs的運作方式。如需詳細資訊，請參閱此 AWS 最新消息文章。

您可將串流細節內容選項傳入 `mkdir()` 方法，以修改使用 CreateBucket 操作可用的參數建立儲存貯體的方式。

```
// Create a bucket in the EU (Ireland) Region
mkdir('s3://amzn-s3-demo-bucket', 0500, true,
        stream_context_create([
            's3' => ['LocationConstraint' => 'eu-west-1']
        ]));
```

您可以使用 `rmdir()` 函式來刪除儲存貯體。

```
// Delete a bucket
rmdir('s3://amzn-s3-demo-bucket);
```

> **Note**
>
> 只有空的儲存貯體才能刪除。

使用 **`mkdir()`** 處理資料夾

建立儲存貯體之後，您可以使用 `mkdir()` 來建立物件，其作用如同檔案系統中的資料夾。

下列程式碼片段會將名為 'my-folder' 的資料夾物件新增至名為 'amzn-s3-demo-bucket' 的現有儲存貯體。使用正斜線 (/) 字元，將資料夾物件名稱與儲存貯體名稱和任何其他資料夾名稱分開。

```
mkdir('s3://amzn-s3-demo-bucket/my-folder')
```

先前有關 2023 年 4 月之後許可變更的備註，也會在您建立資料夾物件時開始播放。此部落格文章提供如何視需要調整許可的相關資訊。

使用 `rmdir()`函數刪除空的資料夾物件，如下列程式碼片段所示。

```
rmdir('s3://amzn-s3-demo-bucket/my-folder')
```

列出儲存貯體的內容

您可以使用 opendir()、readdir()、rewinddir() 和 closedir() PHP 函數搭配 Amazon S3 串流包裝函式來周遊儲存貯體的內容。將 ListObjects 操作可用的參數傳入 `opendir()` 函數做為自訂串流細節內容選項，即可修改列出物件的方式。

```
$dir = "s3://bucket/";

if (is_dir($dir) && ($dh = opendir($dir))) {
    while (($file = readdir($dh)) !== false) {
        echo "filename: {$file} : filetype: " . filetype($dir . $file) . "\n";
    }
    closedir($dh);
}
```

您可以使用 PHP 的 RecursiveDirectoryIterator，用遞迴方式列出儲存貯體中的每個物件和字首。

```
$dir = 's3://bucket';
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));

foreach ($iterator as $file) {
    echo $file->getType() . ': ' . $file . "\n";
}
```

用遞迴方式列出儲存貯體內容的另一種方式，是使用 `Aws\recursive_dir_iterator($path, $context = null)` 函式 (這會減少 HTTP 請求的數量)。

```
<?php
require 'vendor/autoload.php';

$iter = Aws\recursive_dir_iterator('s3://bucket/key');
foreach ($iter as $filename) {
    echo $filename . "\n";
}
```

串流內容選項

您可以藉由傳入自訂的串流細節內容選項，來自訂串流包裝函式所使用的用戶端，或是用來快取先前已載入資訊 (關於儲存貯體和金鑰) 的快取。

串流包裝函式支援所有操作使用下列的串流細節內容選項。

client

用來執行命令的 `Aws\AwsClientInterface` 物件。

cache

`Aws\CacheInterface` 的執行個體，用來快取先前取得的檔案統計資料。在預設情況下，串流包裝函式會使用記憶體內的 LRU 快取。

第 3 版中的 Amazon S3 Transfer Manager 適用於 PHP 的 AWS SDK

中的 Amazon S3 傳輸管理員 適用於 PHP 的 AWS SDK 用於將整個目錄上傳到 Amazon S3 儲存貯體，並將整個儲存貯體下載到本機目錄。

將本機目錄上傳至 Amazon S3

`Aws\S3\Transfer` 物件是用來進行傳輸。下列範例示範如何以遞迴方式將檔案的本機目錄上傳至 Amazon S3 儲存貯體。

```
// Create an S3 client.
$client = new \Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'version' => '2006-03-01',
]);

// Where the files will be sourced from.
$source = '/path/to/source/files';

// Where the files will be transferred to.
$dest = 's3://bucket';

// Create a transfer object.
$manager = new \Aws\S3\Transfer($client, $source, $dest);

// Perform the transfer synchronously.
```

```
$manager->transfer();
```

在此範例中，我們建立 Amazon S3 用戶端、建立 `Transfer` 物件，並同步執行傳輸。使用之前的範例，來示範進行傳輸所需的最低限度程式碼數量。傳輸物件可以非同步方式進行傳輸，並具備各種組態選項，您可用來自訂傳輸。

您可以在 `s3://` URI 中提供金鑰字首，將本機檔案上傳至 Amazon S3 儲存貯體的「子資料夾」。下列的範例將磁碟上的本機檔案上傳到 `bucket` 儲存貯體，並儲存金鑰前綴為 `foo` 的儲存檔案。

```
$source = '/path/to/source/files';
$dest = 's3://bucket/foo';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

下載 Amazon S3 儲存貯體

您可以將`$source`引數指定為 Amazon S3 URI（例如 `s3://bucket`），並將`$dest`引數指定為本機目錄的路徑，以遞迴方式將 Amazon S3 儲存貯體下載至磁碟上的本機目錄。

```
// Where the files will be sourced from.
$source = 's3://bucket';

// Where the files will be transferred to.
$dest = '/path/to/destination/dir';

$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

> **Note**
>
> 在下載儲存貯體中的物件時，軟體開發套件會自動建立所有必要的目錄。

您可以在儲存貯體之後的 Amazon S3 URI 中包含金鑰字首，以僅下載存放在「虛擬資料夾」下的物件。以下範例只會下載指定儲存貯體以 ”/foo” 金鑰前綴儲存的檔案。

```
$source = 's3://bucket/foo';
$dest = '/path/to/destination/dir';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
```

```
$manager->transfer();
```

組態

`Transfer` 物件建構函式接受下列引數。

$client

用來執行傳輸的 `Aws\ClientInterface` 物件。

$source （字串 | **Iterator**)

正在傳輸的原始碼資料。這可以指向磁碟上的本機路徑 （例如 `/path/to/files`) 或 Amazon S3 儲存貯體 （例如 `s3://bucket`)。`s3://` URI 也可以包含金鑰前綴，用來只傳輸使用共同前綴的物件。

如果`$source`引數是 Amazon S3 URI，則`$dest`引數必須是本機目錄 （反之亦然）。

除了提供字串值，您也可以提供 `\Iterator` 物件，此物件會產生絕對的檔案名稱。如果提供了疊代器，則您必須在 `$options` 關聯陣列中提供 `base_dir` 選項。

$dest

檔案傳輸的目的地。如果`$source`引數是磁碟上的本機路徑， `$dest` 必須是 Amazon S3 儲存貯體 URI （例如 `s3://bucket`)。如果`$source`引數是 Amazon S3 儲存貯體 URI，則`$dest`引數必須是磁碟上的本機路徑。

$options

傳輸選項的關聯陣列。下列傳輸選項有效：

add_content_md5 (bool)

設定為 `true`以計算上傳的 MD5 檢查總和。

base_dir (string)

來源的基本目錄 (如果 `$source` 為疊代器)。如果 `$source` 選項不是陣列，則會略過此選項。

before (可呼叫)

在每次傳輸前呼叫的回呼函式。回呼函式應具有像是 `function (Aws\Command $command) {...}` 的函式簽章。提供的命令將會是 `GetObject`、`PutObject`、`CreateMultipartUpload`、`UploadPart` 或 `CompleteMultipartUpload` 命令。

mup_threshold (int)

分段上傳應使用的大小 (而非使用 `PutObject`)，單位為位元組。預設為 16777216 (16 MB)。

concurrency (整數，預設 = 5)

要同時上傳的檔案數量。理想的並行值，將會視待上傳檔案的數量，以及每個檔案的平均大小而有不同。一般而言，相較於大型檔案，較小的檔案在並行數值較高時具有較大的優勢。

debug (bool)

設定為 `true` 可印出傳輸的偵錯資訊。設定為 `fopen()` 資源，可寫入特定串流，而非寫入 STDOUT。

非同步傳輸

`Transfer` 物件是 `GuzzleHttp\Promise\PromisorInterface` 的執行個體。這表示傳輸可以非同步進行，也可透過呼叫物件的 `promise` 方法來起始。

```
$source = '/path/to/source/files';
$dest = 's3://bucket';
$manager = new \Aws\S3\Transfer($client, $source, $dest);

// Initiate the transfer and get a promise.
$promise = $manager->promise();

// Do something when the transfer is complete using the then() method.
$promise->then(function () {
    echo 'Done!';
});
```

如果有任何的檔案無法傳輸，promise 物件的操作將會遭到拒絕。您可以使用 promise 物件的 `otherwise` 方法，來以非同步方式處理失敗的傳輸。`otherwise` 函式可接受回呼，在發生錯誤時呼叫。回呼函式會接受拒絕的 `$reason`，這通常是 `Aws\Exception\AwsException` 的執行個體 (雖然任何類型的值皆可傳送到回呼函式)。

```
$promise->otherwise(function ($reason) {
    echo 'Transfer failed: ';
    var_dump($reason);
});
```

由於 `Transfer` 物件會傳回 promise 的值，這些傳輸可和其他的非同步 promise 物件同時執行。

自訂 Transfer Manager 的命令

透過將回呼函式傳遞至其建構函式，也可在傳輸管理程式執行操作時，設定自訂的選項。

```
$uploader = new Transfer($s3Client, $source, $dest, [
    'before' => function (\Aws\Command $command) {
        // Commands can vary for multipart uploads, so check which command
        // is being processed.
        if (in_array($command->getName(), ['PutObject', 'CreateMultipartUpload'])) {
            // Set custom cache-control metadata.
            $command['CacheControl'] = 'max-age=3600';
            // Apply a canned ACL.
            $command['ACL'] = strpos($command['Key'], 'CONFIDENTIAL') ### false
                ? 'public-read'
                : 'private';
        }
    },
]);
```

第 3 版中的 Amazon S3 用戶端加密 適用於 PHP 的 AWS SDK

使用用戶端加密時，會直接在您的環境中將資料加密和解密。這表示此資料會在傳輸到 Amazon S3 之前加密，而且您不需要依賴外部服務來為您處理加密。對於新的實作，我們建議您使用 `S3EncryptionClientV2`和 `S3EncryptionMultipartUploaderV2`來取代已棄用 `S3EncryptionClient`和 `S3EncryptionMultipartUploader`。建議仍然使用已棄用版本的舊實作嘗試遷移。 會`S3EncryptionClientV2`維持對使用舊版 加密資料的解密支援`S3EncryptionClient`。

適用於 PHP 的 AWS SDK 實作信封加密，並使用 OpenSSL 進行加密和解密。這項實作功能可與符合其功能支援的其他開發套件互通，也相容於軟體開發套件採用 promise 的非同步工作流程。

遷移指南

對於嘗試從已棄用用戶端遷移到 的新用戶端，您可以在這裡找到遷移指南。

設定

若要開始使用用戶端加密，您需要下列項目：

- AWS KMS 加密金鑰
- S3 儲存貯體

在執行任何範例程式碼之前，請設定您的 AWS 登入資料。請參閱第 3 適用於 PHP 的 AWS SDK 版的登入資料。

加密

在 中上傳加密的物件`S3EncryptionClientV2`，除了標準參數之外，還需要三個額外的`PutObject`參數：

- `'@KmsEncryptionContext'` 是金鑰/值對，可用於為您的加密物件新增額外的安全層。加密用戶端必須傳入相同的金鑰，該金鑰會在取得呼叫時自動執行。如果不需要其他內容，請傳入空白陣列。
- `@CipherOptions` 是加密的其他組態，包括要使用哪個密碼和 keysize。
- `@MaterialsProvider` 是處理產生加密金鑰和初始化向量，以及加密加密金鑰的提供者。

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProviderV2;

 // Let's construct our S3EncryptionClient using an S3Client
 $encryptionClient = new S3EncryptionClientV2(
     new S3Client([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ])
 );

 $kmsKeyId = 'kms-key-id';
 $materialsProvider = new KmsMaterialsProviderV2(
     new KmsClient([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ]),
     $kmsKeyId
 );

 $bucket = 'the-bucket-name';
 $key = 'the-file-name';
 $cipherOptions = [
     'Cipher' => 'gcm',
```

```
    'KeySize' => 256,
    // Additional configuration options
];

$result = $encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

> **Note**
>
> 除了 Amazon S3 和 AWS KMS型服務錯誤之外，如果`'@CipherOptions'`未正確設定 ，您可能會收到擲出的`InvalidArgumentException`物件。

解密

除了標準參數之外，下載和解密物件還有四個額外的`GetObject`參數，其中兩個是必要的。用戶端會為您偵測基本密碼選項。

- **`'@SecurityProfile'`**：如果設定為「V2」，則僅限在 V2-compatible中加密的物件

  格式可以解密。將此參數設為「V2_AND_LEGACY」也允許解密以 V1-compatible格式加密的物件。若要支援遷移，請將 @SecurityProfile 設定為「V2_AND_LEGACY」。僅使用「V2」進行新的應用程式開發。

- **`'@MaterialsProvider'`** 是處理產生加密金鑰和初始化向量的供應商，

  以及加密您的密碼金鑰。

- **`'@KmsAllowDecryptWithAnyCmk'`**：（選用）將此參數設為 true 可啟用解密

  但不提供 KMS 金鑰 ID 給 MaterialsProvider 的建構函式。預設值為 false。

- **`'@CipherOptions'`**（選用）是加密的其他組態，包括

  要使用的 密碼和 keysize。

```
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

> **Note**
>
> 除了 Amazon S3 和 AWS KMS型服務錯誤之外，如果`'@CipherOptions'`未正確設定 ，您可能會收到擲出的`InvalidArgumentException`物件。

密碼組態

`'Cipher'` (string)

加密用戶端在加密時所使用的加密方法。目前僅支援「gcm」。

> **Important**
>
> PHP 的 7.1 版已更新，納入了額外的必要參數，以使用 OpenSSL 進行 GCM 加密的加密和解密動作。對於 PHP 7.0 版和更早版本，加密用戶端和 `S3EncryptionClientV2` 會提供並使用 GCM 支援的 polyfill`S3EncryptionMultipartUploaderV2`。不過，使用 polyfill 對大型輸入的效能會比使用 PHP 7.1+ 的原生實作慢得多，因此可能需要升級較舊的 PHP 版本環境才能有效使用這些環境。

`'KeySize'` (int)

進行加密所需產生的內容加密金鑰的長度。預設值為 256 位元。有效的組態選項為 256 和 128 位元。

`'Aad'` (string)

要選用地包含在您加密承載中的「其他身分驗證資料」。解密時會驗證這項資訊。`Aad` 僅在使用「gcm」加密法時才能使用。

> **Important**
>
> AWS SDKs 不支援其他身分驗證資料，因此其他SDKs可能無法解密使用此參數加密的檔案。

中繼資料策略

您也可以選擇針對實作 `Aws\Crypto\MetadataStrategyInterface` 的類別，來提供其執行個體。這個簡單的界面可儲存和載入 `Aws\Crypto\MetadataEnvelope`，其中包含您的信封加密資料。開發套件提供了實作此項目的兩個類別：`Aws\S3\Crypto\HeadersMetadataStrategy` 與 `Aws\S3\Crypto\InstructionFileMetadataStrategy`。預設會使用 `HeadersMetadataStrategy`。

```
$strategy = new InstructionFileMetadataStrategy(
    $s3Client
);

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => $strategy,
    '@KmsEncryptionContext' => [],
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => false,
    '@MaterialsProvider' => $materialsProvider,
    '@SecurityProfile' => 'V2',
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

您也可透過呼叫 `HeadersMetadataStrategy::class`和`InstructionFileMetadataStrategy::class`，來提供 和 的類別名稱常數。

```
$result = $encryptionClient->putObject([
```

```
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => HeadersMetadataStrategy::class,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

> **Note**
>
> 如果在上傳指令檔案後發生故障，不會自動刪除該檔案。

分段上傳

您也可以使用用戶端加密來進行分段上傳。會在上傳之前`Aws\S3\Crypto\S3EncryptionMultipartUploaderV2`準備來源串流以進行加密。建立的方法類似於使用 `Aws\S3\MultipartUploader` 和 `Aws\S3\Crypto\S3EncryptionClientV2` 時的經驗。`S3EncryptionMultipartUploaderV2` 可以處理與 `'@MetadataStrategy'` 相同的 `S3EncryptionClientV2` 選項，以及所有可用的 `'@CipherOptions'` 組態。

```
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-upload-key';
$cipherOptions = [
    'Cipher' => 'gcm'
    'KeySize' => 256,
    // Additional configuration options
];

$multipartUploader = new S3EncryptionMultipartUploaderV2(
    new S3Client([
        'region' => 'us-east-1',
```

```
        'version' => 'latest',
        'profile' => 'default',
    ]),
    fopen('large-file-to-encrypt.txt', 'r'),
    [
        '@MaterialsProvider' => $materialsProvider,
        '@CipherOptions' => $cipherOptions,
        'bucket' => $bucket,
        'key' => $key,
    ]
);
$multipartUploader->upload();
```

> **Note**
>
> 除了 Amazon S3 和 AWS KMS型服務錯誤之外，如果`'@CipherOptions'`未正確設定 ，您可能會收到擲出的`InvalidArgumentException`物件。

使用檢查總和保護資料完整性

Amazon Simple Storage Service (Amazon S3) 可讓您在上傳物件時指定檢查總和。當您指定檢查總和時，它會與 物件一起存放，並在下載物件時驗證。

當您傳輸檔案時，檢查總和可提供多一層的資料完整性。使用檢查總和，您可以透過確認收到的檔案符合原始檔案來驗證資料一致性。如需使用 Amazon S3 檢查總和的詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南，包括支援的演算法。

您可以靈活地選擇最符合您需求的演算法，並讓 SDK 計算檢查總和。或者，您可以使用其中一個支援的演算法來提供預先計算的檢查總和值。

> **Note**
>
> 軟體開發套件也提供全域設定，用於外部設定的資料完整性保護，您可以在軟體AWS SDKs和工具參考指南中閱讀這些保護。

我們討論兩個請求階段的檢查總和：上傳物件和下載物件。

上傳物件

如果您未隨請求提供檢查總和演算法，檢查總和行為會根據您使用的 SDK 版本而有所不同，如下表所示。

未提供檢查總和演算法時的檢查總和行為

使用預先計算的檢查總和值

隨請求提供的預先計算檢查總和值會停用 SDK 的自動運算，並改用提供的值。

下列範例顯示具有預先計算 SHA256 檢查總和的請求。

如果 Amazon S3 判斷指定演算法的檢查總和值不正確，則服務會傳回錯誤回應。

分段上傳

您也可以搭配分段上傳使用檢查總和。

下載物件

當您使用 getObject方法下載物件時，開發套件會在

以下程式碼片段中的請求會指示 SDK 透過計算檢查總和並比較值來驗證回應中的檢查總和。

> **Note**
>
> 如果物件未使用檢查總和上傳，則不會進行驗證。

具有 指引的程式碼範例 適用於 PHP 的 AWS SDK

本節包含程式碼範例，示範使用 的常見 AWS 案例 適用於 PHP 的 AWS SDK。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon CloudFront 範例
- 使用第 3 適用於 PHP 的 AWS SDK 版簽署自訂 Amazon CloudSearch 網域請求

- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon CloudWatch 範例
- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon EC2 範例
- 使用第 3 適用於 PHP 的 AWS SDK 版簽署 Amazon OpenSearch Service 搜尋請求
- AWS Identity and Access Management 使用第 3 適用於 PHP 的 AWS SDK 版的範例
- AWS Key Management Service 使用第 3 適用於 PHP 的 AWS SDK 版的範例
- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon Kinesis 範例
- AWS Elemental MediaConvert 使用第 3 適用於 PHP 的 AWS SDK 版的範例
- 使用第 3 版的 Amazon S3 範例 適用於 PHP 的 AWS SDK
- 使用 Secrets Manager API 和 第 3 適用於 PHP 的 AWS SDK 版管理秘密
- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon SES 範例
- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon SNS 範例
- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon SQS 範例
- 使用第 3 適用於 PHP 的 AWS SDK 版將事件傳送至 Amazon EventBridge 全域端點

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon CloudFront 範例

Amazon CloudFront 是一種 AWS Web 服務，可加速從您自己的 Web 伺服器或 AWS 伺服器提供靜態和動態 Web 內容，例如 Amazon S3。CloudFront 透過稱為節點的資料中心全球網路提供內容。當使用者請求您使用 CloudFront 分發的內容時，它們會路由到提供最低延遲的節點。如果內容尚未快取，CloudFront 會從原始伺服器擷取複本、提供複本，然後快取以供未來請求使用。

如需 CloudFront 的詳細資訊，請參閱《Amazon CloudFront 開發人員指南》。

第 3 適用於 PHP 的 AWS SDK 版的所有範例程式碼都可在 GitHub 上取得。

使用 Amazon CloudFront CloudFront 分佈 適用於 PHP 的 AWS SDK

Amazon CloudFront 會在全球節點快取內容，以加速靜態和動態檔案的分佈，而這些檔案會存放在您自己的伺服器上，或在 Amazon S3 和 Amazon EC2 等 Amazon 服務上。當使用者從您的網站請求內容時，如果檔案快取到該處，CloudFront 會從最近的節點提供內容。否則CloudFront 會擷取檔案的副本、提供該檔案，然後快取該檔案以供下一個請求使用。在節點上快取內容會降低在該區域類似使用者請求的延遲。

對於您建立的每個 CloudFront 分發，您可以指定內容所在的位置，以及在使用者提出請求時如何分發內容。這個主題著重於靜態和動態檔案 (例如 HTML、CSS、JSON 和映像檔案) 的分佈。F 或有關隨需視訊使用 CloudFront 的資訊，請參閱隨需和 CloudFront 即時串流視訊。

下列範例示範如何：

- 使用 CreateDistribution 建立分佈。
- 使用 GetDistribution 取得分佈。
- 使用 ListDistributions 列出分佈。
- 使用 UpdateDistributions 更新分佈。
- 使用 DisableDistribution 停用分佈。
- 使用 DeleteDistributions 刪除分佈。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon CloudFront 的詳細資訊，請參閱 Amazon CloudFront 開發人員指南。

建立 CloudFront 分佈

從 Amazon S3 儲存貯體建立分佈。在以下範例中，已將可選參數標示為註解，但會顯示預設值。若要將自訂新增到分佈，請同時取消 `$distribution` 中值和參數的註解。

若要建立 CloudFront 分佈，請使用 CreateDistribution 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function createS3Distribution($cloudFrontClient, $distribution)
{
    try {
        $result = $cloudFrontClient->createDistribution([
            'DistributionConfig' => $distribution
        ]);
```

```
        $message = '';

        if (isset($result['Distribution']['Id'])) {
            $message = 'Distribution created with the ID of ' .
                $result['Distribution']['Id'];
        }

        $message .= ' and an effective URI of ' .
            $result['@metadata']['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function createsTheS3Distribution()
{
    $originName = 'my-unique-origin-name';
    $s3BucketURL = 'my-bucket-name.s3.amazonaws.com';
    $callerReference = 'my-unique-caller-reference';
    $comment = 'my-comment-about-this-distribution';
    $defaultCacheBehavior = [
        'AllowedMethods' => [
            'CachedMethods' => [
                'Items' => ['HEAD', 'GET'],
                'Quantity' => 2
            ],
            'Items' => ['HEAD', 'GET'],
            'Quantity' => 2
        ],
        'Compress' => false,
        'DefaultTTL' => 0,
        'FieldLevelEncryptionId' => '',
        'ForwardedValues' => [
            'Cookies' => [
                'Forward' => 'none'
            ],
            'Headers' => [
                'Quantity' => 0
            ],
            'QueryString' => false,
            'QueryStringCacheKeys' => [
                'Quantity' => 0
```

```
            ]
        ],
        'LambdaFunctionAssociations' => ['Quantity' => 0],
        'MaxTTL' => 0,
        'MinTTL' => 0,
        'SmoothStreaming' => false,
        'TargetOriginId' => $originName,
        'TrustedSigners' => [
            'Enabled' => false,
            'Quantity' => 0
        ],
        'ViewerProtocolPolicy' => 'allow-all'
    ];
    $enabled = false;
    $origin = [
        'Items' => [
            [
                'DomainName' => $s3BucketURL,
                'Id' => $originName,
                'OriginPath' => '',
                'CustomHeaders' => ['Quantity' => 0],
                'S3OriginConfig' => ['OriginAccessIdentity' => '']
            ]
        ],
        'Quantity' => 1
    ];
    $distribution = [
        'CallerReference' => $callerReference,
        'Comment' => $comment,
        'DefaultCacheBehavior' => $defaultCacheBehavior,
        'Enabled' => $enabled,
        'Origins' => $origin
    ];

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createS3Distribution($cloudFrontClient, $distribution);
}

// Uncomment the following line to run this code in an AWS account.
```

```
// createsTheS3Distribution();
```

擷取 CloudFront 分佈

若要擷取指定 CloudFront 分佈的狀態和詳細資訊，請使用 GetDistribution 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function getDistribution($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId
        ]);

        $message = '';

        if (isset($result['Distribution']['Status'])) {
            $message = 'The status of the distribution with the ID of ' .
                $result['Distribution']['Id'] . ' is currently ' .
                $result['Distribution']['Status'];
        }

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= ', and the effective URI is ' .
                $result['@metadata']['effectiveUri'] . '.';
        } else {
            $message = 'Error: Could not get the specified distribution. ' .
                'The distribution\'s status is not available.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}
```

```
function getsADistribution()
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo getDistribution($cloudFrontClient, $distributionId);
}

// Uncomment the following line to run this code in an AWS account.
// getsADistribution();
```

列出 CloudFront 分佈

使用 ListDistributions 操作，從目前帳戶取得指定 AWS 區域中的現有 CloudFront 分佈清單。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function listDistributions($cloudFrontClient)
{
    try {
        $result = $cloudFrontClient->listDistributions([]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}

function listTheDistributions()
{
    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
```

```
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-2'
    ]);

    $distributions = listDistributions($cloudFrontClient);

    if (count($distributions) == 0) {
        echo 'Could not find any distributions.';
    } else {
        foreach ($distributions['DistributionList']['Items'] as $distribution) {
            echo 'The distribution with the ID of ' . $distribution['Id'] .
                ' has the status of ' . $distribution['Status'] . '.' . "\n";
        }
    }
}

// Uncomment the following line to run this code in an AWS account.
// listTheDistributions();
```

更新 CloudFront 分佈

更新 CloudFront 分佈類似於建立分佈。不過，當您更新分佈時，必須填寫多個欄位且必須包含所有值。若要對現有分佈進行變更，建議您先擷取現有的分佈，並在 `$distribution` 陣列中更新您要變更的值。

若要更新指定的 CloudFront 分佈，請使用 UpdateDistribution 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function updateDistribution(
    $cloudFrontClient,
    $distributionId,
    $distributionConfig,
    $eTag
```

```
) {
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);

        return 'The distribution with the following effective URI has ' .
            'been updated: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['Distribution']['DistributionConfig'])) {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
```

```
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function updateADistribution()
{
    // $distributionId = 'E1BTGP2EXAMPLE';
    $distributionId = 'E1X3BKQ569KEMH';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To change a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }

    // To change a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration.
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);
```

```
    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    // To change a distribution's configuration, you can set the
    // distribution's related configuration value as part of a change request,
    // for example:
    // 'Enabled' => true
    // Some configuration values are required to be specified as part of a change
    // request, even if you don't plan to change their values. For ones you
    // don't want to change but are required to be specified, you can just reuse
    // their current values, as follows.
    $distributionConfig = [
        'CallerReference' => $currentConfig['DistributionConfig']["CallerReference"],
        'Comment' => $currentConfig['DistributionConfig']["Comment"],
        'DefaultCacheBehavior' => $currentConfig['DistributionConfig']
["DefaultCacheBehavior"],
        'DefaultRootObject' => $currentConfig['DistributionConfig']
["DefaultRootObject"],
        'Enabled' => $currentConfig['DistributionConfig']["Enabled"],
        'Origins' => $currentConfig['DistributionConfig']["Origins"],
        'Aliases' => $currentConfig['DistributionConfig']["Aliases"],
        'CustomErrorResponses' => $currentConfig['DistributionConfig']
["CustomErrorResponses"],
        'HttpVersion' => $currentConfig['DistributionConfig']["HttpVersion"],
        'CacheBehaviors' => $currentConfig['DistributionConfig']["CacheBehaviors"],
        'Logging' => $currentConfig['DistributionConfig']["Logging"],
        'PriceClass' => $currentConfig['DistributionConfig']["PriceClass"],
        'Restrictions' => $currentConfig['DistributionConfig']["Restrictions"],
        'ViewerCertificate' => $currentConfig['DistributionConfig']
["ViewerCertificate"],
        'WebACLId' => $currentConfig['DistributionConfig']["WebACLId"]
    ];

    echo updateDistribution(
        $cloudFrontClient,
        $distributionId,
        $distributionConfig,
        $eTag['ETag']
    );
}

// Uncomment the following line to run this code in an AWS account.
```

```
// updateADistribution();
```

停用 CloudFront 分佈

若要停用或移除分佈，請將其狀態從部署變更為停用。

若要停用指定的 CloudFront 分佈，請使用 DisableDistribution 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function disableDistribution(
    $cloudFrontClient,
    $distributionId,
    $distributionConfig,
    $eTag
) {
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution with the following effective URI has ' .
            'been disabled: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);
```

```
        if (isset($result['Distribution']['DistributionConfig'])) {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function disableADistribution()
```

```
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To disable a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }

    // To delete a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration, including setting the new
    // configuration to "disabled".
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    $distributionConfig = [
        'CacheBehaviors' => $currentConfig['DistributionConfig']["CacheBehaviors"],
        'CallerReference' => $currentConfig['DistributionConfig']["CallerReference"],
        'Comment' => $currentConfig['DistributionConfig']["Comment"],
        'DefaultCacheBehavior' => $currentConfig['DistributionConfig']
["DefaultCacheBehavior"],
        'DefaultRootObject' => $currentConfig['DistributionConfig']
["DefaultRootObject"],
        'Enabled' => false,
        'Origins' => $currentConfig['DistributionConfig']["Origins"],
        'Aliases' => $currentConfig['DistributionConfig']["Aliases"],
        'CustomErrorResponses' => $currentConfig['DistributionConfig']
["CustomErrorResponses"],
        'HttpVersion' => $currentConfig['DistributionConfig']["HttpVersion"],
        'Logging' => $currentConfig['DistributionConfig']["Logging"],
        'PriceClass' => $currentConfig['DistributionConfig']["PriceClass"],
        'Restrictions' => $currentConfig['DistributionConfig']["Restrictions"],
```

```
        'ViewerCertificate' => $currentConfig['DistributionConfig']
["ViewerCertificate"],
        'WebACLId' => $currentConfig['DistributionConfig']["WebACLId"]
    ];

    echo disableDistribution(
        $cloudFrontClient,
        $distributionId,
        $distributionConfig,
        $eTag['ETag']
    );
}

// Uncomment the following line to run this code in an AWS account.
// disableADistribution();
```

刪除 CloudFront 分佈

一旦分佈的狀態為停用，即可將其刪除。

若要移除指定的 CloudFront 分佈，請使用 DeleteDistribution 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function deleteDistribution($cloudFrontClient, $distributionId, $eTag)
{
    try {
        $result = $cloudFrontClient->deleteDistribution([
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution at the following effective URI has ' .
            'been deleted: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
```

```
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function deleteADistribution()
{
    $distributionId = 'E17G7YNEXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To delete a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    } else {
```

```
        echo deleteDistribution(
            $cloudFrontClient,
            $distributionId,
            $eTag['ETag']
        );
    }
}

// Uncomment the following line to run this code in an AWS account.
// deleteADistribution();
```

使用 Amazon CloudFront CloudFront 無效 適用於 PHP 的 AWS SDK

Amazon CloudFront 會在全球節點快取靜態和動態檔案的副本。若要在所有節點移除或更新檔案，請為每個檔案或一組檔案建立失效。

每個日曆月的前 1,000 個失效都是免費的。若要進一步了解如何從 CloudFront 節點移除內容，請參閱使檔案失效。

下列範例示範如何：

- 使用 CreateInvalidation 建立分佈失效。
- 使用 GetInvalidation 取得分佈失效。
- 使用 ListInvalidations 列出分佈。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon CloudFront 的詳細資訊，請參閱 Amazon CloudFront 開發人員指南。

建立分佈失效

指定您需要移除之檔案的路徑位置，以建立 CloudFront 分佈失效。此範例會使分佈中的所有檔案失效，但您可以識別 `Items` 下的特定檔案。

若要建立 CloudFront 分佈失效，請使用 CreateInvalidation 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function createInvalidation(
    $cloudFrontClient,
    $distributionId,
    $callerReference,
    $paths,
    $quantity
) {
    try {
        $result = $cloudFrontClient->createInvalidation([
            'DistributionId' => $distributionId,
            'InvalidationBatch' => [
                'CallerReference' => $callerReference,
                'Paths' => [
                    'Items' => $paths,
                    'Quantity' => $quantity,
                ],
            ]
        ]);

        $message = '';

        if (isset($result['Location'])) {
            $message = 'The invalidation location is: ' . $result['Location'];
        }

        $message .= ' and the effective URI is ' . $result['@metadata']
['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheInvalidation()
{
```

```
    $distributionId = 'E17G7YNEXAMPLE';
    $callerReference = 'my-unique-value';
    $paths = ['/*'];
    $quantity = 1;

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createInvalidation(
        $cloudFrontClient,
        $distributionId,
        $callerReference,
        $paths,
        $quantity
    );
}

// Uncomment the following line to run this code in an AWS account.
// createTheInvalidation();
```

取得分佈失效

若要擷取 CloudFront 分佈失效的狀態和詳細資訊，請使用 GetInvalidation 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function getInvalidation($cloudFrontClient, $distributionId, $invalidationId)
{
    try {
        $result = $cloudFrontClient->getInvalidation([
            'DistributionId' => $distributionId,
            'Id' => $invalidationId,
        ]);
```

```
        $message = '';

        if (isset($result['Invalidation']['Status'])) {
            $message = 'The status for the invalidation with the ID of ' .
                $result['Invalidation']['Id'] . ' is ' .
                $result['Invalidation']['Status'];
        }

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= ', and the effective URI is ' .
                $result['@metadata']['effectiveUri'] . '.';
        } else {
            $message = 'Error: Could not get information about ' .
                'the invalidation. The invalidation\'s status ' .
                'was not available.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getsAnInvalidation()
{
    $distributionId = 'E1BTGP2EXAMPLE';
    $invalidationId = 'I1CDEZZEXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo getInvalidation($cloudFrontClient, $distributionId, $invalidationId);
}

// Uncomment the following line to run this code in an AWS account.
// getsAnInvalidation();
```

列出分佈失效

若要列出所有目前的 CloudFront 分佈失效，請使用 ListInvalidations 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
function listInvalidations($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->listInvalidations([
            'DistributionId' => $distributionId
        ]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}

function listTheInvalidations()
{
    $distributionId = 'E1WICG1EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $invalidations = listInvalidations(
        $cloudFrontClient,
        $distributionId
    );

    if (isset($invalidations['InvalidationList'])) {
        if ($invalidations['InvalidationList']['Quantity'] > 0) {
            foreach ($invalidations['InvalidationList']['Items'] as $invalidation) {
                echo 'The invalidation with the ID of ' . $invalidation['Id'] .
                    ' has the status of ' . $invalidation['Status'] . '.' . "\n";
            }
        } else {
```

```
            echo 'Could not find any invalidations for the specified distribution.';
        }
    } else {
        echo 'Error: Could not get invalidation information. Could not get ' .
            'information about the specified distribution.';
    }
}

// Uncomment the following line to run this code in an AWS account.
// listTheInvalidations();
```

使用第 3 適用於 PHP 的 AWS SDK 版簽署 Amazon CloudFront URLs

已簽章 URL 可讓您提供使用者存取您的私有內容。已簽署的 URL 包含附加資訊 (例如過期時間)，以便您更有效地控制對內容的存取。此附加資訊顯示在政策聲明中，該政策聲明基於標準政策或自訂政策。如需有關如何設定私有分佈以及為什麼您需要簽署 URLs的資訊，請參閱《Amazon CloudFront 開發人員指南》中的透過 Amazon CloudFront 提供私有內容。 Amazon CloudFront

- 使用 getSignedURL 建立已簽署的 Amazon CloudFront URL。
- 使用 getSignedCookie 建立已簽署的 Amazon CloudFront cookie。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon CloudFront 的詳細資訊，請參閱 Amazon CloudFront 開發人員指南。

簽署私有分佈的 CloudFront URLs

您可以使用 SDK 中的 CloudFront 用戶端來簽署 URL。首先，您必須建立一個 `CloudFrontClient` 物件。您可以使用標準或自訂政策來簽署視訊資源的 CloudFront URL。

匯入

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function signPrivateDistribution(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function signAPrivateDistribution()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo signPrivateDistribution(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );
}
```

```
// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistribution();
```

建立 CloudFront URLs時使用自訂政策

若要使用自訂政策，請提供 `policy` 金鑰，而非 `expires`。

匯入

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function signPrivateDistributionPolicy(
    $cloudFrontClient,
    $resourceKey,
    $customPolicy,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function signAPrivateDistributionPolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
```

```
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "$resourceKey",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": $expires}
            }
        }
    ]
}
POLICY;
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo signPrivateDistributionPolicy(
        $cloudFrontClient,
        $resourceKey,
        $customPolicy,
        $privateKey,
        $keyPairId
    );
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistributionPolicy();
```

使用 CloudFront 簽章的 URL

簽署 URL 的形式各有不同，取決於您要簽署的 URL 是使用「HTTP」還是「RTMP」機制。如果是「HTTP」，則會傳回完整、絕對的 URL。針對「RTMP」，為了您的方便，則只會傳回相對 URL。這是因為一些播放程式需要將主機和路徑做為單獨的參數提供。

以下範例示範如何使用已簽章的 URL 建構使用 JWPlayer 顯示視訊的網頁。同樣的技巧適用於其他播放程式，例如 FlowPlayer，但需要不同的用戶端程式碼。

```
<html>
<head>
    <title>|CFlong| Streaming Example</title>
    <script type="text/javascript" src="https://example.com/jwplayer.js"></script>
</head>
<body>
    <div id="video">The canned policy video will be here.</div>
    <script type="text/javascript">
        jwplayer('video').setup({
            file: "<?= $streamHostUrl ?>/cfx/st/<?= $signedUrlCannedPolicy ?>",
            width: "720",
            height: "480"
        });
    </script>
</body>
</html>
```

為私有分佈簽署 CloudFront Cookie

做為已簽章的 URL 替代方案，您還可以授予用戶端透過已簽章的 cookie 存取私有分發權限。已簽章的 cookie 可讓您提供對多個限制檔案的存取，例如 HLS 格式視訊的所有檔案或網站中訂閱者區域的所有檔案。如需為何您可能想要使用已簽章的 Cookie 而非已簽章URLs 的詳細資訊（反之亦然），請參閱《Amazon CloudFront 開發人員指南》中的選擇已簽章URLs 和已簽章的 Cookie。

建立一個簽章的 cookie 類似於建立一個簽章的 URL。唯一的區別是呼叫的方法 (`getSignedCookie`，而非 `getSignedUrl`)。

匯入

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function signCookie(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
```

```
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
    }
}

function signACookie()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $result = signCookie(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );

    /* If successful, returns something like:
    CloudFront-Expires = 1589926678
    CloudFront-Signature = Lv1DyC2q...2HPXaQ__
    CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
    */
    foreach ($result as $key => $value) {
        echo $key . ' = ' . $value . "\n";
```

```
    }
}

// Uncomment the following line to run this code in an AWS account.
// signACookie();
```

建立 CloudFront Cookie 時使用自訂政策

如同 `getSignedUrl`，您可以透過自訂政策簽署 cookie 以提供 `'policy'` 參數，而非 `expires` 參數與 `url` 參數。自訂政策可以包含在資源金鑰中的萬用字元。這可讓您建立多個檔案的單一簽章 cookie。

`getSignedCookie` 會傳回一系列的金鑰值對，所有的金鑰值對都必須設定為 cookie，以將存取權授予私有分發。

匯入

```
require 'vendor/autoload.php';

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function signCookiePolicy(
    $cloudFrontClient,
    $customPolicy,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
    }
}
```

```
function signACookiePolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "{$resourceKey}",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": {$expires}}
            }
        }
    ]
}
POLICY;
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $result = signCookiePolicy(
        $cloudFrontClient,
        $customPolicy,
        $privateKey,
        $keyPairId
    );

    /* If successful, returns something like:
    CloudFront-Policy = eyJTdGF0...fX19XX0_
    CloudFront-Signature = RowqEQWZ...N8vetw__
    CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
    */
    foreach ($result as $key => $value) {
        echo $key . ' = ' . $value . "\n";
    }
}
```

```
// Uncomment the following line to run this code in an AWS account.
// signACookiePolicy();
```

將 CloudFront Cookie 傳送至 Guzzle 用戶端

您也可以將這些 cookie 傳遞到 `GuzzleHttp\Cookie\CookieJar` 以搭配 Guzzle 用戶端使用。

```
use GuzzleHttp\Client;
use GuzzleHttp\Cookie\CookieJar;

$distribution = "example-distribution.cloudfront.net";
$client = new \GuzzleHttp\Client([
    'base_uri' => "https://$distribution",
    'cookies' => CookieJar::fromArray($signedCookieCustomPolicy, $distribution),
]);

$client->get('video.mp4');
```

如需詳細資訊，請參閱《Amazon CloudFront 開發人員指南》中的使用已簽章的 Cookie。

使用第 3 適用於 PHP 的 AWS SDK 版簽署自訂 Amazon CloudSearch 網域請求

Amazon CloudSearch 網域請求可以自訂超過 支援的請求 適用於 PHP 的 AWS SDK。如果您需要對受 IAM 身分驗證保護的網域提出自訂請求，您可以使用 SDK 的憑證提供者和簽署者來簽署任何 PSR-7 請求。

例如，您若依照 CloudSearch 入門指南所述步驟進行操作，且希望使用步驟 3 中受 IAM 保護的網域，則必須依照下述方式簽署並執行請求。

下列範例示範如何：

- 使用 SignatureV4 使用 AWS 簽署通訊協定簽署請求。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

簽署 Amazon CloudSearch 網域請求

匯入

```
require './vendor/autoload.php';

use Aws\Credentials\CredentialProvider;
use Aws\Signature\SignatureV4;
use GuzzleHttp\Client;
use GuzzleHttp\Psr7\Request;
```

範例程式碼

```
function searchDomain(
    $client,
    $domainName,
    $domainId,
    $domainRegion,
    $searchString
) {
    $domainPrefix = 'search-';
    $cloudSearchDomain = 'cloudsearch.amazonaws.com';
    $cloudSearchVersion = '2013-01-01';
    $searchPrefix = 'search?';

    // Specify the search to send.
    $request = new Request(
        'GET',
        "https://$domainPrefix$domainName-$domainId.$domainRegion." .
            "$cloudSearchDomain/$cloudSearchVersion/" .
            "$searchPrefix$searchString"
    );

    // Get default AWS account access credentials.
    $credentials = call_user_func(CredentialProvider::defaultProvider())->wait();

    // Sign the search request with the credentials.
    $signer = new SignatureV4('cloudsearch', $domainRegion);
    $request = $signer->signRequest($request, $credentials);

    // Send the signed search request.
    $response = $client->send($request);
```

```
    // Report the search results, if any.
    $results = json_decode($response->getBody());

    $message = '';

    if ($results->hits->found > 0) {
        $message .= 'Search results:' . "\n";

        foreach ($results->hits->hit as $hit) {
            $message .= $hit->fields->title . "\n";
        }
    } else {
        $message .= 'No search results.';
    }

    return $message;
}

function searchADomain()
{
    $domainName = 'my-search-domain';
    $domainId = '7kbitd6nyiglhdtmssxEXAMPLE';
    $domainRegion = 'us-east-1';
    $searchString = 'q=star+wars&return=title';
    $client = new Client();

    echo searchDomain(
        $client,
        $domainName,
        $domainId,
        $domainRegion,
        $searchString
    );
}

// Uncomment the following line to run this code in an AWS account.
// searchADomain();
```

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon CloudWatch 範例

Amazon CloudWatch (CloudWatch) 是一種 Web 服務， AWS 可即時監控您的 Amazon Web Services 資源和您在其上執行的應用程式。您可以使用 CloudWatch 收集和追蹤指標，這些是您可以為您的資

源和應用程式測量的變數。CloudWatch 警示會根據您定義的規則，傳送通知或自動變更您要監控的資源。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon CloudWatch 警示
- 使用第 3 適用於 PHP 的 AWS SDK 版從 Amazon CloudWatch 取得指標
- 使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon CloudWatch 中發佈自訂指標
- 使用第 3 適用於 PHP 的 AWS SDK 版將事件傳送至 Amazon CloudWatch Events
- 使用警示動作搭配 Amazon CloudWatch 警示搭配第 3 適用於 PHP 的 AWS SDK 版

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon CloudWatch 警示

Amazon CloudWatch 警示會在您指定的期間內監看單一指標。警示會根據在數個期間與指定閾值相關的指標值，來執行一個或多個動作。

下列範例示範如何：

- 使用 DescribeAlarms 描述警示。
- 使用 PutMetricAlarm 建立警示。
- 使用 DeleteAlarms 刪除警示。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

描述警示

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function describeAlarms($cloudWatchClient)
{
    try {
        $result = $cloudWatchClient->describeAlarms();

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'Alarms at the effective URI of ' .
                $result['@metadata']['effectiveUri'] . "\n\n";

            if (isset($result['CompositeAlarms'])) {
                $message .= "Composite alarms:\n";

                foreach ($result['CompositeAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= "No composite alarms found.\n";
            }

            if (isset($result['MetricAlarms'])) {
                $message .= "Metric alarms:\n";

                foreach ($result['MetricAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= 'No metric alarms found.';
            }
        } else {
            $message .= 'No alarms found.';
        }

        return $message;
```

```
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function describeTheAlarms()
{
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo describeAlarms($cloudWatchClient);
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarms();
```

建立警示

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function putMetricAlarm(
    $cloudWatchClient,
    $cloudWatchRegion,
    $alarmName,
    $namespace,
    $metricName,
    $dimensions,
    $statistic,
    $period,
    $comparison,
    $threshold,
```

```
    $evaluationPeriods
) {
    try {
        $result = $cloudWatchClient->putMetricAlarm([
            'AlarmName' => $alarmName,
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'Statistic' => $statistic,
            'Period' => $period,
            'ComparisonOperator' => $comparison,
            'Threshold' => $threshold,
            'EvaluationPeriods' => $evaluationPeriods
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully created or updated specified alarm.';
            } else {
                return 'Could not create or update specified alarm.';
            }
        } else {
            return 'Could not create or update specified alarm.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Resource',
```

```
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricAlarm(
        $cloudWatchClient,
        $cloudWatchRegion,
        $alarmName,
        $namespace,
        $metricName,
        $dimensions,
        $statistic,
        $period,
        $comparison,
        $threshold,
        $evaluationPeriods
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricAlarm();
```

刪除警示

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function deleteAlarms($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->deleteAlarms([
            'AlarmNames' => $alarmNames
        ]);

        return 'The specified alarms at the following effective URI have ' .
            'been deleted or do not currently exist: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function deleteTheAlarms()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo deleteAlarms($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// deleteTheAlarms();
```

使用第 3 適用於 PHP 的 AWS SDK 版從 Amazon CloudWatch 取得指標

指標是有關您系統效能的資料。您可以啟用某些資源的詳細監控，例如 Amazon EC2 執行個體或您自己的應用程式指標。

下列範例示範如何：

- 使用 ListMetrics 列出指標。
- 使用 DescribeAlarmsForMetric 擷取指標警示。
- 使用 GetMetricStatistics 取得指定指標的統計資料。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

列出指標

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function listMetrics($cloudWatchClient)
{
    try {
        $result = $cloudWatchClient->listMetrics();

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'For the effective URI at ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if (
```

```
                (isset($result['Metrics'])) and
                (count($result['Metrics']) > 0)
            ) {
                $message .= "Metrics found:\n\n";

                foreach ($result['Metrics'] as $metric) {
                    $message .= 'For metric ' . $metric['MetricName'] .
                        ' in namespace ' . $metric['Namespace'] . ":\n";

                    if (
                        (isset($metric['Dimensions'])) and
                        (count($metric['Dimensions']) > 0)
                    ) {
                        $message .= "Dimensions:\n";

                        foreach ($metric['Dimensions'] as $dimension) {
                            $message .= 'Name: ' . $dimension['Name'] .
                                ', Value: ' . $dimension['Value'] . "\n";
                        }

                        $message .= "\n";
                    } else {
                        $message .= "No dimensions.\n\n";
                    }
                }
            } else {
                $message .= 'No metrics found.';
            }
        } else {
            $message .= 'No metrics found.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function listTheMetrics()
{
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
```

```
    ]);

    echo listMetrics($cloudWatchClient);
}

// Uncomment the following line to run this code in an AWS account.
// listTheMetrics();
```

擷取指標的警示

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function describeAlarmsForMetric(
    $cloudWatchClient,
    $metricName,
    $namespace,
    $dimensions
) {
    try {
        $result = $cloudWatchClient->describeAlarmsForMetric([
            'MetricName' => $metricName,
            'Namespace' => $namespace,
            'Dimensions' => $dimensions
        ]);

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if (
                (isset($result['MetricAlarms'])) and
                (count($result['MetricAlarms']) > 0)
            ) {
```

```
                $message .= 'Matching alarms for ' . $metricName . ":\n\n";

                foreach ($result['MetricAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= 'No matching alarms found for ' . $metricName . '.';
            }
        } else {
            $message .= 'No matching alarms found for ' . $metricName . '.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function describeTheAlarmsForMetric()
{
    $metricName = 'BucketSizeBytes';
    $namespace = 'AWS/S3';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value' => 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'my-bucket'
        ]
    ];

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo describeAlarmsForMetric(
        $cloudWatchClient,
        $metricName,
        $namespace,
        $dimensions
```

```
    );
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarmsForMetric();
```

取得指標統計數字

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function getMetricStatistics(
    $cloudWatchClient,
    $namespace,
    $metricName,
    $dimensions,
    $startTime,
    $endTime,
    $period,
    $statistics,
    $unit
) {
    try {
        $result = $cloudWatchClient->getMetricStatistics([
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'StartTime' => $startTime,
            'EndTime' => $endTime,
            'Period' => $period,
            'Statistics' => $statistics,
            'Unit' => $unit
        ]);

        $message = '';
```

```
        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'For the effective URI at ' .
                $result['@metadata']['effectiveUri'] . "\n\n";

            if (
                (isset($result['Datapoints'])) and
                (count($result['Datapoints']) > 0)
            ) {
                $message .= "Datapoints found:\n\n";

                foreach ($result['Datapoints'] as $datapoint) {
                    foreach ($datapoint as $key => $value) {
                        $message .= $key . ' = ' . $value . "\n";
                    }

                    $message .= "\n";
                }
            } else {
                $message .= 'No datapoints found.';
            }
        } else {
            $message .= 'No datapoints found.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getTheMetricStatistics()
{
    // Average number of Amazon EC2 vCPUs every 5 minutes within
    // the past 3 hours.
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
```

```
        ],
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $startTime = strtotime('-3 hours');
    $endTime = strtotime('now');
    $period = 300; // Seconds. (5 minutes = 300 seconds.)
    $statistics = ['Average'];
    $unit = 'None';

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics(
        $cloudWatchClient,
        $namespace,
        $metricName,
        $dimensions,
        $startTime,
        $endTime,
        $period,
        $statistics,
        $unit
    );

    // Another example: average number of bytes of standard storage in the
    // specified Amazon S3 bucket each day for the past 3 days.

    /*
    $namespace = 'AWS/S3';
    $metricName = 'BucketSizeBytes';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value'=> 'StandardStorage'
```

```
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'my-bucket'
        ]
    ];
    $startTime = strtotime('-3 days');
    $endTime = strtotime('now');
    $period = 86400; // Seconds. (1 day = 86400 seconds.)
    $statistics = array('Average');
    $unit = 'Bytes';

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit);
    */
}

// Uncomment the following line to run this code in an AWS account.
// getTheMetricStatistics();
```

使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon CloudWatch 中發佈自訂指標

指標是有關您系統效能的資料。警示會在您指定的期間，監看單一指標。警示會根據在數個期間與指定閾值相關的指標值，來執行一個或多個動作。

下列範例示範如何：

- 使用 PutMetricData 發佈指標資料。
- 使用 PutMetricAlarm 建立警示。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

發佈指標資料

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function putMetricData(
    $cloudWatchClient,
    $cloudWatchRegion,
    $namespace,
    $metricData
) {
    try {
        $result = $cloudWatchClient->putMetricData([
            'Namespace' => $namespace,
            'MetricData' => $metricData
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully published datapoint(s).';
            } else {
                return 'Could not publish datapoint(s).';
            }
        } else {
            return 'Error: Could not publish datapoint(s).';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function putTheMetricData()
{
```

```
    $namespace = 'MyNamespace';
    $metricData = [
        [
            'MetricName' => 'MyMetric',
            'Timestamp' => 1589228818, // 11 May 2020, 20:26:58 UTC.
            'Dimensions' => [
                [
                    'Name' => 'MyDimension1',
                    'Value' => 'MyValue1'

                ],
                [
                    'Name' => 'MyDimension2',
                    'Value' => 'MyValue2'
                ]
            ],
            'Unit' => 'Count',
            'Value' => 1
        ]
    ];

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricData(
        $cloudWatchClient,
        $cloudWatchRegion,
        $namespace,
        $metricData
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricData();
```

建立警示

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function putMetricAlarm(
    $cloudWatchClient,
    $cloudWatchRegion,
    $alarmName,
    $namespace,
    $metricName,
    $dimensions,
    $statistic,
    $period,
    $comparison,
    $threshold,
    $evaluationPeriods
) {
    try {
        $result = $cloudWatchClient->putMetricAlarm([
            'AlarmName' => $alarmName,
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'Statistic' => $statistic,
            'Period' => $period,
            'ComparisonOperator' => $comparison,
            'Threshold' => $threshold,
            'EvaluationPeriods' => $evaluationPeriods
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully created or updated specified alarm.';
            } else {
                return 'Could not create or update specified alarm.';
            }
```

```
        } else {
            return 'Could not create or update specified alarm.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);
```

```
    echo putMetricAlarm(
        $cloudWatchClient,
        $cloudWatchRegion,
        $alarmName,
        $namespace,
        $metricName,
        $dimensions,
        $statistic,
        $period,
        $comparison,
        $threshold,
        $evaluationPeriods
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricAlarm();
```

使用第 3 適用於 PHP 的 AWS SDK 版將事件傳送至 Amazon CloudWatch Events

CloudWatch Events 提供近乎即時的系統事件串流，將 Amazon Web Services (AWS) 資源的變更描述到任何各種目標。使用簡單的規則，您可以比對事件，並將這些事件轉傳到一或多個目標函數或串流。

下列範例示範如何：

- 使用 PutRule 建立規則。
- 使用 PutTargets 在規則中加入目標。
- 使用 PutEvents 將自訂事件傳送至 CloudWatch Events。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立規則

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
]);

try {
    $result = $client->putRule([
        'Name' => 'DEMO_EVENT', // REQUIRED
        'RoleArn' => 'IAM_ROLE_ARN',
        'ScheduleExpression' => 'rate(5 minutes)',
        'State' => 'ENABLED',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

將目標新增至規則

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
```

```
]);

try {
    $result = $client->putTargets([
        'Rule' => 'DEMO_EVENT', // REQUIRED
        'Targets' => [ // REQUIRED
            [
                'Arn' => 'LAMBDA_FUNCTION_ARN', // REQUIRED
                'Id' => 'myCloudWatchEventsTarget' // REQUIRED
            ],
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

傳送自訂事件

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
]);

try {
    $result = $client->putEvents([
        'Entries' => [ // REQUIRED
            [
                'Detail' => '<string>',
                'DetailType' => '<string>',
```

```
                'Resources' => ['<string>'],
                'Source' => '<string>',
                'Time' => time()
            ],
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用警示動作搭配 Amazon CloudWatch 警示搭配第 3 適用於 PHP 的 AWS SDK 版

使用警示動作來建立自動停止、終止、重新啟動或復原 Amazon EC2 執行個體的警示。當執行個體不再需要執行，您可以使用停止或終止動作。您可以使用重新啟動和恢復動作來自動重新啟動這些執行個體。

下列範例示範如何：

- 使用 EnableAlarmActions 對指定的警示啟用動作。
- 使用 DisableAlarmActions 對指定的警示停用動作。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

啟用警示動作

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
function enableAlarmActions($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->enableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been enabled.';
        } else {
            return'Actions for some matching alarms ' .
                'might not have been enabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function enableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo enableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// enableTheAlarmActions();
```

停用警示動作

匯入

```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
```

```
use Aws\Exception\AwsException;
```

範例程式碼

```
function disableAlarmActions($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->disableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been disabled.';
        } else {
            return 'Actions for some matching alarms ' .
                'might not have been disabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function disableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo disableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// disableTheAlarmActions();
```

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon EC2 範例

Amazon Elastic Compute Cloud (Amazon EC2) 是一種 Web 服務，可在雲端提供虛擬伺服器託管。它旨在透過提供可調整大小的運算容量，讓開發人員更輕鬆地進行 Web 規模雲端運算。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon EC2 執行個體
- 搭配 Amazon EC2 與第 3 適用於 PHP 的 AWS SDK 版使用彈性 IP 地址
- 搭配第 3 適用於 PHP 的 AWS SDK 版使用 Amazon EC2 的區域和可用區域
- 使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon EC2 金鑰對
- 使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon EC2 中使用安全群組

使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon EC2 執行個體

下列範例示範如何：

- 使用 DescribeInstances 描述 Amazon EC2 執行個體。
- 使用 MonitorInstances 對執行中的執行個體啟用詳細監控。
- 使用 UnmonitorInstances 對執行中的執行個體停用監控。
- 使用 StartInstances 啟動您先前已停止的 Amazon EBS 後端 AMI。
- 使用 StopInstances 停止 Amazon EBS 支援的執行個體。
- 使用 RebootInstances 請求重新啟動一個或多個執行個體。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

描述執行個體

匯入

```
require 'vendor/autoload.php';

use Aws\Ec2\Ec2Client;
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
$result = $ec2Client->describeInstances();
echo "Instances: \n";
foreach ($result['Reservations'] as $reservation) {
    foreach ($reservation['Instances'] as $instance) {
        echo "InstanceId: {$instance['InstanceId']} - {$instance['State']['Name']} \n";
    }
}
```

啟用和停用監控

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
```

```
$instanceIds = ['InstanceID1', 'InstanceID2'];

$monitorInstance = 'ON';

if ($monitorInstance == 'ON') {
    $result = $ec2Client->monitorInstances([
        'InstanceIds' => $instanceIds
    ]);
} else {
    $result = $ec2Client->unmonitorInstances([
        'InstanceIds' => $instanceIds
    ]);
}

var_dump($result);
```

啟動及停止 執行個體

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$action = 'START';

$instanceIds = ['InstanceID1', 'InstanceID2'];

if ($action == 'START') {
    $result = $ec2Client->startInstances([
        'InstanceIds' => $instanceIds,
    ]);
} else {
```

```
    $result = $ec2Client->stopInstances([
        'InstanceIds' => $instanceIds,
    ]);
}

var_dump($result);
```

重新啟動執行個體

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$instanceIds = ['InstanceID1', 'InstanceID2'];

$result = $ec2Client->rebootInstances([
    'InstanceIds' => $instanceIds
]);

var_dump($result);
```

搭配 Amazon EC2 與第 3 適用於 PHP 的 AWS SDK 版使用彈性 IP 地址

彈性 IP 地址是針對動態雲端運算設計的靜態 IP 地址。彈性 IP 地址與您的 相關聯 AWS 帳戶。這是一個公有 IP 地址，可從網際網路存取。如果您的執行個體沒有公有 IP 地址，可以將彈性 IP 地址與執行個體建立關聯，來啟用與網際網路通訊的功能。

下列範例示範如何：

- 使用 DescribeInstances 描述一個或多個執行個體。

- 使用 AllocateAddress 擷取彈性 IP 地址。
- 使用 AssociateAddress 建立彈性 IP 地址與執行個體的關聯。
- 使用 ReleaseAddress 發佈彈性 IP 地址。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

描述執行個體

匯入

```
require 'vendor/autoload.php';

use Aws\Ec2\Ec2Client;
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
$result = $ec2Client->describeInstances();
echo "Instances: \n";
foreach ($result['Reservations'] as $reservation) {
    foreach ($reservation['Instances'] as $instance) {
        echo "InstanceId: {$instance['InstanceId']} - {$instance['State']['Name']} \n";
    }
}
```

配置和關聯地址

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$instanceId = 'InstanceID';

$allocation = $ec2Client->allocateAddress(array(
    'DryRun' => false,
    'Domain' => 'vpc',
));

$result = $ec2Client->associateAddress(array(
    'DryRun' => false,
    'InstanceId' => $instanceId,
    'AllocationId' => $allocation->get('AllocationId')
));

var_dump($result);
```

釋出地址

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
```

```
    'profile' => 'default'
]);

$associationID = 'AssociationID';

$allocationID = 'AllocationID';

$result = $ec2Client->disassociateAddress([
    'AssociationId' => $associationID,
]);

$result = $ec2Client->releaseAddress([
    'AllocationId' => $allocationID,
]);

var_dump($result);
```

搭配第 3 適用於 PHP 的 AWS SDK 版使用 Amazon EC2 的區域和可用區域

Amazon EC2 託管於全球多個位置。這些位置由 AWS 區域和可用區域組成。每個區域都是獨立的地理區域，具有多個稱為可用區域的隔離位置。Amazon EC2 可讓您將執行個體和資料放置在多個位置。

下列範例示範如何：

- 使用 DescribeAvailabilityZones 描述可供您使用的可用區域。
- 使用 DescribeRegions 描述目前可供您使用 AWS 的區域。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

描述可用區域

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeAvailabilityZones();

var_dump($result);
```

描述區域

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeRegions();

var_dump($result);
```

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon EC2 金鑰對

Amazon EC2 使用公有金鑰加密法將登入資訊進行加密及解密。公有金鑰加密法使用公開金鑰來加密資料。之後，收件人會使用私有金鑰來解密資料。公有金鑰和私有金鑰稱為金鑰對。

下列範例示範如何：

- 使用 CreateKeyPair 建立 2048 位元的 RSA 金鑰對。
- 使用 DeleteKeyPair 刪除指定的金鑰對。
- 使用 DescribeKeyPairs 描述一個或多個金鑰對。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立金鑰對

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$keyPairName = 'my-keypair';

$result = $ec2Client->createKeyPair(array(
    'KeyName' => $keyPairName
));

// Save the private key
$saveKeyLocation = getenv('HOME') . "/.ssh/{$keyPairName}.pem";
file_put_contents($saveKeyLocation, $result['keyMaterial']);

// Update the key's permissions so it can be used with SSH
chmod($saveKeyLocation, 0600);
```

刪除金鑰對

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$keyPairName = 'my-keypair';

$result = $ec2Client->deleteKeyPair(array(
    'KeyName' => $keyPairName
));

var_dump($result);
```

描述金鑰對

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
```

```
]);

$result = $ec2Client->describeKeyPairs();

var_dump($result);
```

使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon EC2 中使用安全群組

Amazon EC2 安全群組充當虛擬防火牆，可控制一或多個執行個體的流量。您在各個安全群組新增規則，允許流量往返於建立關聯的執行個體。您可隨時修改安全群組規則。新規則會自動套用至與安全群組建立關聯的所有執行個體。

下列範例示範如何：

- 使用 DescribeSecurityGroups 描述一個或多個安全群組。
- 使用 AuthorizeSecurityGroupIngress 新增傳入規則到安全群組。
- 使用 CreateSecurityGroup 建立安全群組。
- 使用 DeleteSecurityGroup 刪除安全群組。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

描述安全群組

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
```

```
    'profile' => 'default'
]);

$result = $ec2Client->describeSecurityGroups();

var_dump($result);
```

新增輸入規則

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->authorizeSecurityGroupIngress(array(
    'GroupName' => 'string',
    'SourceSecurityGroupName' => 'string'
));

var_dump($result);
```

建立安全群組

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

// Create the security group
$securityGroupName = 'my-security-group';
$result = $ec2Client->createSecurityGroup(array(
    'GroupId' => $securityGroupName,

));

// Get the security group ID (optional)
$securityGroupId = $result->get('GroupId');

echo "Security Group ID: " . $securityGroupId . '\n';
```

刪除安全群組

匯入

```
require 'vendor/autoload.php';
```

範例程式碼

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$securityGroupId = 'my-security-group-id';

$result = $ec2Client->deleteSecurityGroup([
    'GroupId' => $securityGroupId
]);

var_dump($result);
```

使用第 3 適用於 PHP 的 AWS SDK 版簽署 Amazon OpenSearch Service 搜尋請求

Amazon OpenSearch Service 是一種受管服務，可讓您輕鬆部署、操作和擴展 Amazon OpenSearch Service，這是一種熱門的開放原始碼搜尋和分析引擎。OpenSearch Service 可讓您直接存取 Amazon OpenSearch Service API。這表示開發人員可以使用他們熟悉的工具，以及強大的安全選項。許多 Amazon OpenSearch Service 用戶端支援請求簽署，但如果您使用的用戶端不是 ，您可以使用 的內建憑證提供者和簽署者簽署任意 PSR-7 請求 適用於 PHP 的 AWS SDK。

下列範例示範如何：

- 使用 SignatureV4 使用 AWS 簽署通訊協定簽署請求。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

簽署 OpenSearch Service 請求

OpenSearch Service 使用 Signature 第 4 版。這表示您需要針對服務的簽署名稱（在此案例中`es`為）和 OpenSearch Service 網域 AWS 的區域簽署請求。您可以在 的區域和端點頁面上找到 OpenSearch Service 支援的區域完整清單 Amazon Web Services 一般參考。 AWS不過，在此範例中，我們會針對`us-west-2`區域中的 OpenSearch Service 網域簽署請求。

您需要提供登入資料，您可以使用 SDK 的預設提供者鏈結或第 3 適用於 PHP 的 AWS SDK 版登入資料中所述的任何形式的登入資料來執行此操作。您也將需要 PSR-7 請求 (假設在下方的程式碼中命名為 `$psr7Request`)。

```
// Pull credentials from the default provider chain
$provider = Aws\Credentials\CredentialProvider::defaultProvider();
$credentials = call_user_func($provider)->wait();

// Create a signer with the service's signing name and Region
$signer = new Aws\Signature\SignatureV4('es', 'us-west-2');
```

```
// Sign your request
$signedRequest = $signer->signRequest($psr7Request, $credentials);
```

AWS Identity and Access Management 使用第 3 適用於 PHP 的 AWS SDK 版的範例

AWS Identity and Access Management (IAM) 是一種 Web 服務，可讓 Amazon Web Services 客戶管理 中的使用者和使用者許可 AWS。此服務以雲端中有多個使用者或系統使用 AWS 產品的組織為目標。透過 IAM，您可以集中管理使用者、存取金鑰等安全登入資料，以及控制使用者可以存取哪些 AWS 資源的許可。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版管理 IAM 存取金鑰
- 使用第 3 適用於 PHP 的 AWS SDK 版管理 IAM 使用者
- 搭配第 3 適用於 PHP 的 AWS SDK 版使用 IAM 帳戶別名
- 使用第 3 適用於 PHP 的 AWS SDK 版的 IAM 政策
- 使用第 3 適用於 PHP 的 AWS SDK 版的 IAM 伺服器憑證

使用第 3 適用於 PHP 的 AWS SDK 版管理 IAM 存取金鑰

使用者需要自己的存取金鑰才能對 進行程式設計呼叫 AWS。為了滿足這個需求，您可以為 IAM 使用者建立、修改、查看或輪換存取金鑰 (存取金鑰 ID 和私密存取金鑰)。在預設情況下，您建立存取金鑰時，其狀態為「作用中」。這表示使用者可以使用存取金鑰進行 API 呼叫。

下列範例示範如何：

- 使用 CreateAccessKey 建立私密存取金鑰與對應的存取金鑰 ID。
- 使用 ListAccessKeys 傳回與 IAM 使用者相關聯的存取金鑰 IDs 的相關資訊。
- 使用 GetAccessKeyLastUsed 擷取有關上次使用存取金鑰的資訊。

- 使用 UpdateAccessKey 將存取金鑰的狀態從「作用中」變更為「非作用中」(或相反)。
- 使用 DeleteAccessKey 刪除與 IAM 使用者相關聯的存取金鑰對。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立存取金鑰

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->createAccessKey([
        'UserName' => 'IAM_USER_NAME',
    ]);
    $keyID = $result['AccessKey']['AccessKeyId'];
    $createDate = $result['AccessKey']['CreateDate'];
    $userName = $result['AccessKey']['UserName'];
    $status = $result['AccessKey']['Status'];
    // $secretKey = $result['AccessKey']['SecretAccessKey']
    echo "<p>AccessKey " . $keyID . " created on " . $createDate . "</p>";
    echo "<p>Username: " . $userName . "</p>";
    echo "<p>Status: " . $status . "</p>";
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

列出存取金鑰

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->listAccessKeys();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

取得存取金鑰上次使用的相關資訊

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->getAccessKeyLastUsed([
        'AccessKeyId' => 'ACCESS_KEY_ID', // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

更新存取金鑰

匯入

```

require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
```

```
    $result = $client->updateAccessKey([
        'AccessKeyId' => 'ACCESS_KEY_ID', // REQUIRED
        'Status' => 'Inactive', // REQUIRED
        'UserName' => 'IAM_USER_NAME',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除存取金鑰

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteAccessKey([
        'AccessKeyId' => 'ACCESS_KEY_ID', // REQUIRED
        'UserName' => 'IAM_USER_NAME',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版管理 IAM 使用者

IAM 使用者是您在 中建立的實體 AWS ，代表用來與之互動的人員或服務 AWS。中的使用者 AWS 包含名稱和登入資料。

下列範例示範如何：

- 使用 CreateUser 建立新的 IAM 使用者。
- 使用 ListUsers 列出 IAM 使用者。
- 使用 UpdateUser 更新 IAM 使用者。
- 使用 GetUser 擷取 IAM 使用者的相關資訊。
- 使用 DeleteUser 刪除 IAM 使用者。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立 IAM 使用者

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
```

```
    $result = $client->createUser(array(
        // UserName is required
        'UserName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

列出 IAM 使用者

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->listUsers();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

更新 IAM 使用者

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->updateUser([
        // UserName is required
        'UserName' => 'string1',
        'NewUserName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

取得 IAM 使用者的相關資訊

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
```

```
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->getUser([
        'UserName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除 IAM 使用者

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteUser([
        // UserName is required
        'UserName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

搭配第 3 適用於 PHP 的 AWS SDK 版使用 IAM 帳戶別名

如果您希望登入頁面的 URL 包含您的公司名稱或其他易記識別符，而不是您的 AWS 帳戶 ID，您可以為您的 AWS 帳戶 ID 建立別名。如果您建立 AWS 帳戶 別名，您的登入頁面 URL 會變更為包含別名。

下列範例示範如何：

- 使用 CreateAccountAlias 建立別名。
- AWS 帳戶 使用 ListAccountAliases 列出與 相關聯的別名。
- 使用 DeleteAccountAlias 刪除別名。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立別名

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
```

```
    'version' => '2010-05-08'
]);

try {
    $result = $client->createAccountAlias(array(
        // AccountAlias is required
        'AccountAlias' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

列出帳戶別名

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->listAccountAliases();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除別名

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteAccountAlias([
        // AccountAlias is required
        'AccountAlias' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版的 IAM 政策

您將授與建立政策的權限給使用者。政策為列出使用者可執行的動作以及這些動作可影響的資源之清單文件，在預設情況下，未明確允許的任何動作或資源將被拒絕。可建立政策並連接至使用者、使用者群組、使用者擔任的角色以及資源。

下列範例示範如何：

- 使用 CreatePolicy 建立受管政策。
- 使用 AttachRolePolicy 附加政策到角色。
- 使用 AttachUserPolicy 附加政策到使用者。

- 使用 AttachGroupPolicy 將政策連接到角色。
- 使用 DetachRolePolicy 移除角色政策。
- 使用 DetachUserPolicy 移除使用者政策。
- 使用 DetachGroupPolicy 移除群組政策。
- 使用 DeletePolicy 刪除受管政策。
- 使用 DeleteRolePolicy 刪除角色政策。
- 使用 DeleteUserPolicy 刪除使用者政策。
- 使用 DeleteGroupPolicy 刪除群組政策。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立政策

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

$myManagedPolicy = '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "logs:CreateLogGroup",
            "Resource": "RESOURCE_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
            "dynamodb:DeleteItem",
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:Scan",
            "dynamodb:UpdateItem"
        ],
            "Resource": "RESOURCE_ARN"
        }
    ]
}';

try {
    $result = $client->createPolicy(array(
        // PolicyName is required
        'PolicyName' => 'myDynamoDBPolicy',
        // PolicyDocument is required
        'PolicyDocument' => $myManagedPolicy
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

將政策連接至角色

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

$roleName = 'ROLE_NAME';

$policyName = 'AmazonDynamoDBFullAccess';

$policyArn = 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess';

try {
    $attachedRolePolicies = $client->getIterator('ListAttachedRolePolicies', ([
        'RoleName' => $roleName,
    ]));
    if (count($attachedRolePolicies) > 0) {
        foreach ($attachedRolePolicies as $attachedRolePolicy) {
            if ($attachedRolePolicy['PolicyName'] == $policyName) {
                echo $policyName . " is already attached to this role. \n";
                exit();
            }
        }
    }
    $result = $client->attachRolePolicy(array(
        // RoleName is required
        'RoleName' => $roleName,
        // PolicyArn is required
        'PolicyArn' => $policyArn
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

將政策連接至使用者

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

$userName = 'USER_NAME';

$policyName = 'AmazonDynamoDBFullAccess';

$policyArn = 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess';

try {
    $attachedUserPolicies = $client->getIterator('ListAttachedUserPolicies', ([
        'UserName' => $userName,
    ]));
    if (count($attachedUserPolicies) > 0) {
        foreach ($attachedUserPolicies as $attachedUserPolicy) {
            if ($attachedUserPolicy['PolicyName'] == $policyName) {
                echo $policyName . " is already attached to this role. \n";
                exit();
            }
        }
    }
    $result = $client->attachUserPolicy(array(
        // UserName is required
        'UserName' => $userName,
        // PolicyArn is required
        'PolicyArn' => $policyArn,
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

將政策連接至群組

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->attachGroupPolicy(array(
        // GroupName is required
        'GroupName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

分離使用者政策

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->detachUserPolicy([
        // UserName is required
        'UserName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

分離群組政策

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);
```

```
try {
    $result = $client->detachGroupPolicy([
        // GroupName is required
        'GroupName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除政策

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deletePolicy(array(
        // PolicyArn is required
        'PolicyArn' => 'string'
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

刪除角色政策

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteRolePolicy([
        // RoleName is required
        'RoleName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除使用者政策

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteUserPolicy([
        // UserName is required
        'UserName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除群組政策

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
```

```
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteGroupPolicy(array(
        // GroupName is required
        'GroupName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版的 IAM 伺服器憑證

若要啟用網站或應用程式的 HTTPS 連線 AWS，您需要 SSL/TLS 伺服器憑證。若要將從外部供應商取得的憑證與網站或應用程式搭配使用 AWS，您必須將憑證上傳至 IAM 或將其匯入其中 AWS Certificate Manager。

下列範例示範如何：

- 使用 ListServerCertificates 列出存放在 IAM 中的憑證。
- 使用 GetServerCertificate 擷取關於憑證的資訊。
- 使用 UpdateServerCertificate 更新憑證。
- 使用 DeleteServerCertificate 刪除憑證。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

列出伺服器憑證

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->listServerCertificates();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

擷取伺服器憑證

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
```

```
]);

try {
    $result = $client->getServerCertificate([
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

更新伺服器憑證

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->updateServerCertificate([
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
        'NewServerCertificateName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

刪除伺服器憑證

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

範例程式碼

```
$client = new IamClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

try {
    $result = $client->deleteServerCertificate([
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS Key Management Service 使用第 3 適用於 PHP 的 AWS SDK 版的範例

AWS Key Management Service (AWS KMS) 是一種受管服務，可讓您輕鬆建立和控制用來加密資料的加密金鑰。如需 的詳細資訊 AWS KMS，請參閱 Amazon KMS 文件。無論您是撰寫安全的 PHP 應用程式，還是將資料傳送到其他 AWS 服務， 都 AWS KMS 可協助您維持對誰可以使用您的金鑰的控制權，並存取您的加密資料。

第 3 適用於 PHP 的 AWS SDK 版的所有範例程式碼都可在 GitHub 上取得。

主題

- 使用 AWS KMS API 和 第 適用於 PHP 的 AWS SDK 3 版使用金鑰
- 使用第 3 適用於 PHP 的 AWS SDK 版加密和解密 AWS KMS 資料金鑰
- 使用第 3 適用於 PHP 的 AWS SDK 版使用 AWS KMS 金鑰政策
- 使用 AWS KMS API 和 適用於 PHP 的 AWS SDK 第 3 版使用授予
- 使用 AWS KMS API 和 適用於 PHP 的 AWS SDK 第 3 版使用別名

使用 AWS KMS API 和 第 適用於 PHP 的 AWS SDK 3 版使用金鑰

AWS Key Management Service (AWS KMS) 中的主要資源為 AWS KMS keys。您可以使用 KMS 金鑰來加密您的資料。

下列範例示範如何：

- 使用 CreateKey 建立客戶 KMS 金鑰。
- 使用 GenerateDataKey 產生資料金鑰。
- 使用 DescribeKey 檢視 KMS 金鑰。
- 使用 ListKeys 取得 KMS 金鑰的金鑰 IDs 和金鑰 ARNS。
- 使用 EnableKey 啟用 KMS 金鑰。
- 使用 DisableKey 停用 KMS 金鑰。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 AWS Key Management Service (AWS KMS) 的詳細資訊，請參閱 AWS KMS 開發人員指南。

建立 KMS 金鑰

若要建立 KMS 金鑰，請使用 CreateKey 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

//Creates a customer master key (CMK) in the caller's AWS account.
$desc = "Key for protecting critical data";

try {
    $result = $KmsClient->createKey([
        'Description' => $desc,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

產生資料金鑰

若要產生資料加密金鑰，請使用 GenerateDataKey 操作。這個操作會傳回它建立的純文字和加密的資料金鑰副本。指定要在 AWS KMS key 其中產生資料金鑰的 。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$keySpec = 'AES_256';

try {
    $result = $KmsClient->generateDataKey([
        'KeyId' => $keyId,
        'KeySpec' => $keySpec,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

檢視 KMS 金鑰

若要取得 KMS 金鑰的詳細資訊，包括 KMS 金鑰的 Amazon Resource Name (ARN) 和金鑰狀態，請使用 DescribeKey 操作。

`DescribeKey` 不會取得別名。若要取得別名，請使用 ListAliases 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
```

```
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->describeKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

取得 KMS 金鑰的金鑰 ARNs

若要取得 KMS 金鑰的 ID 和 ARN，請使用 ListAliases 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$limit = 10;

try {
    $result = $KmsClient->listKeys([
        'Limit' => $limit,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

啟用 KMS 金鑰

若要啟用已停用的 KMS 金鑰，請使用 EnableKey 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->enableKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

停用 KMS 金鑰

若要停用 KMS 金鑰，請使用 DisableKey 操作。停用 KMS 金鑰可防止其被使用。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->disableKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用第 3 適用於 PHP 的 AWS SDK 版加密和解密 AWS KMS 資料金鑰

資料金鑰是加密金鑰，您可以用來加密資料，包括大量資料和其他資料加密金鑰。

您可以使用 AWS Key Management Service的 (AWS KMS) AWS KMS key來產生、加密和解密資料金鑰。

下列範例示範如何：

- 使用 Encrypt 加密資料金鑰。
- 使用 Decrypt 解密資料金鑰。
- 使用 ReEncrypt 以新的 KMS 金鑰重新加密資料金鑰。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 AWS Key Management Service (AWS KMS) 的詳細資訊，請參閱 AWS KMS 開發人員指南。

加密

Encrypt 操作旨在加密資料金鑰，但並不常用。GenerateDataKey 和 GenerateDataKeyWithoutPlaintext 操作會傳回加密的資料金鑰。當您將加密的資料移至新 AWS 區域，並想要在新區域中使用 KMS 金鑰來加密其資料金鑰時，您可以使用 `Encypt` 方法。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$message = pack('c*', 1, 2, 3, 4, 5, 6, 7, 8, 9, 0);

try {
    $result = $KmsClient->encrypt([
```

```
        'KeyId' => $keyId,
        'Plaintext' => $message,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

解密

若要解密資料金鑰，請使用 Decrypt 操作。

`ciphertextBlob` 您指定的 必須是 GenerateDataKey、GenerateDataKeyWithoutPlaintext 或 Encrypt 回應中 `CiphertextBlob` 欄位的值。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$ciphertext = 'Place your cipher text blob here';

try {
    $result = $KmsClient->decrypt([
        'CiphertextBlob' => $ciphertext,
    ]);
    $plaintext = $result['Plaintext'];
    var_dump($plaintext);
} catch (AwsException $e) {
```

```
    // Output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

重新加密

若要解密加密的資料金鑰，然後在不同的 KMS 金鑰下立即重新加密資料金鑰，請使用 ReEncrypt 操作。這些操作完全在內部的伺服器端執行 AWS KMS，因此絕不會在外部公開您的純文字 AWS KMS。

`ciphertextBlob` 您指定的 必須是 GenerateDataKey、GenerateDataKeyWithoutPlaintext 或 Encrypt 回應中 `CiphertextBlob` 欄位的值。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$ciphertextBlob = 'Place your cipher text blob here';

try {
    $result = $KmsClient->reEncrypt([
        'CiphertextBlob' => $ciphertextBlob,
        'DestinationKeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
```

```
}
```

使用第 3 適用於 PHP 的 AWS SDK 版使用 AWS KMS 金鑰政策

建立 時AWS KMS key，您可以判斷誰可以使用和管理該 KMS 金鑰。這些許可包含在稱為金鑰政策的文件中。您可以隨時使用金鑰政策來新增、移除或修改客戶受管 KMS 金鑰的許可，但您無法編輯 AWS 受管 KMS 金鑰的金鑰政策。如需詳細資訊，請參閱 的身分驗證和存取控制 AWS KMS。

下列範例示範如何：

- 使用 ListKeyPolicies 列出金鑰政策的名稱。
- 使用 GetKeyPolicy 取得金鑰政策。
- 使用 PutKeyPolicy 設定金鑰政策。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 AWS Key Management Service (AWS KMS) 的詳細資訊，請參閱 AWS KMS 開發人員指南。

列出所有金鑰政策

若要取得 KMS 金鑰的金鑰政策名稱，請使用 `ListKeyPolicies`操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
```

```
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$limit = 10;

try {
    $result = $KmsClient->listKeyPolicies([
        'KeyId' => $keyId,
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

擷取金鑰政策

若要取得 KMS 金鑰的金鑰政策，請使用 `GetKeyPolicy`操作。

`GetKeyPolicy` 需要政策名稱。除非您在建立 KMS 金鑰時建立金鑰政策，否則唯一的有效政策名稱為預設值。進一步了解《 AWS Key Management Service 開發人員指南》中的預設金鑰政策。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);
```

```
$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$policyName = "default";

try {
    $result = $KmsClient->getKeyPolicy([
        'KeyId' => $keyId,
        'PolicyName' => $policyName
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

設定金鑰政策

若要建立或變更 KMS 金鑰的金鑰政策，請使用 `PutKeyPolicy`操作。

`PutKeyPolicy` 需要政策名稱。除非您在建立 KMS 金鑰時建立金鑰政策，否則唯一的有效政策名稱為預設值。進一步了解《 AWS Key Management Service 開發人員指南》中的預設金鑰政策。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$policyName = "default";

try {
```

```
    $result = $KmsClient->putKeyPolicy([
        'KeyId' => $keyId,
        'PolicyName' => $policyName,
        'Policy' => '{
            "Version": "2012-10-17",
            "Id": "custom-policy-2016-12-07",
            "Statement": [
                { "Sid": "Enable IAM User Permissions",
                "Effect": "Allow",
                "Principal":
                   { "AWS": "arn:aws:iam::111122223333:user/root" },
                "Action": [ "kms:*" ],
                "Resource": "*" },
                { "Sid": "Enable IAM User Permissions",
                "Effect": "Allow",
                "Principal":
                   { "AWS": "arn:aws:iam::111122223333:user/ExampleUser" },
                "Action": [
                    "kms:Encrypt*",
                    "kms:GenerateDataKey*",
                    "kms:Decrypt*",
                    "kms:DescribeKey*",
                    "kms:ReEncrypt*"
                ],
                "Resource": "*" }
            ]
        } '
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 AWS KMS API 和 適用於 PHP 的 AWS SDK 第 3 版使用授予

授與是提供許可的另一個機制。這是金鑰政策的替代方案。您可以使用授予來提供長期存取權，讓 AWS 主體使用您的 AWS Key Management Service (AWS KMS) 客戶受管 AWS KMS keys。如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的在 中授予 AWS KMS。

下列範例示範如何：

- 使用 CreateGrant 建立 KMS 金鑰的授予。
- 使用 ListGrants 檢視 KMS 金鑰的授予。
- 使用 RetireGrant 淘汰 KMS 金鑰的授予。
- 使用 RevokeGrant 撤銷 KMS RevokeGrant。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 AWS Key Management Service (AWS KMS) 的詳細資訊，請參閱 AWS KMS 開發人員指南。

建立授與

若要為 建立授予 AWS KMS key，請使用 CreateGrant 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$granteePrincipal = "arn:aws:iam::111122223333:user/Alice";
$operation = ['Encrypt', 'Decrypt']; // A list of operations that the grant allows.

try {
    $result = $KmsClient->createGrant([
```

```
        'GranteePrincipal' => $granteePrincipal,
        'KeyId' => $keyId,
        'Operations' => $operation
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

檢視授權

若要取得 上授予的詳細資訊 AWS KMS key，請使用 ListGrants 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$limit = 10;

try {
    $result = $KmsClient->listGrants([
        'KeyId' => $keyId,
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

淘汰授予

若要淘汰 的授予 AWS KMS key，請使用 RetireGrant 操作。授予使用完畢後，將其淘汰以進行清理。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$grantToken = 'Place your grant token here';

try {
    $result = $KmsClient->retireGrant([
        'GrantToken' => $grantToken,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

//Can also identify grant to retire by a combination of the grant ID
//and the Amazon Resource Name (ARN) of the customer master key (CMK)
$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$grantId = 'Unique identifier of the grant returned during CreateGrant operation';
```

```
try {
    $result = $KmsClient->retireGrant([
        'GrantId' => $grantToken,
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

撤銷授予

若要撤銷對 的授予 AWS KMS key，請使用 RevokeGrant 操作。您可以撤銷授與以明確拒絕依賴它的操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$grantId = "grant1";

try {
    $result = $KmsClient->revokeGrant([
        'KeyId' => $keyId,
        'GrantId' => $grantId,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 AWS KMS API 和 適用於 PHP 的 AWS SDK 第 3 版使用別名

AWS Key Management Service (AWS KMS) 為AWS KMS key稱為別名的 提供選用的顯示名稱。

下列範例示範如何：

- 使用 CreateAlias 建立別名。
- 使用 ListAliases 檢視別名。
- 使用 UpdateAlias 更新別名。
- 使用 DeleteAlias 刪除別名。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 AWS Key Management Service (AWS KMS) 的詳細資訊，請參閱 AWS KMS 開發人員指南。

建立別名

若要建立 KMS 金鑰的別名，請使用 CreateAlias 操作。別名在帳戶和 AWS 區域中必須是唯一的。如果您為已有別名的 KMS 金鑰建立別名， 會為相同的 KMS 金鑰`CreateAlias`建立另一個別名。其並不會取代現有的別名。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$aliasName = "alias/projectKey1";

try {
    $result = $KmsClient->createAlias([
        'AliasName' => $aliasName,
        'TargetKeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

檢視別名

若要列出呼叫者 AWS 帳戶 和 中的所有別名 AWS 區域，請使用 ListAliases 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
```

```
]);

$limit = 10;

try {
    $result = $KmsClient->listAliases([
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

更新別名

若要將現有的別名關聯至不同的 KMS 金鑰，請使用 UpdateAlias 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$aliasName = "alias/projectKey1";

try {
    $result = $KmsClient->updateAlias([
        'AliasName' => $aliasName,
        'TargetKeyId' => $keyId,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

刪除別名

若要刪除別名，請使用 DeleteAlias 操作。刪除別名不會影響基礎 KMS 金鑰。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$aliasName = "alias/projectKey1";

try {
    $result = $KmsClient->deleteAlias([
        'AliasName' => $aliasName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon Kinesis 範例

Amazon Kinesis 是一種 AWS 服務，可即時收集、處理和分析資料。使用 Amazon Kinesis Data Streams 設定您的資料串流，或使用 Amazon Data Firehose 將資料傳送至 Amazon S3、OpenSearch Service、Amazon Redshift 或 Splunk。

如需 Kinesis 的詳細資訊，請參閱 Amazon Kinesis 文件。

第 3 適用於 PHP 的 AWS SDK 版的所有範例程式碼都可在 GitHub 上取得。

主題

- 使用 Kinesis Data Streams API 和第 3 適用於 PHP 的 AWS SDK 版建立資料串流
- 使用 Kinesis Data Streams API 和第 3 適用於 PHP 的 AWS SDK 版管理資料碎片
- 使用 Firehose API 和 第 3 適用於 PHP 的 AWS SDK 版建立交付串流

使用 Kinesis Data Streams API 和第 3 適用於 PHP 的 AWS SDK 版建立資料串流

Amazon Kinesis Data Streams 可讓您傳送即時資料。使用 Kinesis Data Streams 建立資料生產者，以在每次新增資料時將資料交付至設定的目的地。

如需詳細資訊，請參閱《Amazon Kinesis 開發人員指南》中的建立和管理串流。

下列範例示範如何：

- 使用 CreateAlias 建立資料串流。
- 使用 DescribeStream 取得單一資料串流的詳細資訊。
- 使用 ListStreams 列出現有的資料串流。
- 使用 PutRecord 傳送資料至現有的資料串流。
- 使用 DeleteStream 刪除資料串流。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon Kinesis 開發人員指南的詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南。

使用 Kinesis 資料串流建立資料串流

建立 Kinesis 資料串流，您可以在其中使用下列程式碼範例傳送 Kinesis 要處理的資訊。請參閱 Amazon Kinesis 開發人員指南，進一步了解建立和更新資料串流。

若要建立 Kinesis 資料串流，請使用 CreateStream 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$shardCount = 2;
$name = "my_stream_name";

try {
    $result = $kinesisClient->createStream([
        'ShardCount' => $shardCount,
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

擷取資料串流

使用以下程式碼範例，取得現有資料串流的詳細資訊。根據預設，這會傳回連接到指定 Kinesis 資料串流的前 10 個碎片的相關資訊。請記得先`StreamStatus`檢查回應，再將資料寫入 Kinesis 資料串流。

若要擷取指定 Kinesis 資料串流的詳細資訊，請使用 DescribeStream 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $kinesisClient->describeStream([
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出連接到 Kinesis 的現有資料串流

列出所選 AWS 區域中來自 的前 10 AWS 帳戶 個資料串流。使用傳回的 `HasMoreStreams 判斷您的帳戶是否還有更多相關聯的串流。

若要列出 Kinesis 資料串流，請使用 ListStreams 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

try {
    $result = $kinesisClient->listStreams();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

將資料傳送至現有的資料串流

建立資料串流之後，使用以下範例傳送資料。傳送資料之前，使用 `DescribeStream` 檢查該資料的 `StreamStatus` 是否為作用中。

若要將單一資料記錄寫入 Kinesis 資料串流，請使用 PutRecord 操作。若要將最多 500 筆記錄寫入 Kinesis 資料串流，請使用 PutRecords 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-1'
]);
```

```
$name = "my_stream_name";
$content = '{"ticker_symbol":"QXZ", "sector":"HEALTHCARE", "change":-0.05,
 "price":84.51}';
$groupID = "input to a hash function that maps the partition key (and associated data)
 to a specific shard";

try {
    $result = $kinesisClient->PutRecord([
        'Data' => $content,
        'StreamName' => $name,
        'PartitionKey' => $groupID
    ]);
    print("<p>ShardID = " . $result["ShardId"] . "</p>");
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

刪除資料串流

本範例示範如何刪除資料串流。刪除資料串流也會一併刪除您已傳送至該資料串流的任何資料。作用中 Kinesis 資料串流會切換到 DELETING 狀態，直到串流刪除完成為止。若處於 DELETING 狀態，串流仍會繼續處理資料。

若要刪除 Kinesis 資料串流，請使用 DeleteStream 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
```

```
]);

$name = "my_stream_name";

try {
    $result = $kinesisClient->deleteStream([
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Kinesis Data Streams API 和第 3 適用於 PHP 的 AWS SDK 版管理資料碎片

Amazon Kinesis Data Streams 可讓您將即時資料傳送至端點。資料流程速率取決於串流中的碎片數目。

每秒寫入單個碎片的記錄可多達 1,000 筆。各碎片另有每秒 1 MiB 的上傳限制。用量將按碎片數目計算和收費，所以使用這些範例可控管您的串流資料容量及成本。

下列範例示範如何：

- 使用 ListShards 列出串流中的碎片。
- 使用 UpdateShardCount 增加或減少串流中的碎片數目。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon Kinesis Data Streams 的詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南。

列出資料串流碎片

列出特定串流中多達 100 個碎片的詳細資訊。

若要列出 Kinesis 資料串流中的碎片，請使用 ListShards 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $kinesisClient->ListShards([
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

新增更多資料串流碎片

如果您需要更多資料串流碎片，則可增加您目前的碎片數目。建議您將碎片數目增加為兩倍。如此會為您目前可用的每個碎片各建立一份複本以增加容量。您在 24 小時期間內可將碎片數目加倍的上限為兩次。

請記住，Kinesis Data Streams 用量的計費是以每個碎片計算，因此當需求減少時，建議您將碎片計數減少一半。移除碎片時，您可以將碎片總數僅縮減為目前碎片數目的一半。

若要更新 Kinesis 資料串流的碎片計數，請使用 UpdateShardCount 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$totalshards = 4;

try {
    $result = $kinesisClient->UpdateShardCount([
        'ScalingType' => 'UNIFORM_SCALING',
        'StreamName' => $name,
        'TargetShardCount' => $totalshards
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Firehose API 和 第 3 適用於 PHP 的 AWS SDK 版建立交付串流

Amazon Data Firehose 可讓您將即時資料傳送至其他 AWS 服務，包括 Amazon Kinesis Data Streams、Amazon S3、Amazon OpenSearch Service (OpenSearch Service) 和 Amazon Redshift，或傳送至 Splunk。為交付串流建立資料生產者，以便在您每次添加資料後將資料交付至設定的目的地。

下列範例示範如何：

- 使用 CreateDeliveryStream 建立交付串流。
- 使用 DescribeDeliveryStream 取得單一交付串流的詳細資訊。

- 使用 ListDeliveryStreams 列出您的交付串流。
- 使用 PutRecord 傳送資料至交付串流。
- 使用 DeleteDeliveryStream 刪除交付串流。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon Data Firehose 的詳細資訊，請參閱《Amazon Kinesis Data Firehose 開發人員指南》。

使用 Kinesis 資料串流建立交付串流

若要建立將資料放入現有 Kinesis 資料串流的交付串流，請使用 CreateDeliveryStream 操作。

這可讓開發人員將現有的 Kinesis 服務遷移至 Firehose。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$stream_type = "KinesisStreamAsSource";
$kinesis_stream = "arn:aws:kinesis:us-east-2:0123456789:stream/my_stream_name";
$role = "arn:aws:iam::0123456789:policy/Role";

try {
```

```
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
        'DeliveryStreamType' => $stream_type,
        'KinesisStreamSourceConfiguration' => [
            'KinesisStreamARN' => $kinesis_stream,
            'RoleARN' => $role,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Amazon S3 儲存貯體建立交付串流

若要建立將資料放入現有 Amazon S3 儲存貯體的交付串流，請使用 CreateDeliveryStream 操作。

提供目的地參數，如目的地參數一文所述。然後，確保您授予 Firehose 存取 Amazon S3 儲存貯體的權限，如授予 Kinesis Data Firehose 存取 Amazon S3 目的地中所述。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_S3_stream_name";
$stream_type = "DirectPut";
$s3bucket = 'arn:aws:s3:::bucket_name';
$s3Role = 'arn:aws:iam::0123456789:policy/Role';
```

```
try {
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
        'DeliveryStreamType' => $stream_type,
        'S3DestinationConfiguration' => [
            'BucketARN' => $s3bucket,
            'CloudWatchLoggingOptions' => [
                'Enabled' => false,
            ],
            'RoleARN' => $s3Role
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 OpenSearch Service 建立交付串流

若要建立將資料放入 OpenSearch Service 叢集的 Firehose 交付串流，請使用 CreateDeliveryStream 操作。

提供目的地參數，如目的地參數一文所述。請確定您授予 Firehose 存取 OpenSearch Service 叢集的權限，如授予 Kinesis Data Firehose 存取 Amazon ES 目的地中所述。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
```

```
]);

$name = "my_ES_stream_name";
$stream_type = "DirectPut";
$esDomainARN = 'arn:aws:es:us-east-2:0123456789:domain/Name';
$esRole = 'arn:aws:iam::0123456789:policy/Role';
$esIndex = 'root';
$esType = 'PHP_SDK';
$s3bucket = 'arn:aws:s3:::bucket_name';
$s3Role = 'arn:aws:iam::0123456789:policy/Role';

try {
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
        'DeliveryStreamType' => $stream_type,
        'ElasticsearchDestinationConfiguration' => [
            'DomainARN' => $esDomainARN,
            'IndexName' => $esIndex,
            'RoleARN' => $esRole,
            'S3Configuration' => [
                'BucketARN' => $s3bucket,
                'CloudWatchLoggingOptions' => [
                    'Enabled' => false,
                ],
                'RoleARN' => $s3Role,
            ],
            'TypeName' => $esType,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

擷取交付串流

若要取得現有 Firehose 交付串流的詳細資訊，請使用 DescribeDeliveryStream 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $firehoseClient->describeDeliveryStream([
        'DeliveryStreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出連接到 Kinesis Data Streams 的現有交付串流

若要列出傳送資料至 Kinesis Data Streams 的所有現有 Firehose 交付串流，請使用 ListDeliveryStreams 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
```

```
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

try {
    $result = $firehoseClient->listDeliveryStreams([
        'DeliveryStreamType' => 'KinesisStreamAsSource',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出將資料傳送至其他服務的現有交付串流 AWS

若要列出傳送資料至 Amazon S3、OpenSearch Service 或 Amazon Redshift 或 Splunk 的所有現有 Firehose 交付串流，請使用 ListDeliveryStreams 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

try {
    $result = $firehoseClient->listDeliveryStreams([
        'DeliveryStreamType' => 'DirectPut',
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

將資料傳送至現有的 Firehose 交付串流

若要透過 Firehose 交付串流將資料傳送至指定的目的地，請在建立 Firehose 交付串流後使用 PutRecord 操作。

將資料傳送至 Firehose 交付串流之前，請使用 `DescribeDeliveryStream`來查看交付串流是否處於作用中狀態。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$content = '{"ticker_symbol":"QXZ", "sector":"HEALTHCARE", "change":-0.05,
 "price":84.51}';

try {
    $result = $firehoseClient->putRecord([
        'DeliveryStreamName' => $name,
        'Record' => [
            'Data' => $content,
        ],
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

刪除 Firehose 交付串流

若要刪除 Firehose 交付串流，請使用 DeleteDeliveryStreams 操作。如此也會一併刪除您已傳送至該交付串流的任何資料。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $firehoseClient->deleteDeliveryStream([
        'DeliveryStreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS Elemental MediaConvert 使用第 3 適用於 PHP 的 AWS SDK 版的範例

AWS Elemental MediaConvert 是以檔案為基礎的影片轉碼服務，具有廣播級功能。您可以將其用於建立跨網際網路的廣播資產及隨選視訊 (VOD) 交付資產。如需詳細資訊，請參閱「AWS Elemental MediaConvert 使用者指南」。

的 PHP API 透過*AWS.MediaConvert*用戶端類別 AWS Elemental MediaConvert 公開。如需詳細資訊，請參閱 API 參考`Class: AWS.MediaConvert`中的 。

在 中建立和管理轉碼任務 AWS Elemental MediaConvert

在此範例中，您使用第 3 適用於 PHP 的 AWS SDK 版來呼叫 AWS Elemental MediaConvert 和建立轉碼任務。開始之前，您需要將輸入視訊上傳至您為輸入儲存體佈建的 Amazon S3 儲存貯體。如需支援的輸入視訊轉碼器和容器清單，請參閱AWS Elemental MediaConvert 《 使用者指南》中的支援的輸入轉碼器和容器。

下列範例示範如何：

- 在 中建立轉碼任務 AWS Elemental MediaConvert。CreateJob。
- 從 AWS Elemental MediaConvert 佇列取消轉碼任務。CancelJob
- 擷取已完成轉碼任務的 JSON。GetJob
- 擷取高達 20 個最近建立任務的 JSON 陣列。ListJobs

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

若要存取 MediaConvert 用戶端，請建立可讓您 AWS Elemental MediaConvert 存取輸入檔案的 IAM 角色，以及存放輸出檔案的 Amazon S3 儲存貯體。如需詳細資訊，請參閱《 使用者指南》中的設定 IAM 許可。 AWS Elemental MediaConvert

建立用戶端

適用於 PHP 的 AWS SDK 使用程式碼的區域建立 MediaConvert 用戶端來設定 。在這個範例中，區域設為 us-west-2。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\MediaConvert\MediaConvertClient;
```

範例程式碼

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);
```

定義簡單的轉碼任務

建立 JSON，定義轉碼任務參數。

這些參數是詳細的。您可以使用AWS Elemental MediaConvert 主控台在主控台中選擇您的任務設定，然後選擇任務區段底部的顯示任務 JSON，以產生 JSON 任務參數。此範例顯示簡單任務的 JSON。

範例程式碼

```
$jobSetting = [
    "OutputGroups" => [
        [
            "Name" => "File Group",
            "OutputGroupSettings" => [
                "Type" => "FILE_GROUP_SETTINGS",
                "FileGroupSettings" => [
                    "Destination" => "s3://OUTPUT_BUCKET_NAME/"
                ]
            ],
            "Outputs" => [
                [
                    "VideoDescription" => [
                        "ScalingBehavior" => "DEFAULT",
                        "TimecodeInsertion" => "DISABLED",
                        "AntiAlias" => "ENABLED",
                        "Sharpness" => 50,
                        "CodecSettings" => [
                            "Codec" => "H_264",
```

```
                            "H264Settings" => [
                                "InterlaceMode" => "PROGRESSIVE",
                                "NumberReferenceFrames" => 3,
                                "Syntax" => "DEFAULT",
                                "Softness" => 0,
                                "GopClosedCadence" => 1,
                                "GopSize" => 90,
                                "Slices" => 1,
                                "GopBReference" => "DISABLED",
                                "SlowPal" => "DISABLED",
                                "SpatialAdaptiveQuantization" => "ENABLED",
                                "TemporalAdaptiveQuantization" => "ENABLED",
                                "FlickerAdaptiveQuantization" => "DISABLED",
                                "EntropyEncoding" => "CABAC",
                                "Bitrate" => 5000000,
                                "FramerateControl" => "SPECIFIED",
                                "RateControlMode" => "CBR",
                                "CodecProfile" => "MAIN",
                                "Telecine" => "NONE",
                                "MinIInterval" => 0,
                                "AdaptiveQuantization" => "HIGH",
                                "CodecLevel" => "AUTO",
                                "FieldEncoding" => "PAFF",
                                "SceneChangeDetect" => "ENABLED",
                                "QualityTuningLevel" => "SINGLE_PASS",
                                "FramerateConversionAlgorithm" => "DUPLICATE_DROP",
                                "UnregisteredSeiTimecode" => "DISABLED",
                                "GopSizeUnits" => "FRAMES",
                                "ParControl" => "SPECIFIED",
                                "NumberBFramesBetweenReferenceFrames" => 2,
                                "RepeatPps" => "DISABLED",
                                "FramerateNumerator" => 30,
                                "FramerateDenominator" => 1,
                                "ParNumerator" => 1,
                                "ParDenominator" => 1
                            ]
                        ],
                        "AfdSignaling" => "NONE",
                        "DropFrameTimecode" => "ENABLED",
                        "RespondToAfd" => "NONE",
                        "ColorMetadata" => "INSERT"
                    ],
                    "AudioDescriptions" => [
                        [
```

```
                            "AudioTypeControl" => "FOLLOW_INPUT",
                            "CodecSettings" => [
                                "Codec" => "AAC",
                                "AacSettings" => [
                                    "AudioDescriptionBroadcasterMix" => "NORMAL",
                                    "RateControlMode" => "CBR",
                                    "CodecProfile" => "LC",
                                    "CodingMode" => "CODING_MODE_2_0",
                                    "RawFormat" => "NONE",
                                    "SampleRate" => 48000,
                                    "Specification" => "MPEG4",
                                    "Bitrate" => 64000
                                ]
                            ],
                            "LanguageCodeControl" => "FOLLOW_INPUT",
                            "AudioSourceName" => "Audio Selector 1"
                        ]
                    ],
                    "ContainerSettings" => [
                        "Container" => "MP4",
                        "Mp4Settings" => [
                            "CslgAtom" => "INCLUDE",
                            "FreeSpaceBox" => "EXCLUDE",
                            "MoovPlacement" => "PROGRESSIVE_DOWNLOAD"
                        ]
                    ],
                    "NameModifier" => "_1"
                ]
            ]
        ]
    ],
    "AdAvailOffset" => 0,
    "Inputs" => [
        [
            "AudioSelectors" => [
                "Audio Selector 1" => [
                    "Offset" => 0,
                    "DefaultSelection" => "NOT_DEFAULT",
                    "ProgramSelection" => 1,
                    "SelectorType" => "TRACK",
                    "Tracks" => [
                        1
                    ]
                ]
```

```
            ],
            "VideoSelector" => [
                "ColorSpace" => "FOLLOW"
            ],
            "FilterEnable" => "AUTO",
            "PsiControl" => "USE_PSI",
            "FilterStrength" => 0,
            "DeblockFilter" => "DISABLED",
            "DenoiseFilter" => "DISABLED",
            "TimecodeSource" => "EMBEDDED",
            "FileInput" => "s3://INPUT_BUCKET_AND_FILE_NAME"
        ]
    ],
    "TimecodeConfig" => [
        "Source" => "EMBEDDED"
    ]
];
```

建立任務。

建立任務參數 JSON 後，透過叫用 `AWS.MediaConvert service object` 和傳遞參數來呼叫 createJob 方法。回應資料中會傳回所建立任務的 ID。

範例程式碼

```
try {
    $result = $mediaConvertClient->createJob([
        "Role" => "IAM_ROLE_ARN",
        "Settings" => $jobSetting, //JobSettings structure
        "Queue" => "JOB_QUEUE_ARN",
        "UserMetadata" => [
            "Customer" => "Amazon"
        ],
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

擷取任務

使用呼叫 createjob 時傳回的 JobID，您可以取得 JSON 格式的最近任務詳細說明。

範例程式碼

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->getJob([
        'Id' => 'JOB_ID',
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

取消任務

使用呼叫 createjob 時傳回的 JobID，您可以取消仍在佇列中的任務。您無法取消已開始轉碼的任務。

範例程式碼

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->cancelJob([
        'Id' => 'JOB_ID', // REQUIRED The Job ID of the job to be cancelled.
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
```

```
}
```

列出最近的轉碼任務

建立參數 JSON，包括指定以 ASCENDING 或 DESCENDING 順序排序清單的值、要檢查的任務佇列 ARN，以及要包含的任務狀態。這會傳回最多 20 個任務。若要擷取後續 20 個最新任務，請使用結果傳回的 nextToken 字串。

範例程式碼

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->listJobs([
        'MaxResults' => 20,
        'Order' => 'ASCENDING',
        'Queue' => 'QUEUE_ARN',
        'Status' => 'SUBMITTED',
        // 'NextToken' => '<string>', //OPTIONAL To retrieve the twenty next most
 recent jobs
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用第 3 版的 Amazon S3 範例 適用於 PHP 的 AWS SDK

Amazon Simple Storage Service (Amazon S3) 是一種 Web 服務，可提供高度可擴展的雲端儲存。Amazon S3 提供易於使用的物件儲存，具有簡單的 Web 服務介面，可從 Web 上的任何位置儲存和擷取任意數量的資料。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

主題

- 使用第 3 版建立和使用 Amazon S3 儲存貯體 適用於 PHP 的 AWS SDK
- 使用第 3 版管理 Amazon S3 儲存貯體存取許可 適用於 PHP 的 AWS SDK
- 使用第 3 版設定 Amazon S3 儲存貯體 適用於 PHP 的 AWS SDK
- 搭配第 3 版使用 Amazon S3 分段上傳 適用於 PHP 的 AWS SDK
- 使用第 3 版的 Amazon S3 預先簽章 URL 適用於 PHP 的 AWS SDK
- 使用第 Amazon S3POSTs 適用於 PHP 的 AWS SDK
- 使用 Amazon S3 儲存貯體做為靜態 Web 主機搭配第 3 適用於 PHP 的 AWS SDK 版
- 使用 Amazon S3 儲存貯體政策搭配第 3 適用於 PHP 的 AWS SDK 版
- 使用 S3 存取點 ARNs 第 3 適用於 PHP 的 AWS SDK 版
- 搭配第 3 版使用 Amazon S3 多區域存取點 適用於 PHP 的 AWS SDK

使用第 3 版建立和使用 Amazon S3 儲存貯體 適用於 PHP 的 AWS SDK

下列範例示範如何：

- 使用 ListBuckets 傳回已驗證的請求發送者擁有的儲存貯體清單。
- 使用 CreateBucket 建立新的儲存貯體。
- 使用 PutObject 將物件新增到儲存貯體。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\S3\S3Client;
```

列出儲存貯體

使用下列程式碼建立 PHP 檔案。首先建立指定 AWS 區域和版本的 AWS.S3 用戶端服務。然後呼叫 `listBuckets`方法，將請求的寄件者擁有的所有 Amazon S3 儲存貯體傳回為儲存貯體結構陣列。

範例程式碼

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Listing all S3 Bucket
$buckets = $s3Client->listBuckets();
foreach ($buckets['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}
```

建立 儲存貯體

使用下列程式碼建立 PHP 檔案。首先建立指定 AWS 區域和版本的 AWS.S3 用戶端服務。然後以陣列呼叫 `createBucket` 方法做為參數。唯一的必要欄位是 ‘Bucket’ 鍵，此字串值是欲建立儲存貯體的名稱。不過，您可以使用「CreateBucketConfiguration」欄位指定 AWS 區域。如果成功，此方法會傳回儲存貯體的 ‘Location’ (位置)。

範例程式碼

```
function createBucket($s3Client, $bucketName)
{
    try {
        $result = $s3Client->createBucket([
            'Bucket' => $bucketName,
        ]);
        return 'The bucket\'s location is: ' .
            $result['Location'] . '. ' .
            'The bucket\'s effective URI is: ' .
```

```
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheBucket()
{
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2006-03-01'
    ]);

    echo createBucket($s3Client, 'my-bucket');
}

// Uncomment the following line to run this code in an AWS account.
// createTheBucket();
```

將物件放入儲存貯體

若要將檔案新增到您的新儲存貯體，請以下列程式碼建立 PHP 檔案。

由命令列執行此檔案，並以字串傳入您想上傳檔案的儲存貯體的名稱，接著是欲上傳的檔案所在完整檔案路徑。

範例程式碼

```
$USAGE = "\n" .
    "To run this example, supply the name of an S3 bucket and a file to\n" .
    "upload to it.\n" .
    "\n" .
    "Ex: php PutObject.php <bucketname> <filename>\n";

if (count($argv) <= 2) {
    echo $USAGE;
    exit();
}

$bucket = $argv[1];
$file_Path = $argv[2];
$key = basename($argv[2]);
```

```
try {
    //Create a S3Client
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-west-2',
        'version' => '2006-03-01'
    ]);
    $result = $s3Client->putObject([
        'Bucket' => $bucket,
        'Key' => $key,
        'SourceFile' => $file_Path,
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

使用第 3 版管理 Amazon S3 儲存貯體存取許可 適用於 PHP 的 AWS SDK

存取控制清單 (ACL) 是可用來管理儲存貯體與物件存取的其中一個資源類型存取政策選項。您可以使用 ACLs將基本讀取/寫入許可授予其他 AWS 帳戶。如需進一步了解，請參閱使用 ACL 管理存取。

下列範例將說明：

- 使用 GetBucketAcl 取得儲存貯體的存取控制政策。
- 使用 PutBucketAcl 憑藉 ACL 對儲存貯體設定許可。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

取得並設定存取控制清單政策

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;
```

範例程式碼

```
// Create a S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Gets the access control policy for a bucket
$bucket = 'amzn-s3-demo-bucket';
try {
    $resp = $s3Client->getBucketAcl([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving bucket ACL as follows: \n";
    var_dump($resp);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Sets the permissions on a bucket using access control lists (ACL).
$params = [
    'ACL' => 'public-read',
    'AccessControlPolicy' => [
        // Information can be retrieved from `getBucketAcl` response
        'Grants' => [
            [
                'Grantee' => [
                    'DisplayName' => '<string>',
                    'EmailAddress' => '<string>',
                    'ID' => '<string>',
                    'Type' => 'CanonicalUser',
                    'URI' => '<string>',
                ],
                'Permission' => 'FULL_CONTROL',
            ],
            // ...
        ],
        'Owner' => [
            'DisplayName' => '<string>',
```

```
            'ID' => '<string>',
        ],
    ],
    'Bucket' => $bucket,
];

try {
    $resp = $s3Client->putBucketAcl($params);
    echo "Succeed in setting bucket ACL.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

使用第 3 版設定 Amazon S3 儲存貯體 適用於 PHP 的 AWS SDK

跨來源資源分享 (CORS) 會定義一種方式，讓載入單一個網域的用戶端 Web 應用程式，能與不同網域中的資源互動。透過 Amazon S3 中的 CORS 支援，您可以使用 Amazon S3 建置豐富的用戶端 Web 應用程式，並選擇性地允許跨來源存取您的 Amazon S3 資源。

如需搭配 Amazon S3 儲存貯體使用 CORS 組態的詳細資訊，請參閱跨來源資源共用 (CORS)。

下列範例示範如何：

- 使用 GetBucketCors 取得儲存貯體的 CORS 組態。
- 使用 PutBucketCors 設定儲存貯體的 CORS 組態。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

取得 CORS 組態

使用下列程式碼建立 PHP 檔案。首先，建立一個 AWS.S3 用戶端服務，然後呼叫 `getBucketCors` 方法，並指定具有您所需要 CORS 組態的儲存貯體。

唯一需要的參數，就是所選取儲存貯體的名稱。如果儲存貯體目前具有 CORS 組態，Amazon S3 會將該組態傳回為 CORSRules 物件。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

範例程式碼

```
$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

try {
    $result = $client->getBucketCors([
        'Bucket' => $bucketName, // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

設定 CORS 組態

使用下列程式碼建立 PHP 檔案。首先，建立一個 AWS.S3 用戶端服務。接著呼叫 `putBucketCors` 方法並指定您要設定其 CORS 組態的儲存貯體，然後指定 CORSConfiguration 為 CORSRules JSON 物件。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

範例程式碼

```
$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

try {
    $result = $client->putBucketCors([
        'Bucket' => $bucketName, // REQUIRED
        'CORSConfiguration' => [ // REQUIRED
            'CORSRules' => [ // REQUIRED
                [
                    'AllowedHeaders' => ['Authorization'],
                    'AllowedMethods' => ['POST', 'GET', 'PUT'], // REQUIRED
                    'AllowedOrigins' => ['*'], // REQUIRED
                    'ExposeHeaders' => [],
                    'MaxAgeSeconds' => 3000
                ],
            ],
        ]
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

搭配第 3 版使用 Amazon S3 分段上傳 適用於 PHP 的 AWS SDK

透過單次 `PutObject` 操作，您最多可上傳大小 5 GB 的物件。不過，如果使用分段上傳方法 (例如，`CreateMultipartUpload`、`UploadPart`、`CompleteMultipartUpload`、`AbortMultipartUplo
您可以上傳大小從 5 MB 到 5 TB 的物件。

下列範例將說明：

- 使用 ObjectUploader 將物件上傳至 Amazon S3。
- 使用 MultipartUploader 為 Amazon S3 物件建立分段上傳。
- 使用 ObjectCopier 將物件從一個 Amazon S3 位置複製到另一個位置。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

物件上傳程式

如果您不確定 `PutObject`或 `MultipartUploader` 是否最適合任務，請使用 `ObjectUploader`。會根據承載大小`MultipartUploader`，使用 `PutObject`或 將大型檔案`ObjectUploader`上傳至 Amazon S3。

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\ObjectUploader;
use Aws\S3\S3Client;
```

範例程式碼

```
// Create an S3Client.
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2006-03-01'
]);

$bucket = 'your-bucket';
$key = 'my-file.zip';

// Use a stream instead of a file path.
$source = fopen('/path/to/large/file.zip', 'rb');

$uploader = new ObjectUploader(
    $s3Client,
    $bucket,
    $key,
    $source
);
```

```
do {
    try {
        $result = $uploader->upload();
        if ($result["@metadata"]["statusCode"] == '200') {
            print('<p>File successfully uploaded to ' . $result["ObjectURL"] . '.</
p>');
        }
        print($result);
        // If the SDK chooses a multipart upload, try again if there is an exception.
        // Unlike PutObject calls, multipart upload calls are not automatically
 retried.
    } catch (MultipartUploadException $e) {
        rewind($source);
        $uploader = new MultipartUploader($s3Client, $source, [
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));

fclose($source);
```

組態

`ObjectUploader` 物件建構函式接受下列引數：

`$client`

用來執行傳輸的 `Aws\ClientInterface` 物件。這應該是 `Aws\S3\S3Client` 的執行個體。

`$bucket`

(`string`，必要) 做為物件上傳目的地的儲存貯體的名稱。

`$key`

(`string`，必要) 待上傳的物件所要使用的金鑰。

`$body`

(`mixed`，必要) 要上傳的物件資料。可以是 `StreamInterface`、PHP 串流資源或要上傳的資料字串。

`$acl`

(`string`) 針對待上傳的物件，所要設定的存取控制清單 (ACL)。物件預設為私有。

$options

分段上傳的組態選項的關聯式陣列。下列組態選項為有效：

add_content_md5

(bool) 設為 true 可自動計算上傳的 MD5 檢查總和。

mup_threshold

(int，預設：int(16777216)) 檔案大小的位元組數。如果檔案大小超過此限制，則會使用分段上傳。

before_complete

(callable) 要在 CompleteMultipartUpload 操作之前呼叫的回呼函式。回呼應具有類似 的函數簽章：function (Aws\Command $command) {...}。如需可新增至CommandInterface物件的參數，請參閱 CompleteMultipartUpload API 參考。

before_initiate

(callable) 要在 CreateMultipartUpload 操作之前呼叫的回呼函式。回呼應具有類似 的函數簽章：function (Aws\Command $command) {...}。如果檔案大小超過mup_threshold值，軟體開發套件會叫用此回呼。如需可新增至CommandInterface物件的參數，請參閱 CreateMultipartUpload API 參考。

before_upload

(callable) 回呼在 PutObject或 UploadPart操作之前叫用。回呼應具有類似 的函數簽章：function (Aws\Command $command) {...}。如果檔案大小小於或等於mup_threshold值，軟體開發套件會叫用此回呼。如需可套用至PutObject請求的參數，請參閱 PutObject API 參考。如需套用至UploadPart請求的參數，請參閱 UploadPart API 參考。SDK 會忽略任何不適用於CommandInterface物件所代表之操作的參數。

concurrency

(int，預設：int(3)) 分段上傳期間所允許的並行 UploadPart 操作的數目上限。

part_size

(int，預設：int(5242880)) 進行分段上傳時所使用的分段大小 (位元組)。此值必須介於 5 MB 到 5 GB 之間。

state

(Aws\Multipart\UploadState) 代表分段上傳狀態的物件，用來恢復先前的上傳作業。提供此選項時，會忽略 $bucket和 $key引數和 part_size選項。

params

關聯陣列，提供每個子命令的組態選項。例如：

```
new ObjectUploader($bucket, $key, $body, $acl, ['params' => ['CacheControl'
 => <some_value>])
```

MultipartUploader

分段上傳是專為改善較大型物件上傳的體驗所設計。這些方法可讓您以任意順序，個別同時上傳物件的各部分。

建議 Amazon S3 客戶針對大於 100 MB 的物件使用分段上傳。

MultipartUploader 物件

軟體開發套件具有特殊的 `MultipartUploader` 物件，可簡化分段上傳的流程。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

範例程式碼

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Use multipart upload
$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
```

```
]);

try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}\n";
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . "\n";
}
```

上傳程式會根據隨附的原始碼和組態，建立分段資料的產生器，並嘗試上傳所有部分。如果某些部分上傳失敗，上傳程式會持續上傳之後的部分，直到讀取完整個原始碼的資料。之後，上傳工具會嘗試上傳失敗的部分，或擲出包含與無法上傳之部分相關資訊的例外狀況。

自訂分段上傳

您可以透過將回呼函式傳遞至其建構函式，來針對多段上傳程式所執行的 `CreateMultipartUpload`、`UploadPart` 與 `CompleteMultipartUpload` 操作，設定自訂選項。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

範例程式碼

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Customizing a multipart upload
$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
```

```
    'before_initiate' => function (Command $command) {
        // $command is a CreateMultipartUpload operation
        $command['CacheControl'] = 'max-age=3600';
    },
    'before_upload' => function (Command $command) {
        // $command is an UploadPart operation
        $command['RequestPayer'] = 'requester';
    },
    'before_complete' => function (Command $command) {
        // $command is a CompleteMultipartUpload operation
        $command['RequestPayer'] = 'requester';
    },
]);
```

部分上傳之間的手動垃圾回收

如果您達到大型上傳的記憶體限制，這可能是因為在達到記憶體限制時，PHP 廢棄項目收集器尚未收集軟體開發套件產生的循環參考。在操作之間手動叫用集合演算法會允許在達到該限制前收集循環。以下範例會在每次分段上傳前，使用回呼叫用集合演算法。請注意，叫用廢棄項目收集器不會伴隨效能成本，最佳使用方式將取決於您的使用案例和環境。

```
$uploader = new MultipartUploader($client, $source, [
   'bucket' => 'your-bucket',
   'key' => 'your-key',
   'before_upload' => function(\Aws\Command $command) {
      gc_collect_cycles();
   }
]);
```

從錯誤中復原

在分段上傳過程中發生錯誤時，會丟出 `MultipartUploadException`。此例外狀況提供了對 `UploadState` 物件的存取，而此物件包含關於分段上傳進度的資訊。`UploadState` 可用來恢復無法完成的上傳作業。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
```

```
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

範例程式碼

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

//Recover from errors
do {
    try {
        $result = $uploader->upload();
    } catch (MultipartUploadException $e) {
        $uploader = new MultipartUploader($s3Client, $source, [
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));

//Abort a multipart upload if failed
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

透過 UploadState 恢復上傳的動作時，會嘗試上傳尚未上傳的部分。狀態物件會追蹤缺少的部分，即使這些部分並不連續。上傳程式會針對所提供的原始碼檔案，讀取或搜尋屬於仍需上傳部分的位元組範圍，

UploadState 物件可序列化，因此您也可以在不同的程序中恢復上傳。即使不是在處理例外狀況時，您也可以透過呼叫 $uploader->getState()，來取得 UploadState 物件。

> **Important**
>
> 做為資源傳遞到 MultipartUploader 的串流，在上傳之前不會自動跳回開頭。如果您使用的是串流，而非類似於先前範例中迴圈內的檔案路徑，則請重設 catch 區塊中的 $source 變數。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

範例程式碼

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Using stream instead of file path
$source = fopen('/path/to/large/file.zip', 'rb');
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

do {
    try {
        $result = $uploader->upload();
    } catch (MultipartUploadException $e) {
        rewind($source);
        $uploader = new MultipartUploader($s3Client, $source, [
```

```
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));
fclose($source);
```

中止分段上傳

您可以擷取 UploadState 物件中包含的 UploadId ，並將它傳遞至 abortMultipartUpload，以中止分段上傳。

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

非同步分段上傳

呼叫 upload() 的 MultipartUploader 是封鎖請求。如果是在非同步的情境中作業，您可以取得分段上傳的 promise 物件。

```
require 'vendor/autoload.php';

use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

範例程式碼

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

$source = '/path/to/large/file.zip';
```

```
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

$promise = $uploader->promise();
```

組態

`MultipartUploader` 物件建構函式接受下列引數：

$client

用來執行傳輸的 `Aws\ClientInterface` 物件。這應該是 `Aws\S3\S3Client` 的執行個體。

$source

正在上傳的原始碼資料。這可以是路徑或 URL (例如，`/path/to/file.jpg`)、資源控制代碼 (例如，`fopen('/path/to/file.jpg', 'r')`)，或是 PSR-7 stream 的執行個體。

$config

分段上傳的組態選項的關聯式陣列。

下列組態選項為有效：

acl

(`string`) 針對待上傳的物件，所要設定的存取控制清單 (ACL)。物件預設為私有。

before_complete

(`callable`) 要在 `CompleteMultipartUpload` 操作之前呼叫的回呼函式。回呼函式應具有像是 `function (Aws\Command $command) {...}` 的函式簽章。

before_initiate

(`callable`) 要在 `CreateMultipartUpload` 操作之前呼叫的回呼函式。回呼函式應具有像是 `function (Aws\Command $command) {...}` 的函式簽章。

before_upload

(`callable`) 要在任何 `UploadPart` 操作之前呼叫的回呼函式。回呼函式應具有像是 `function (Aws\Command $command) {...}` 的函式簽章。

bucket

(string，必要) 做為物件上傳目的地的儲存貯體的名稱。

concurrency

(int，預設：int(5)) 分段上傳期間所允許的並行 UploadPart 操作的數目上限。

key

(string，必要) 待上傳的物件所要使用的金鑰。

part_size

(int，預設：int(5242880)) 進行分段上傳時所使用的分段大小 (位元組)。這必須介於 5 MB 到 5 GB 之間 (含)。

state

(Aws\Multipart\UploadState) 代表分段上傳狀態的物件，用來恢復先前的上傳作業。提供此選項時，會略過 bucket、key 和 part_size 選項。

add_content_md5

(boolean) 設為 true 可自動計算上傳的 MD5 檢查總和。

params

關聯陣列，提供每個子命令的組態選項。例如：

```
new MultipartUploader($client, $source, ['params' => ['CacheControl'
 => <some_value>]])
```

分段副本

適用於 PHP 的 AWS SDK 也包含物件MultipartCopy，其使用方式與 類似MultipartUploader，但設計用於在 Amazon S3 內複製大小介於 5 GB 到 5 TB 之間的物件。

```
require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartCopy;
use Aws\S3\S3Client;
```

範例程式碼

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Copy objects within S3
$copier = new MultipartCopy($s3Client, '/bucket/key?versionId=foo', [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

try {
    $result = $copier->copy();
    echo "Copy complete: {$result['ObjectURL']}\n";
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . "\n";
}
```

使用第 3 版的 Amazon S3 預先簽章 URL 適用於 PHP 的 AWS SDK

您可以將必要資訊當做查詢字串參數來傳遞，以驗證特定類型的要求，而不使用授權 HTTP 標頭。這有助於啟用直接第三方瀏覽器存取您的私有 Amazon S3 資料，而無需代理請求。想法是建構「預先簽署」請求，並將其編碼為另一個使用者可以使用的 URL。此外，您可以透過指定過期時間，來限制預先簽章的要求。

建立 HTTP GET 請求的預先簽章 URL

下列程式碼範例示範如何使用適用於 PHP 的 開發套件，為 HTTP GET 請求建立預先簽章的 URL。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

```
$s3Client = new S3Client([
    'region' => 'us-west-2',
]);

// Supply a CommandInterface object and an expires parameter to the
 `createPresignedRequest` method.
$request = $s3Client->createPresignedRequest(
    $s3Client->getCommand('GetObject', [
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'demo-key',
    ]),
    '+1 hour'
);

// From the resulting RequestInterface object, you can get the URL.
$presignedUrl = (string) $request->getUri();

echo $presignedUrl;
```

方法的 API 參考`createPresignedRequest`提供更多詳細資訊。

其他人可以使用 `$presignedUrl`值，在下一個小時內擷取物件。當提出 HTTP GET 請求時，例如使用瀏覽器時，它會向 S3 服務顯示呼叫來自建立預先簽章 URL 的使用者。

建立 HTTP PUT 請求的預先簽章 URL

下列程式碼範例示範如何使用適用於 PHP 的 開發套件，為 HTTP PUT 請求建立預先簽章的 URL。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$s3Client = new S3Client([
    'region' => 'us-west-2',
]);

$request = $s3Client->createPresignedRequest(
    $s3Client->getCommand('PutObject', [
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'demo-key',
```

```
    ]),
    '+1 hour'
);

// From the resulting RequestInterface object, you can get the URL.
$presignedUrl = (string) $request->getUri();
```

其他人現在可以在 HTTP PUT 請求中使用預先簽章的 URL 來上傳檔案：

```
use GuzzleHttp\Psr7\Request;
use GuzzleHttp\Psr7\Response;

// ...

function uploadWithPresignedUrl($presignedUrl, $filePath, $s3Client): ?Response
{
    // Get the HTTP handler from the S3 client.
    $handler = $s3Client->getHandlerList()->resolve();

    // Create a stream from the file.
    $fileStream = new Stream(fopen($filePath, 'r'));

    // Create the request.
    $request = new Request(
        'PUT',
        $presignedUrl,
        [
            'Content-Type' => mime_content_type($filePath),
            'Content-Length' => filesize($filePath)
        ],
        $fileStream
    );

    // Send the request using the handler.
    try {
        $promise = $handler($request, []);
        $response = $promise->wait();
        return $response;
    } catch (Exception $e) {
        echo "Error uploading file: " . $e->getMessage() . "\n";
        return null;
    }
}
```

使用第 Amazon S3POSTs 適用於 PHP 的 AWS SDK

與預先簽章URLs 類似，預先簽章POSTs 可讓您提供寫入存取權給使用者，而不需提供 AWS 登入資料。預先簽章的 POST 表單可以由 AwsS3PostObjectV4 的執行個體來協助建立。

下列範例示範如何：

- 使用 PostObjectV4 取得 S3 物件 POST 上傳的資料。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

> **Note**
>
> `PostObjectV4` 不適用於來自 的登入資料 AWS IAM Identity Center。

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立 PostObjectV4

要建立 `PostObjectV4` 執行個體，您必須提供下列項目：

- `Aws\S3\S3Client` 執行個體
- 儲存貯體
- 表單輸入欄位的關聯陣列
- 政策條件陣列（請參閱《Amazon Simple Storage Service 使用者指南》中的政策建構)
- 政策的過期時間字串 (選擇性，預設為一小時)。

匯入

```
require '../vendor/autoload.php';

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;
```

範例程式碼

```
require '../vendor/autoload.php';

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;

$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-1',
]);
$bucket = 'amzn-s3-demo-bucket10';
$starts_with = 'user/eric/';
$client->listBuckets();

// Set defaults for form input fields.
$formInputs = ['acl' => 'public-read'];

// Construct an array of conditions for policy.
$options = [
    ['acl' => 'public-read'],
    ['bucket' => $bucket],
    ['starts-with', '$key', $starts_with],
];

// Set an expiration time (optional).
$expires = '+2 hours';

$postObject = new PostObjectV4(
    $client,
    $bucket,
    $formInputs,
    $options,
    $expires
);

// Get attributes for the HTML form, for example, action, method, enctype.
$formAttributes = $postObject->getFormAttributes();

// Get attributes for the HTML form values.
$formInputs = $postObject->getFormInputs();
?>
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>PHP</title>
</head>
<body>
<form action="<?php echo $formAttributes['action'] ?>" method="<?php echo
 $formAttributes['method'] ?>"
      enctype="<?php echo $formAttributes['enctype'] ?>">
    <label id="key">
        <input hidden type="text" name="key" value="<?php echo $starts_with ?><?php
 echo $formInputs['key'] ?>"/>
    </label>
    <h3>$formInputs:</h3>
    acl: <label id="acl">
        <input readonly type="text" name="acl" value="<?php echo $formInputs['acl'] ?
>"/>
    </label><br/>
    X-Amz-Credential: <label id="credential">
        <input readonly type="text" name="X-Amz-Credential" value="<?php echo
 $formInputs['X-Amz-Credential'] ?>"/>
    </label><br/>
    X-Amz-Algorithm: <label id="algorithm">
        <input readonly type="text" name="X-Amz-Algorithm" value="<?php echo
 $formInputs['X-Amz-Algorithm'] ?>"/>
    </label><br/>
    X-Amz-Date: <label id="date">
        <input readonly type="text" name="X-Amz-Date" value="<?php echo $formInputs['X-
Amz-Date'] ?>"/>
    </label><br/><br/><br/>
    Policy: <label id="policy">
        <input readonly type="text" name="Policy" value="<?php echo
 $formInputs['Policy'] ?>"/>
    </label><br/>
    X-Amz-Signature: <label id="signature">
        <input readonly type="text" name="X-Amz-Signature" value="<?php echo
 $formInputs['X-Amz-Signature'] ?>"/>
    </label><br/><br/>
    <h3>Choose file:</h3>
    <input type="file" name="file"/> <br/><br/>
    <h3>Upload file:</h3>
    <input type="submit" name="submit" value="Upload to Amazon S3"/>
</form>
</body>
```

```
</html>
```

使用 Amazon S3 儲存貯體做為靜態 Web 主機搭配第 3 適用於 PHP 的 AWS SDK 版

您可以在 Amazon S3 上託管靜態網站。若要進一步了解，請參閱在 Amazon S3 上託管靜態網站。

下列範例將說明：

- 使用 GetBucketWebsite 取得儲存貯體的網站組態。
- 使用 PutBucketWebsite 設定儲存貯體的網站組態。
- 使用 DeleteBucketWebsite 從儲存貯體移除網站組態。

第 3 適用於 PHP 的 AWS SDK 版的所有範例程式碼都可在 GitHub 上取得。

登入資料

在執行範例程式碼之前，請設定您的 AWS 登入資料。請參閱第 3 適用於 PHP 的 AWS SDK 版的登入資料。

取得、設定和刪除儲存貯體的網站組態

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

範例程式碼

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Retrieving the Bucket Website Configuration
$bucket = 'my-s3-bucket';
try {
    $resp = $s3Client->getBucketWebsite([
```

```
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving website configuration for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Setting a Bucket Website Configuration
$params = [
    'Bucket' => $bucket,
    'WebsiteConfiguration' => [
        'ErrorDocument' => [
            'Key' => 'foo',
        ],
        'IndexDocument' => [
            'Suffix' => 'bar',
        ],
    ]
];

try {
    $resp = $s3Client->putBucketWebsite($params);
    echo "Succeed in setting bucket website configuration.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deleting a Bucket Website Configuration
try {
    $resp = $s3Client->deleteBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Amazon S3 儲存貯體政策搭配第 3 適用於 PHP 的 AWS SDK 版

您可以使用儲存貯體政策將許可授予 Amazon S3 資源。如需進一步了解，請參閱使用儲存貯體政策和使用者政策。

下列範例將說明：

- 使用 GetBucketPolicy 傳回所指定儲存貯體的政策。
- 使用 PutBucketPolicy 取代儲存貯體的政策。
- 使用 DeleteBucketPolicy 從儲存貯體刪除政策。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

取得、刪除和取代儲存貯體上的政策

匯入

```
require "vendor/autoload.php";

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
```

範例程式碼

```
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

$bucket = 'my-s3-bucket';

// Get the policy of a specific bucket
try {
    $resp = $s3Client->getBucketPolicy([
```

```
        'Bucket' => $bucket
    ]);
    echo "Succeed in receiving bucket policy:\n";
    echo $resp->get('Policy');
    echo "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deletes the policy from the bucket
try {
    $resp = $s3Client->deleteBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy of bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Replaces a policy on the bucket
try {
    $resp = $s3Client->putBucketPolicy([
        'Bucket' => $bucket,
        'Policy' => 'foo policy',
    ]);
    echo "Succeed in put a policy on bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

使用 S3 存取點 ARNs 第 3 適用於 PHP 的 AWS SDK 版

S3 導入存取點，這是一種與 S3 儲存貯體互動的新方式。存取點可以套用唯一的政策和組態，而不是直接套用至儲存貯體。 適用於 PHP 的 AWS SDK 可讓您將儲存貯體欄位中ARNs 用於 API 操作，而不是明確指定儲存貯體名稱。如需 S3 存取點和 ARNs運作方式的詳細資訊，請參閱此處。下列範例示範如何：

- 將 GetObject 與存取點 ARN 搭配使用，從儲存貯體擷取物件。
- 將 PuTOObject 與存取點 ARN 搭配使用，將物件新增至儲存貯體。
- 將 S3 用戶端設定為使用 ARN 區域，而不是用戶端區域。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

匯入

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

取得物件

首先建立指定 AWS 區域和版本的 AWS.S3 用戶端服務。然後使用您的金鑰和 `Bucket` 欄位中的 S3 存取點 ARN 呼叫該 `getObject` 方法，該方法將從與該存取點關聯的儲存貯體中獲取物件。

範例程式碼

```
$s3 = new S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
]);
$result = $s3->getObject([
    'Bucket' => 'arn:aws:s3:us-west-2:123456789012:accesspoint:endpoint-name',
    'Key' => 'MyKey'
]);
```

將物件放入儲存貯體

首先建立指定 AWS 區域和版本的 AWS.S3 用戶端服務。然後使 `putObject` 用所需的金鑰、主體或來源檔案和 `Bucket` 欄位中的 S3 存取點 ARN 呼叫該方法，這會將物件放入與該存取點關聯的儲存貯體中。

範例程式碼

```
$s3 = new S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
]);
$result = $s3->putObject([
    'Bucket' => 'arn:aws:s3:us-west-2:123456789012:accesspoint:endpoint-name',
    'Key' => 'MyKey',
    'Body' => 'MyBody'
]);
```

將 S3 用戶端設定為使用 ARN 區域而不是用戶端區域

在 S3 用戶端操作中使用 S3 存取點 ARN 時，預設情況下，用戶端將確保 ARN 區域與用戶端區域相符，如果不相符，則擲回例外狀況。您可以將 `use_arn_region` 組態選項設定為 `true`，將此行為變更為透過用戶端區域接受 ARN 區域。依預設，選項會設定為 `false`。

範例程式碼

```
$s3 = new S3Client([
    'version'        => 'latest',
    'region'         => 'us-west-2',
    'use_arn_region' => true
]);
```

用戶端還將按照以下優先順序檢查環境變數和組態檔案選項：

1. 用戶端選項 `use_arn_region`，如上例所示。
2. 環境變數 `AWS_S3_USE_ARN_REGION`

```
export AWS_S3_USE_ARN_REGION=true
```

1. AWS 共用組態檔案中`s3_use_arn_region`的組態變數（預設為）`~/.aws/config`。

```
[default]
s3_use_arn_region = true
```

搭配第 3 版使用 Amazon S3 多區域存取點 適用於 PHP 的 AWS SDK

Amazon Simple Storage Service (S3) 多區域存取點提供全域端點，用於路由 Amazon S3 請求流量 AWS 區域。

您可以使用適用於 PHP 的 SDK、另一個 AWS SDK、S3 主控台或 AWS CLI 來建立多區域存取點，

> **Important**
>
> 若要搭配適用於 PHP 的 SDK 使用多區域存取點，您的 PHP 環境必須安裝AWS 通用執行期 (AWS CRT) 延伸模組。

當您建立多區域存取點時，Amazon S3 會產生格式如下的 Amazon Resource Name (ARN)：

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

您可以使用產生的 ARN 取代 `getObject()`和 `putObject()`方法的儲存貯體名稱。

```
<?php
require './vendor/autoload.php';

use Aws\S3\S3Client;

// Assign the Multi-Region Access Point to a variable and use it place of a bucket
 name.
$mrap = 'arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap';
$key = 'my-key';

$s3Client = new S3Client([
    'region' => 'us-east-1'
]);

$s3Client->putObject([
    'Bucket' => $mrap,
    'Key' => $key,
    'Body' => 'Hello World!'
]);

$result = $s3Client->getObject([
    'Bucket' => $mrap,
    'Key' => $key
]);
```

```
echo $result['Body'] . "\n";

// Clean up.
$result = $s3Client->deleteObject([
    'Bucket' => $mrap,
    'Key' => $key
]);

$s3Client->waitUntil('ObjectNotExists', ['Bucket' => $mrap, 'Key' => $key]);

echo "Object deleted\n";
```

使用 Secrets Manager API 和 第 3 適用於 PHP 的 AWS SDK 版管理秘密

AWS Secrets Manager 會存放和管理共用秘密，例如密碼、API 金鑰和資料庫登入資料。透過 Secrets Manager 服務，開發人員可以將已部署程式碼中的硬式編碼登入資料取代為對 Secrets Manager 的內嵌呼叫。

Secrets Manager 原生支援 Amazon Relational Database Service (Amazon RDS) 資料庫的自動排程登入資料輪換，以提高應用程式安全性。Secrets Manager 也可以使用 無縫輪換其他資料庫和第三方服務的秘密 AWS Lambda ，以實作服務特定的詳細資訊。

下列範例示範如何：

- 使用 CreateSecret 建立秘密。
- 使用 GetSecretValue 擷取秘密。
- 使用 ListSecrets 列出 Secrets Manager 存放的所有秘密。
- 使用 DescribeSecret 取得特定秘密的詳細資訊。
- 使用 PutSecretValue 更新指定的秘密。
- 使用 RotateSecret 設定秘密輪換。
- 使用 DeleteSecret 將秘密標為刪除。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

在 Secrets Manager 中建立秘密

若要在 Secrets Manager 中建立秘密，請使用 CreateSecret 操作。

在這個範例中，使用者名稱和密碼的存放格式為 JSON 字串。

匯入

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
$description = '<<Description>>';
try {
    $result = $client->createSecret([
        'Description' => $description,
        'Name' => $secretName,
        'SecretString' => $secret,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

從 Secrets Manager 擷取秘密

若要擷取存放在 Secrets Manager 中的秘密值，請使用 GetSecretValue 操作。

在下列範例中， secret 是包含儲存值的字串。如果 的值username是 ，<<USERNAME>>而 的值password是 <<PASSWORD>>，則 的輸出secret是：

```
{"username":"<<USERNAME>>","password":"<<PASSWORD>>"}
```

使用 json_decode($secret, true) 存取陣列值。

匯入

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-east-1',
]);

$secretName = 'MySecretName';

try {
    $result = $client->getSecretValue([
        'SecretId' => $secretName,
    ]);
} catch (AwsException $e) {
    $error = $e->getAwsErrorCode();
    if ($error == 'DecryptionFailureException') {
        // Secrets Manager can't decrypt the protected secret text using the provided
 AWS KMS key.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InternalServiceErrorException') {
        // An error occurred on the server side.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
```

```
    if ($error == 'InvalidParameterException') {
        // You provided an invalid value for a parameter.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InvalidRequestException') {
        // You provided a parameter value that is not valid for the current state of
 the resource.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'ResourceNotFoundException') {
        // We can't find the resource that you asked for.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
}
// Decrypts secret using the associated KMS CMK.
// Depending on whether the secret is a string or binary, one of these fields will be
 populated.
if (isset($result['SecretString'])) {
    $secret = $result['SecretString'];
} else {
    $secret = base64_decode($result['SecretBinary']);
}
print $secret;
$secretArray = json_decode($secret, true);
$username = $secretArray['username'];
$password = $secretArray['password'];

// Your code goes here;
```

列出存放在 Secrets Manager 中的秘密

取得 Secrets Manager 使用 ListSecrets 操作存放的所有秘密清單。

匯入

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

try {
    $result = $client->listSecrets([
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

擷取秘密的詳細資訊

存放的秘密包含與輪換規則相關的中繼資料、前次遭存取或變更的時間、使用者建立的標籤，以及 Amazon Resource Name (ARN)。若要取得存放在 Secrets Manager 中指定秘密的詳細資訊，請使用 DescribeSecret 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';
```

```
try {
    $result = $client->describeSecret([
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

更新機密值

若要在 Secrets Manager 中存放新的加密秘密值，請使用 PutSecretValue 操作。

這會建立秘密的新版本。如果某個版本的秘密已存在，請在 `AWSCURRENT` 中新增含該值的 `VersionStages` 參數，以確保擷取該值時所用的是新的值。

匯入

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
try {
    $result = $client->putSecretValue([
        'SecretId' => $secretName,
        'SecretString' => $secret,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

將值輪換到 Secrets Manager 中的現有秘密

若要輪換存放在 Secrets Manager 中現有秘密的值，請使用 Lambda 輪換函數和 RotateSecret 操作。

開始之前，請建立 Lambda 函數來輪換秘密。Code AWS Sample Catalog 目前包含數個用於輪換 Amazon RDS 資料庫登入資料的 Lambda 程式碼範例。

> **Note**
>
> 如需輪換秘密的詳細資訊，請參閱 AWS Secrets Manager 《使用者指南》中的輪換您的 AWS Secrets Manager 秘密。

設定 Lambda 函數之後，請設定新的秘密輪換。

匯入

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';
$lambda_ARN = 'arn:aws:lambda:us-
west-2:123456789012:function:MyTestDatabaseRotationLambda';
$rules = ['AutomaticallyAfterDays' => 30];

try {
```

```
    $result = $client->rotateSecret([
        'RotationLambdaARN' => $lambda_ARN,
        'RotationRules' => $rules,
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

輪換設定好時，您可以使用 RotateSecret 操作來實作輪換。

匯入

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';

try {
    $result = $client->rotateSecret([
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

從 Secrets Manager 刪除秘密

若要從 Secrets Manager 移除指定的秘密，請使用 DeleteSecret 操作。為避免不小心刪除秘密，會自動將 DeletionDate 戳記新增到秘密，來指定您可以反轉刪除的復原時間時段。如果未指定復原時段的時間，則預設的時間為 30 天。

匯入

```
require 'vendor/autoload.php';

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

範例程式碼

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';

try {
    $result = $client->deleteSecret([
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

相關資訊

這些 適用於 PHP 的 AWS SDK 範例使用 AWS Secrets Manager API 參考中的下列 REST 操作：

- CreateSecret
- GetSecretValue

- ListSecrets
- DescribeSecret
- PutSecretValue
- RotateSecret
- DeleteSecret

如需使用 的詳細資訊 AWS Secrets Manager，請參閱 AWS Secrets Manager 使用者指南。

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon SES 範例

Amazon Simple Email Service (Amazon SES) 是一種電子郵件平台，可讓您使用自己的電子郵件地址和網域，以簡單、節省成本的方式來傳送和接收電子郵件。如需 Amazon SES 的詳細資訊，請參閱《Amazon SES 開發人員指南》。

AWS 提供兩種版本的 Amazon SES 服務，相對地，適用於 PHP 的 SDK 提供兩種版本的用戶端：SesClient 和 SesV2Client。雖然呼叫方法或結果的方式可能不同，但用戶端的功能在許多情況下會重疊。這兩個 APIs 也提供獨佔功能，因此您可以使用這兩個用戶端來存取所有功能。

本節中的範例都使用原始 `SesClient`。

第 3 適用於 PHP 的 AWS SDK 版的所有範例程式碼都可在 GitHub 上取得。

主題

- 使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版驗證電子郵件身分
- 使用 Amazon SES API 和 第 3 適用於 PHP 的 AWS SDK 版建立自訂電子郵件範本
- 使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版管理電子郵件篩選條件
- 使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版建立和管理電子郵件規則
- 使用 Amazon SES API 和 第 3 適用於 PHP 的 AWS SDK 版監控您的傳送活動
- 使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版授權寄件者

使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版驗證電子郵件身分

當您第一次開始使用 Amazon Simple Email Service (Amazon SES) 帳戶時，所有寄件者和收件人都必須 AWS 在您傳送電子郵件的相同區域中進行驗證。如需如何傳送電子郵件的詳細資訊，請參閱使用 Amazon SES 傳送電子郵件。

下列範例示範如何：

- 使用 VerifyEmailIdentity 驗證電子郵件地址。
- 使用 VerifyDomainIdentity 驗證電子郵件網域。
- 使用 ListIdentities 列出所有電子郵件地址。
- 使用 ListIdentities 列出所有電子郵件網域。
- 使用 DeleteIdentity 移除電子郵件地址。
- 使用 DeleteIdentity 移除電子郵件網域。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon SES 的詳細資訊，請參閱《Amazon SES 開發人員指南》。

驗證電子郵件地址

Amazon SES 只能從已驗證的電子郵件地址或網域傳送電子郵件。透過驗證電子郵件地址，您可以證明您是該地址的擁有者，並希望允許 Amazon SES 從該地址傳送電子郵件。

當您執行下列程式碼範例時，Amazon SES 會傳送電子郵件到您指定的地址。當您 (或電子郵件的收件人) 按一下電子郵件中的連結後，該地址即經過驗證。

若要將電子郵件地址新增至 Amazon SES 帳戶，請使用 VerifyEmailIdentity 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
```

```
]);

$email = 'email_address';

try {
    $result = $SesClient->verifyEmailIdentity([
        'EmailAddress' => $email,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

驗證電子郵件網域

Amazon SES 只能從已驗證的電子郵件地址或網域傳送電子郵件。透過驗證網域，可以證明您是該網域的擁有者。驗證網域時，您允許 Amazon SES 從該網域上的任何地址傳送電子郵件。

當您執行下列程式碼範例時，Amazon SES 會為您提供驗證字符。您必須將此字符新增至您網域的 DNS 組態。如需詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的使用 Amazon SES 驗證網域。

若要將傳送網域新增至 Amazon SES 帳戶，請使用 VerifyDomainIdentity 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
```

```
]);

$domain = 'domain.name';

try {
    $result = $SesClient->verifyDomainIdentity([
        'Domain' => $domain,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出電子郵件地址

若要擷取目前區域中提交的電子郵件地址清單 AWS ，無論驗證狀態為何，請使用 ListIdentities 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listIdentities([
        'IdentityType' => 'EmailAddress',
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出電子郵件網域

若要擷取目前區域中提交的電子郵件網域清單 AWS ，無論驗證狀態為何，請使用 ListIdentities 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listIdentities([
        'IdentityType' => 'Domain',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

刪除電子郵件地址

若要從身分清單中刪除已驗證的電子郵件地址，請使用 DeleteIdentity 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$email = 'email_address';

try {
    $result = $SesClient->deleteIdentity([
        'Identity' => $email,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

刪除電子郵件網域

若要從已驗證的身分清單中刪除已驗證的電子郵件網域，請使用 DeleteIdentity 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$domain = 'domain.name';

try {
    $result = $SesClient->deleteIdentity([
        'Identity' => $domain,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Amazon SES API 和 第 3 適用於 PHP 的 AWS SDK 版建立自訂電子郵件範本

Amazon Simple Email Service (Amazon SES) 可讓您使用 範本傳送為每個收件人個人化的電子郵件。範本包含主旨行以及電子郵件內文的文字和 HTML 部分。主旨和內文區段可能還包含專為每位收件人個人化的獨特值。

如需詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的使用 Amazon SES 傳送個人化電子郵件。

下列範例示範如何：

- 使用 CreateTemplate 建立電子郵件範本。
- 使用 ListTemplates 列出所有電子郵件範本。
- 使用 GetTemplate 擷取電子郵件範本。
- 使用 UpdateTemplate 更新電子郵件範本。
- 使用 DeleteTemplate 移除電子郵件範本。
- 使用 SendTemplatedEmail 傳送範本電子郵件。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon SES 的詳細資訊，請參閱《Amazon SES 開發人員指南》。

建立電子郵件範本

若要建立範本以便傳送個人化的電子郵件訊息，請使用 CreateTemplate 操作。範本可供任何有權在新增範本的 AWS 區域中傳送訊息的帳戶使用。

> **Note**
>
> Amazon SES 不會驗證您的 HTML，因此請在傳送電子郵件之前確定 HtmlPart 有效。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Template_Name';
$html_body = '<h1>AWS Amazon Simple Email Service Test Email</h1>' .
    '<p>This email was sent with <a href="https://aws.amazon.com/ses/">' .
    'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-php/">' .
    'AWS SDK for PHP</a>.</p>';
$subject = 'Amazon SES test (AWS SDK for PHP)';
$plaintext_body = 'This email was send with Amazon SES using the AWS SDK for PHP.';
```

```
try {
    $result = $SesClient->createTemplate([
        'Template' => [
            'HtmlPart' => $html_body,
            'SubjectPart' => $subject,
            'TemplateName' => $name,
            'TextPart' => $plaintext_body,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

取得電子郵件範本

若要檢視現有電子郵件範本的內容，包括主旨行、HTML 內文和純文字，請使用 GetTemplate 操作。僅需提供 TemplateName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Template_Name';

try {
    $result = $SesClient->getTemplate([
```

```
        'TemplateName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出所有電子郵件範本

若要擷取 AWS 帳戶 目前 AWS 區域中與 相關聯的所有電子郵件範本清單，請使用 ListTemplates 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listTemplates([
        'MaxItems' => 10,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

更新電子郵件範本

若要變更特定電子郵件範本的內容，包括主旨行、HTML 內文和純文字，請使用 UpdateTemplate 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Template_Name';
$html_body = '<h1>AWS Amazon Simple Email Service Test Email</h1>' .
    '<p>This email was sent with <a href="https://aws.amazon.com/ses/">' .
    'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-php/">' .
    'AWS SDK for PHP</a>.</p>';
$subject = 'Amazon SES test (AWS SDK for PHP)';
$plaintext_body = 'This email was send with Amazon SES using the AWS SDK for PHP.';

try {
    $result = $SesClient->updateTemplate([
        'Template' => [
            'HtmlPart' => $html_body,
            'SubjectPart' => $subject,
            'TemplateName' => $name,
            'TextPart' => $plaintext_body,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
```

```
}
```

刪除電子郵件範本

若要移除特定的電子郵件範本，請使用 DeleteTemplate 操作。您只需要提供 TemplateName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Template_Name';

try {
    $result = $SesClient->deleteTemplate([
        'TemplateName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用範本傳送電子郵件

若要使用範本將電子郵件傳送給收件人，請使用 SendTemplatedEmail 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$template_name = 'Template_Name';
$sender_email = 'email_address';
$recipient_emails = ['email_address'];

try {
    $result = $SesClient->sendTemplatedEmail([
        'Destination' => [
            'ToAddresses' => $recipient_emails,
        ],
        'ReplyToAddresses' => [$sender_email],
        'Source' => $sender_email,

        'Template' => $template_name,
        'TemplateData' => '{ }'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版管理電子郵件篩選條件

除了傳送電子郵件之外，您也可以透過 Amazon Simple Email Service (Amazon SES接收電子郵件。IP 地址篩選條件讓您可自由指定是否接受或拒絕來自某個 IP 地址或某範圍 IP 地址的郵件。如需詳細資訊，請參閱管理 Amazon SES 電子郵件接收的 IP 地址篩選條件。

下列範例示範如何：

- 使用 CreateReceiptFilter 建立電子郵件篩選條件。
- 使用 ListReceiptFilters 列出所有電子郵件篩選條件。
- 使用 DeleteReceiptFilter 刪除電子郵件篩選條件。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon SES 的詳細資訊，請參閱《Amazon SES 開發人員指南》。

建立電子郵件篩選條件

若要允許或封鎖來自特定 IP 地址的電子郵件，請使用 CreateReceiptFilter 操作。提供 IP 地址或地址範圍，以及此篩選條件的唯一識別名稱。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$filter_name = 'FilterName';
$ip_address_range = '10.0.0.1/24';

try {
```

```
    $result = $SesClient->createReceiptFilter([
        'Filter' => [
            'IpFilter' => [
                'Cidr' => $ip_address_range,
                'Policy' => 'Block|Allow',
            ],
            'Name' => $filter_name,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出所有電子郵件篩選條件

若要列出目前 AWS 區域中與 相關聯的 IP AWS 帳戶 地址篩選條件，請使用 ListReceiptFilters 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listReceiptFilters();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

刪除電子郵件篩選條件

若要移除特定 IP 地址現有的篩選條件，請使用 DeleteReceiptFilter 操作。提供獨一無二的篩選條件名稱以識別欲刪除的收件篩選條件。

如果您需要變更所篩選的地址範圍，則可刪除收件篩選條件後再建立新的篩選條件。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$filter_name = 'FilterName';

try {
    $result = $SesClient->deleteReceiptFilter([
        'FilterName' => $filter_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版建立和管理電子郵件規則

除了傳送電子郵件之外，您也可以透過 Amazon Simple Email Service (Amazon SES接收電子郵件。接收規則可讓您指定 Amazon SES 對您擁有的電子郵件地址或網域所接收的電子郵件執行的操作。規則可以傳送電子郵件到其他 AWS 服務，包括但不限於 Amazon S3、Amazon SNS 或 AWS Lambda。

如需詳細資訊，請參閱管理 Amazon SES 電子郵件接收的接收規則集和管理 Amazon SES 電子郵件接收的接收規則。

下列範例示範如何：

- 使用 CreateReceiptRuleSet 建立接收規則集。
- 使用 CreateReceiptRule 建立接收規則。
- 使用 DescribeReceiptRuleSet 描述接收規則集。
- 使用 DescribeReceiptRule 描述接收規則。
- 使用 ListReceiptRuleSets 列出所有接收規則集。
- 使用 UpdateReceiptRule 更新接收規則。
- 使用 DeleteReceiptRule 移除接收規則。
- 使用 DeleteReceiptRuleSet 移除接收規則集。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon SES 的詳細資訊，請參閱《Amazon SES 開發人員指南》。

建立接收規則集

接收規則集包含一組接收規則。您的帳戶必須至少有一個關聯的接收規則集，您才能建立接收規則。若要建立接收規則集，請使用 CreateReceiptRuleSet 操作並提供獨一無二的 RuleSetName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Rule_Set_Name';

try {
    $result = $SesClient->createReceiptRuleSet([
        'RuleSetName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

建立接收規則

透過為現有的接收規則集新增接收規則，控制您的內送電子郵件。此範例說明如何建立接收規則，將傳入的訊息傳送至 Amazon S3 儲存貯體，但您也可以傳送訊息至 Amazon SNS 和 AWS Lambda。若要建立接收規則，請使用 CreateReceiptRule 操作並提供規則的名稱和 RuleSetName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
```

```
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';
$s3_bucket = 'Bucket_Name';

try {
    $result = $SesClient->createReceiptRule([
        'Rule' => [
            'Actions' => [
                [
                    'S3Action' => [
                        'BucketName' => $s3_bucket,
                    ],
                ],
            ],
            'Name' => $rule_name,
            'ScanEnabled' => true,
            'TlsPolicy' => 'Optional',
            'Recipients' => ['<string>']
        ],
        'RuleSetName' =>  $rule_set_name,

     ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

描述接收規則集

每秒傳回一次指定的接收規則集的詳細資訊。若要使用 DescribeReceiptRuleSet 操作，請提供 RuleSetName。

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Rule_Set_Name';

try {
    $result = $SesClient->describeReceiptRuleSet([
        'RuleSetName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

描述接收規則

傳回指定的接收規則的詳細資訊。若要使用 DescribeReceiptRule 操作，請提供 RuleName 和 RuleSetName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
```

```
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';

try {
    $result = $SesClient->describeReceiptRule([
        'RuleName' => $rule_name,
        'RuleSetName' => $rule_set_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出所有接收規則集

若要列出 AWS 帳戶 目前 AWS 區域中 下存在的接收規則集，請使用 ListReceiptRuleSets 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

try {
    $result = $SesClient->listReceiptRuleSets();
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

更新接收規則

此範例說明如何更新將傳入訊息傳送至 AWS Lambda 函數的接收規則，但您也可以傳送訊息至 Amazon SNS 和 Amazon S3。若要使用 UpdateReceiptRule 操作，請提供新的接收規則名稱和 RuleSetName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';
$lambda_arn = 'Amazon Resource Name (ARN) of the AWS Lambda function';
$sns_topic_arn = 'Amazon Resource Name (ARN) of the Amazon SNS topic';

try {
    $result = $SesClient->updateReceiptRule([
        'Rule' => [
            'Actions' => [
                'LambdaAction' => [
                    'FunctionArn' => $lambda_arn,
                    'TopicArn' => $sns_topic_arn,
```

```
                ],
            ],
            'Enabled' => true,
            'Name' => $rule_name,
            'ScanEnabled' => false,
            'TlsPolicy' => 'Require',
        ],
        'RuleSetName' => $rule_set_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

刪除接收規則集

移除所指定而目前未停用的接收規則集。如此亦將刪除其包含的所有接收規則。若要刪除接收規則集，請使用 DeleteReceiptRuleSet 操作並提供 RuleSetName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$name = 'Rule_Set_Name';

try {
    $result = $SesClient->deleteReceiptRuleSet([
        'RuleSetName' => $name,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

刪除接收規則

若要刪除指定的接收規則，請使用 DeleteReceiptRule 操作並提供 RuleName 和 RuleSetName。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

範例程式碼

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';

try {
    $result = $SesClient->deleteReceiptRule([
        'RuleName' => $rule_name,
        'RuleSetName' => $rule_set_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Amazon SES API 和 第 3 適用於 PHP 的 AWS SDK 版監控您的傳送活動

Amazon Simple Email Service (Amazon SES) 提供監控傳送活動的方法。我們建議您實作這些方法，以持續追蹤重要指標，例如帳戶的退信、抱怨與拒收率等。過高的退信率和投訴率可能會影響您使用 Amazon SES 傳送電子郵件的能力。

下列範例示範如何：

- 使用 GetSendQuota 檢查您的傳送份額。
- 使用 GetSendStatistics 監控您的傳送活動。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon SES 的詳細資訊，請參閱《Amazon SES 開發人員指南》。

檢查您的傳送配額

您在 24 小時期間內所能傳送的訊息數目有特定限制。若要檢查您尚餘幾則訊息可以傳送，請使用 GetSendQuota 操作。如需詳細資訊，請參閱管理您的 Amazon SES 傳送限制。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

範例程式碼

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
```

```
    'region' => 'us-east-1'

]);

try {
    $result = $SesClient->getSendQuota();
    $send_limit = $result["Max24HourSend"];
    $sent = $result["SentLast24Hours"];
    $available = $send_limit - $sent;
    print("<p>You can send " . $available . " more messages in the next 24 hours.</
p>");
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

監控您的傳送活動

若要擷取您在過去兩週已傳送的訊息各項指標，請使用 GetSendStatistics 操作。此範例將以 15 分鐘為增量，傳回嘗試交付、退信、投訴與拒收的訊息數目。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

範例程式碼

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

try {
    $result = $SesClient->getSendStatistics();
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用 Amazon SES API 和第 3 適用於 PHP 的 AWS SDK 版授權寄件者

若要讓其他 AWS 帳戶、 AWS Identity and Access Management 使用者 AWS 或服務代表您透過 Amazon Simple Email Service (Amazon SES) 傳送電子郵件，您可以建立傳送授權政策。此為附加至您自有身分的 JSON 文件。

政策將明確列出您允許誰代表該身分傳送，以及有哪些傳送條件。除了您本人以及政策中明確授予許可的實體，其餘所有寄件者皆不得傳送電子郵件。一個身分可以沒有政策、有一個政策或有多個政策。您也可以使用含有多個陳述式的單一政策來達成多個政策的效果。

如需詳細資訊，請參閱透過 Amazon SES 使用傳送授權。

下列範例示範如何：

- 使用 PutIdentityPolicy 建立已獲授權的寄件者。
- 使用 GetIdentityPolicies 擷取已獲授權寄件者的政策。
- 使用 ListIdentityPolicies 列出已獲授權的寄件者。
- 使用 DeleteIdentityPolicy 對已獲授權的寄件者撤銷許可。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

如需使用 Amazon SES 的詳細資訊，請參閱《Amazon SES 開發人員指南》。

建立授權寄件者

若要授權其他人代表您 AWS 帳戶 傳送電子郵件，請使用身分政策來新增或更新授權，以從已驗證的電子郵件地址或網域傳送電子郵件。若要建立身分政策，請使用 PutIdentityPolicy 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

範例程式碼

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";
$other_aws_account = "0123456789";
$policy = <<<EOT
{
  "Id":"ExampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeAccount",
      "Effect":"Allow",
      "Resource":"$identity",
      "Principal":{
        "AWS":[ "$other_aws_account" ]
      },
      "Action":[
        "SES:SendEmail",
        "SES:SendRawEmail"
      ]
    }
  ]
}
EOT;
$name = "policyName";

try {
    $result = $SesClient->putIdentityPolicy([
```

```
        'Identity' => $identity,
        'Policy' => $policy,
        'PolicyName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

擷取授權寄件者的政策

傳回與特定的電子郵件身分或網域身分相關聯的傳送授權政策。若要取得指定的電子郵件地址或網域的傳送授權，請使用 GetIdentityPolicy 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

範例程式碼

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";
$policies = ["policyName"];

try {
    $result = $SesClient->getIdentityPolicies([
        'Identity' => $identity,
        'PolicyNames' => $policies,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

列出授權寄件者

若要列出與目前 AWS 區域中的特定電子郵件身分或網域身分相關聯的傳送授權政策，請使用 ListIdentityPolicies 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

範例程式碼

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";

try {
    $result = $SesClient->listIdentityPolicies([
        'Identity' => $identity,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

撤銷授權寄件者的許可

使用 DeleteIdentityPolicy 操作刪除相關聯的身分政策，移除另一個 傳送授權 AWS 帳戶 ，以使用電子郵件身分或網域身分傳送電子郵件。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

範例程式碼

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";
$name = "policyName";

try {
    $result = $SesClient->deleteIdentityPolicy([
        'Identity' => $identity,
        'PolicyName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon SNS 範例

Amazon Simple Notification Service (Amazon SNS) 是一種 Web 服務，會協調和管理消息傳遞或發送到訂閱端點或客戶端。

在 Amazon SNS 中，有兩種類型的用戶端：發佈者（也稱為生產者）和訂閱者（也稱為消費者）。發佈者透過製作並傳送訊息到主題 (其為邏輯存取點和通訊管道) 與訂閱者進行非同步的通訊。訂閱者（網路伺服器、電子郵件地址、Amazon SQS 佇列、 AWS Lambda 函數）訂閱主題時，會使用或接收其中一個支援的通訊協定 (Amazon SQS、HTTP/HTTPS URLs、電子郵件 AWS SMS、Lambda) 的訊息或通知。

第 3 適用於 PHP 的 AWS SDK 版的所有範例程式碼都可在 GitHub 上取得。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon SNS 中的主題
- 使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon SNS 中的訂閱
- 使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SNS 中傳送簡訊

使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon SNS 中的主題

若要傳送通知至 Amazon Simple Queue Service (Amazon SQS)、HTTP/HTTPS URLs AWS SMS、電子郵件或 AWS Lambda，您必須先建立主題來管理訊息交付給該主題的任何訂閱者。

就觀察者設計模式而言，主題即有如主旨。建立主題之後，您便要新增訂閱者，其將於訊息發佈至該主題時自動收到通知。

進一步了解如何使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SNS 中管理訂閱中的主題。

下列範例示範如何：

- 使用 CreateTopic 建立要發佈通知的主題。
- 使用 ListTopics 傳回申請者的主題清單。
- 使用 DeleteTopic 刪除主題及其所有訂閱。
- 使用 GetTopicAttributes 傳回主題的所有屬性。
- 使用 SetTopicAttributes 允許主題擁有者將主題的屬性設為新的值。

如需使用 Amazon SNS 的詳細資訊，請參閱訊息傳遞狀態的 Amazon SNS 主題屬性。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

建立主題

若要建立主題，請使用 CreateTopic 操作。

您 中的每個主題名稱 AWS 帳戶 都必須是唯一的。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

列出您的主題

若要列出目前 AWS 區域中最多 100 個現有主題，請使用 ListTopics 操作。

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除主題

若要移除現有的主題及其所有訂閱，請使用 DeleteTopic 操作。

凡是仍未交付予訂閱者的任何訊息都將一併刪除。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

取得主題屬性

若要擷取單一現有主題的各項屬性，請使用 GetTopicAttributes 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

設定主題屬性

若要更新單一現有主題的各項屬性，請使用 SetTopicAttributes 操作。

您只能設定 `Policy`、`DisplayName` 和 `DeliveryPolicy` 屬性。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon SNS 中的訂閱

使用 Amazon Simple Notification Service (Amazon SNS) 主題將通知傳送至 Amazon Simple Queue Service (Amazon SQS)、HTTP/HTTPS、電子郵件地址 AWS Server Migration Service (AWS SMS) 或 AWS Lambda。

訂閱會連接到主題，以管理對訂閱者的訊息傳送。進一步了解如何使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SNS 中管理主題中建立主題。

下列範例示範如何：

- 使用 Subscribe 訂閱現有的主題。
- 使用 ConfirmSubscription 驗證訂閱。
- 使用 ListSubscriptionsByTopic 列出現有的訂閱。
- 使用 Unsubscribe 刪除訂閱。
- 使用 Publish 傳送訊息給某主題的所有訂閱者。

如需使用 Amazon SNS 的詳細資訊，請參閱使用 Amazon SNS 進行System-to-System傳訊。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

讓電子郵件地址訂閱主題

若要為電子郵件地址起始訂閱，請使用 Subscribe 操作。

您可以使用 訂閱方法將數個不同的端點訂閱至 Amazon SNS 主題，具體取決於傳遞的參數所使用的值。本主題的其他範例將示範做法。

在此範例中，端點是電子郵件地址。確認字符會傳送至該電子郵件。收到確認符記後，可於三天內驗證訂閱。

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

訂閱應用程式端點至主題

若要為 web 應用程式起始訂閱，請使用 Subscribe 操作。

您可以使用 訂閱方法將數個不同的端點訂閱至 Amazon SNS 主題，具體取決於傳遞的參數所使用的值。本主題的其他範例將示範做法。

在此範例中，端點是 URL。確認字符會傳送至該網址。收到確認符記後，可於三天內驗證訂閱。

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

將 Lambda 函數訂閱至主題

若要啟動 Lambda 函數的訂閱，請使用訂閱操作。

您可以使用 訂閱方法將數個不同的端點訂閱至 Amazon SNS 主題，具體取決於傳遞的參數所使用的值。本主題的其他範例將示範做法。

在此範例中，端點是 Lambda 函數。確認字符會傳送至此 Lambda 函數。收到確認符記後，可於三天內驗證訂閱。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'lambda';
$endpoint = 'arn:aws:lambda:us-east-1:123456789023:function:messageStore';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

訂閱文字簡訊至主題

如需傳送簡訊至多個電話號碼，各號碼都要訂閱主題。

若要為電話號碼起始訂閱，請使用 Subscribe 操作。

您可以使用 訂閱方法將數個不同的端點訂閱至 Amazon SNS 主題，具體取決於傳遞的參數所使用的值。本主題的其他範例將示範做法。

在此範例中，端點是 E.164 格式的電話號碼，遵照國際電信通訊所採用的標準。

確認字符會傳送至該電話號碼。收到確認符記後，可於三天內驗證訂閱。

如需使用 Amazon SNS 傳送簡訊的替代方法，請參閱使用 第 3 版在 適用於 PHP 的 AWS SDK Amazon SNS 中傳送簡訊。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'sms';
$endpoint = '+1XXX5550100';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

確認主題的訂閱

若要實際建立訂閱，端點擁有者必須使用最初建立訂閱時傳送的符記，確認其有意接收來自該主題的訊息，如前所述。確認符記的有效期為三天。三天之後，您可以透過建立新的訂閱重新傳送符記。

若要確認訂閱，請使用 ConfirmSubscription 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

列出主題的訂閱

若要列出指定區域中最多 100 個現有訂閱 AWS ，請使用 ListSubscriptions 操作。

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

取消訂閱主題

若要移除已訂閱主題的端點，請使用 Unsubscribe 操作。

如果訂閱需要身分驗證才能刪除，則只有訂閱的擁有者或主題的擁有者可以取消訂閱，而且需要 AWS 簽章。如果取消訂閱呼叫無須身分驗證，而申請者也並非訂閱擁有者，便會向端點交付一則最終取消訊息。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

發佈訊息至 Amazon SNS 主題

若要將訊息傳遞至訂閱 Amazon SNS 主題的每個端點，請使用發佈操作。

建立物件，其中包含發佈訊息的參數，包括訊息文字和 Amazon SNS 主題的 Amazon Resource Name (ARN)。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';
```

```
try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SNS 中傳送簡訊

您可以使用 Amazon Simple Notification Service (Amazon SNS) 將文字訊息或簡訊傳送至啟用 SMS 的裝置。您可以直接傳送訊息至一組電話號碼，或一次傳送一則訊息至多組電話號碼，只要訂閱那些電話號碼到主題並且傳送您的訊息到該主題即可。

使用 Amazon SNS 指定簡訊的偏好設定，例如如何最佳化交付（用於成本或可靠交付）、每月花費限制、如何記錄訊息交付，以及是否訂閱每日簡訊用量報告。系統會擷取這些偏好設定，並將其設定為 Amazon SNS 的 SMS 屬性。

當您傳送簡訊時，請指定使用 E.164 格式的電話號碼。E.164 是國際電信通訊所採用的電話號碼結構的標準。遵循此格式的電話號碼最多可以有 15 位數，前面加上加號 (+) 字元和國碼。例如，E.164 格式的美國電話號碼顯示為 +1001XXX5550100。

下列範例示範如何：

- 使用 GetSMSAttributes 擷取從您的帳戶傳送簡訊的預設設定。
- 使用 SetSMSAttributes 更新從您的帳戶傳送簡訊的預設設定。
- 使用 CheckIfPhoneNumberISOptedOut 檢查指定的電話號碼擁有者是否已選擇停止從您的帳戶接收簡訊。
- 使用 ListPhoneNumberOptedOut 列出其擁有者已選擇停止從您的帳戶接收簡訊的電話號碼。
- 使用 Publish 直接傳送文字訊息 (簡訊) 至電話號碼。

如需使用 Amazon SNS 的詳細資訊，請參閱將 Amazon SNS 用於使用者通知搭配行動電話號碼做為訂閱者（傳送簡訊）。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

取得 SMS 屬性

若要擷取簡訊的預設設定，請使用 GetSMSAttributes 操作。

本範例將取得 `DefaultSMSType` 屬性。此屬性控制著簡訊的傳送方式，`Promotional` 會將訊息交付最佳化為產生最低的成本，而 `Transactional` 會將訊息交付最佳化為達成最高的可靠性。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

設定 SMS 屬性

若要更新簡訊的預設設定，請使用 SetSMSAttributes 操作。

此範例將 DefaultSMSType 屬性設為 Transactional，藉此將訊息交付最佳化為達成最高的可靠性。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

檢查電話號碼是否已選擇退出

若要判斷指定的電話號碼擁有者是否已選擇停止從您的帳戶接收簡訊，請使用 CheckIfPhoneNumberIsOptedOut 操作。

本範例指定 E.164 格式的電話號碼，此為國際電信通訊所採用的標準。

匯入

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

列出選擇退出的電話號碼

若要擷取其擁有者已選擇停止從您的帳戶接收簡訊的電話號碼清單，請使用 ListPhoneNumbersOptedOut 操作。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

發佈至文字訊息 (SMS 訊息)

若要直接交付文字訊息 (簡訊) 至電話號碼，請使用 Publish 操作。

本範例指定 E.164 格式的電話號碼，此為國際電信通訊所採用的標準。

每則簡訊最多可包含 140 個位元組。單次簡訊發佈動作的大小上限為 1,600 個位元組。

如需有關傳送簡訊的詳細資訊，請參閱傳送簡訊。

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

範例程式碼

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
```

```
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版的 Amazon SQS 範例

Amazon Simple Queue Service (SQS) 是一種快速、可靠、可擴展、全受管的訊息佇列服務。Amazon SQS 可讓您解耦雲端應用程式的元件。Amazon SQS 包括具有高輸送量和at-least-once處理的標準佇列，以及提供 FIFO (first>-in、first>-out) 交付和恰好一次處理的 FIFO 佇列。

第 3 適用於 PHP 的 AWS SDK 版的所有範例程式碼都可在 GitHub 上取得。

主題

- 使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SQS 中啟用長輪詢
- 使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon SQS 中的可見性逾時
- 使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SQS 中傳送和接收訊息
- 在 Amazon SQS 中使用無效字母佇列搭配第 3 適用於 PHP 的 AWS SDK 版
- 將 Amazon SQS 中的佇列與第 3 適用於 PHP 的 AWS SDK 版搭配使用

使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SQS 中啟用長輪詢

長輪詢可讓 Amazon SQS 在傳送回應之前等待指定的時間，讓訊息在佇列中變成可用，以減少空回應的數量。此外，長輪詢可查詢所有伺服器而非指查詢取樣的伺服器，來減少假的空白回應。若要啟用長輪詢，請針對接收的訊息指定非零的等待時間。如需進一步了解，請參閱 SQS 長輪詢。

下列範例示範如何：

- 使用 SetQueueAttributes，在 Amazon SQS 佇列上設定屬性以啟用長輪詢。
- 使用 ReceiveMessage 透過長輪詢擷取一則或多則訊息。
- 使用 CreateQueue 建立長輪詢佇列。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

在佇列上設定屬性以啟用長輪詢

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->setQueueAttributes([
        'Attributes' => [
            'ReceiveMessageWaitTimeSeconds' => 20
        ],
        'QueueUrl' => $queueUrl, // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用長輪詢擷取訊息

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage([
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 1,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 20,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

建立具有長輪詢的佇列

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueName = "QUEUE_NAME";


$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue([
        'QueueName' => $queueName,
        'Attributes' => [
            'ReceiveMessageWaitTimeSeconds' => 20
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版管理 Amazon SQS 中的可見性逾時

可見性逾時是 Amazon SQS 防止其他耗用元件接收和處理訊息的一段時間。如需進一步了解，請參閱可見性逾時。

下列範例將說明：

- 使用 ChangeMessageVisibilityBatch 將佇列中的指定訊息可見性逾時變更為新值。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

變更多個訊息的可見性逾時

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage(array(
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 10,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
    ));
    $messages = $result->get('Messages');
    if ($messages != null) {
        $entries = array();
        for ($i = 0; $i < count($messages); $i++) {
            $entries[] = [
                'Id' => 'unique_is_msg' . $i, // REQUIRED
                'ReceiptHandle' => $messages[$i]['ReceiptHandle'], // REQUIRED
                'VisibilityTimeout' => 3600
            ];
        }
        $result = $client->changeMessageVisibilityBatch([
            'Entries' => $entries,
            'QueueUrl' => $queueUrl
        ]);

        var_dump($result);
    } else {
        echo "No messages in queue \n";
    }
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版在 Amazon SQS 中傳送和接收訊息

若要了解 Amazon SQS 訊息，請參閱 Service Quotas 使用者指南中的傳送訊息至 SQS 佇列，以及從 SQS 佇列接收和刪除訊息。

下列範例示範如何：

- 使用 SendMessage 交付訊息到指定的佇列。
- 使用 ReceiveMessage 從指定的佇列擷取一則或多則訊息 (最多 10 則)。
- 使用 DeleteMessage 從佇列刪除訊息。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

傳送訊息

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);
```

```
$params = [
    'DelaySeconds' => 10,
    'MessageAttributes' => [
        "Title" => [
            'DataType' => "String",
            'StringValue' => "The Hitchhiker's Guide to the Galaxy"
        ],
        "Author" => [
            'DataType' => "String",
            'StringValue' => "Douglas Adams."
        ],
        "WeeksOn" => [
            'DataType' => "Number",
            'StringValue' => "6"
        ]
    ],
    'MessageBody' => "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
    'QueueUrl' => 'QUEUE_URL'
];

try {
    $result = $client->sendMessage($params);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

接收和刪除訊息

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueUrl = "QUEUE_URL";
```

```
$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage([
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 1,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 0,
    ]);
    if (!empty($result->get('Messages'))) {
        var_dump($result->get('Messages')[0]);
        $result = $client->deleteMessage([
            'QueueUrl' => $queueUrl, // REQUIRED
            'ReceiptHandle' => $result->get('Messages')[0]['ReceiptHandle'] // REQUIRED
        ]);
    } else {
        echo "No messages in queue. \n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

在 Amazon SQS 中使用無效字母佇列搭配第 3 適用於 PHP 的 AWS SDK 版

無效字母佇列是其他 (來源) 佇列針對無法成功處理的訊息，可做為目標瞄準的佇列。無效字母佇列可讓您擱置並隔離這類訊息，以判定無法成功處理訊息的原因。您必須針對傳送訊息至無效字母佇列的各個來源佇列單獨進行設定。多個佇列可以將目標設為同一個無效字母佇列。

若要進一步了解，請參閱使用 SQS 無效字母佇列。

下列範例將說明：

- 使用 SetQueueAttributes 啟用無效字母佇列。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

啟用無效字母佇列

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->setQueueAttributes([
        'Attributes' => [
            'RedrivePolicy' => "{\"deadLetterTargetArn\":\"DEAD_LETTER_QUEUE_ARN\",
\"maxReceiveCount\":\"10\"}"
        ],
        'QueueUrl' => $queueUrl // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

將 Amazon SQS 中的佇列與第 3 適用於 PHP 的 AWS SDK 版搭配使用

若要了解 Amazon SQS 佇列，請參閱 SQS 佇列的運作方式。

下列範例示範如何：

- 使用 ListQueues 傳回佇列的清單。
- 使用 CreateQueue 建立新的佇列。
- 使用 GetQueueUrl 傳回現有佇列的 URL。
- 使用 DeleteQueue 刪除指定的佇列。

您可以在 GitHub 上 適用於 PHP 的 AWS SDK 取得 的所有範例程式碼。 GitHub

登入資料

在執行範例程式碼之前，請先設定您的 AWS 登入資料，如中所述the section called “使用 驗證 AWS”。然後匯入 適用於 PHP 的 AWS SDK，如 中所述the section called “安裝 ”。

傳回佇列清單

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->listQueues();
    foreach ($result->get('QueueUrls') as $queueUrl) {
        echo "$queueUrl\n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

建立佇列

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueName = "SQS_QUEUE_NAME";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue([
        'QueueName' => $queueName,
        'Attributes' => [
            'DelaySeconds' => 5,
            'MaximumMessageSize' => 4096, // 4 KB
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

傳回佇列的 URL

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueName = "SQS_QUEUE_NAME";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->getQueueUrl([
        'QueueName' => $queueName // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

刪除佇列

匯入

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

範例程式碼

```
$queueUrl = "SQS_QUEUE_URL";
```

```
$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->deleteQueue([
        'QueueUrl' => $queueUrl // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

使用第 3 適用於 PHP 的 AWS SDK 版將事件傳送至 Amazon EventBridge 全域端點

您可以使用 Amazon EventBridge 全域端點來改善事件驅動應用程式的可用性和可靠性。

設定 EventBridge 全域端點之後，您可以使用適用於 PHP 的 SDK 將事件傳送給它。

> **Important**
>
> 若要搭配適用於 PHP 的 SDK 使用 EventBridge 全域端點，您的 PHP 環境必須安裝AWS 通用執行期 (AWS CRT) 延伸模組。

下列範例使用 的 PutEvents 方法`EventBridgeClient`，將單一事件傳送至 EventBridge 全域端點。

```
<?php
/* Send a single event to an existing Amazon EventBridge global endpoint. */
require '../vendor/autoload.php';

use Aws\EventBridge\EventBridgeClient;

$evClient = new EventBridgeClient([
    'region' => 'us-east-1'
```

```
]);

$endpointId = 'xxxx123456.xxx';  // Existing EventBridge global endpointId.
$eventBusName = 'default';       // Existing event bus in the us-east-1 Region.

$event =  [
    'Source' => 'my-php-app',
    'DetailType' => 'test',
    'Detail' => json_encode(['foo' => 'bar']),
    'Time' => new DateTime(),
    'Resources' => ['php-script'],
    'EventBusName' => $eventBusName,
    'TraceHeader' => 'test'
];

$result = $evClient->putEvents([
    'EndpointId' => $endpointId,
    'Entries' => [$event]
]);
```

此部落格文章包含有關 EventBridge 全域端點的詳細資訊。

適用於 PHP 的 SDK 程式碼範例

本主題中的程式碼範例說明如何 適用於 PHP 的 AWS SDK 搭配 使用 AWS。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

有些服務包含其他範例類別，示範如何利用服務特定的程式庫或函數。

服務

- 使用適用於 PHP 的 SDK 的 API Gateway 範例
- 使用適用於 PHP 的 SDK 的 Aurora 範例
- 使用適用於 PHP 的 SDK 的 Auto Scaling 範例
- 使用適用於 PHP 的 SDK 的 Amazon Bedrock 範例
- 使用適用於 PHP 的 SDK 的 Amazon Bedrock 執行期範例
- 使用適用於 PHP 的 SDK 的 Amazon DocumentDB 範例
- 使用適用於 PHP 的 SDK 的 DynamoDB 範例
- 使用適用於 PHP 的 SDK 的 Amazon EC2 範例
- AWS Glue 使用適用於 PHP 的 SDK 的範例
- 使用適用於 PHP 的 SDK 的 IAM 範例
- 使用適用於 PHP 的 SDK 的 Kinesis 範例
- AWS KMS 使用適用於 PHP 的 SDK 的範例
- 使用適用於 PHP 的 SDK 的 Lambda 範例
- 使用適用於 PHP 的 SDK 的 Amazon MSK 範例
- 使用適用於 PHP 的 SDK 的 Amazon RDS 範例
- 使用適用於 PHP 的 SDK 的 Amazon RDS Data Service 範例
- 使用適用於 PHP 的 SDK 的 Amazon Rekognition 範例
- 使用適用於 PHP 的 SDK 的 Amazon S3 範例
- 使用適用於 PHP 的 SDK 的 S3 目錄儲存貯體範例
- 使用適用於 PHP 的 SDK 的 Amazon SES 範例

- 使用適用於 PHP 的 SDK 的 Amazon SNS 範例
- 使用適用於 PHP 的 SDK 的 Amazon SQS 範例

使用適用於 PHP 的 SDK 的 API Gateway 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 API Gateway 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 動作
- 案例

動作

`GetBasePathMapping`

以下程式碼範例顯示如何使用 `GetBasePathMapping`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;
```

```
/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * /////////////////////////////////////////////////////////////////////////// */

function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function getsTheBasePathMapping()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 GetBasePathMapping。

ListBasePathMappings

以下程式碼範例顯示如何使用 ListBasePathMappings。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * /////////////////////////////////////////////////////////////////////////// */

function listBasePathMappings($apiGatewayClient, $domainName)
{
```

```
    try {
        $result = $apiGatewayClient->getBasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function listTheBasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 ListBasePathMappings。

UpdateBasePathMapping

以下程式碼範例顯示如何使用 UpdateBasePathMapping。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 *
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function updateBasePathMapping(
    $apiGatewayClient,
    $basePath,
    $domainName,
    $patchOperations
) {
    try {
        $result = $apiGatewayClient->updateBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
            'patchOperations' => $patchOperations
        ]);
        return 'The updated base path\'s URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function updateTheBasePathMapping()
```

```
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo updateBasePathMapping(
        $apiGatewayClient,
        '(none)',
        'example.com',
        $patchOperations
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 UpdateBasePathMapping。

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for PHP

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

使用適用於 PHP 的 SDK 的 Aurora 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Aurora 來執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 案例

案例

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼範例示範如何建立 Web 應用程式，追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES傳送報告。

SDK for PHP

示範如何使用 適用於 PHP 的 AWS SDK 建立 Web 應用程式，以使用 Amazon Simple Email Service (Amazon SES) 追蹤 Amazon RDS 資料庫中的工作項目和電子郵件報告。這個範例使用以 React.js 建置的前端與 RESTful PHP 後端互動。

- 將 React.js Web 應用程式與 AWS 服務整合。
- 列出、新增、更新和刪除 Amazon RDS 資料表中的項目。
- 使用 Amazon SES 傳送篩選工作項目的電子郵件報告。
- 使用隨附的 AWS CloudFormation 指令碼部署和管理範例資源。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

使用適用於 PHP 的 SDK 的 Auto Scaling 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Auto Scaling 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

開始使用

Hello Auto Scaling

下列程式碼範例示範如何開始使用 Auto Scaling。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);
```

```
        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 DescribeAutoScalingGroups。

主題

- 基本概念
- 動作

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 使用啟動範本和可用區域建立 Amazon EC2 Auto Scaling 群組，並取得執行中執行個體的相關資訊。
- 啟用 Amazon CloudWatch 指標集合。
- 更新群組所需的容量，並等待執行個體啟動。
- 終止 群組中的執行個體。
- 列出為了回應使用者請求和容量變更而發生的擴展活動。
- 取得 CloudWatch 指標的統計資料，然後清除資源。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace AutoScaling;
```

```
use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an Auto
 Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
```

```
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you created
 in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
```

```
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups: {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
```

```
        $this->autoScalingService->setDesiredCapacity($this->autoScalingGroupName,
 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
```

```
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
```

```
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']} recorded:
\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
```

```
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

動作

`CreateAutoScalingGroup`

以下程式碼範例顯示如何使用 `CreateAutoScalingGroup`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateAutoScalingGroup。

DeleteAutoScalingGroup

以下程式碼範例顯示如何使用 DeleteAutoScalingGroup。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function deleteAutoScalingGroup($autoScalingGroupName)
{
    return $this->autoScalingClient->deleteAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'ForceDelete' => true,
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteAutoScalingGroup。

DescribeAutoScalingGroups

以下程式碼範例顯示如何使用 DescribeAutoScalingGroups。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
```

```
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 DescribeAutoScalingGroups。

DescribeAutoScalingInstances

以下程式碼範例顯示如何使用 DescribeAutoScalingInstances。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function describeAutoScalingInstances($instanceIds)
    {
        return $this->autoScalingClient->describeAutoScalingInstances([
            'InstanceIds' => $instanceIds
        ]);
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 DescribeAutoScalingInstances。

DescribeScalingActivities

以下程式碼範例顯示如何使用 DescribeScalingActivities。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 DescribeScalingActivities。

`DisableMetricsCollection`

以下程式碼範例顯示如何使用 `DisableMetricsCollection`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function disableMetricsCollection($autoScalingGroupName)
{
    return $this->autoScalingClient->disableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 DisableMetricsCollection。

EnableMetricsCollection

以下程式碼範例顯示如何使用 `EnableMetricsCollection`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function enableMetricsCollection($autoScalingGroupName, $granularity)
{
    return $this->autoScalingClient->enableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'Granularity' => $granularity,
    ]);
}
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 EnableMetricsCollection。

SetDesiredCapacity

以下程式碼範例顯示如何使用 `SetDesiredCapacity`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 SetDesiredCapacity。

`TerminateInstanceInAutoScalingGroup`

以下程式碼範例顯示如何使用 `TerminateInstanceInAutoScalingGroup`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient->terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' => $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
$attempts < 5) {
                error_log("Cannot terminate an instance while it is still pending.
Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
```

```
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 TerminateInstanceInAutoScalingGroup。

`UpdateAutoScalingGroup`

以下程式碼範例顯示如何使用 `UpdateAutoScalingGroup`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
        if (array_key_exists('MaxSize', $args)) {
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 UpdateAutoScalingGroup。

使用適用於 PHP 的 SDK 的 Amazon Bedrock 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon Bedrock 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 動作

動作

`ListFoundationModels`

以下程式碼範例顯示如何使用 `ListFoundationModels`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出可用的 Amazon Bedrock 基礎模型。

```
    public function listFoundationModels()
    {
        $bedrockClient = new BedrockClient([
            'region' => 'us-west-2',
            'profile' => 'default'
```

```
        ]);
        $response = $bedrockClient->listFoundationModels();
        return $response['modelSummaries'];
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 ListFoundationModels。

使用適用於 PHP 的 SDK 的 Amazon Bedrock 執行期範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon Bedrock 執行期來執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 案例
- Amazon Nova
- Amazon Titan Image Generator
- Anthropic Claude
- Stable Diffusion

案例

在 Amazon Bedrock 上調用多個基礎模型

下列程式碼範例示範如何在 Amazon Bedrock 上準備並傳送提示至各種大型語言模型 (LLMs)

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

在 Amazon Bedrock 上調用多個 LLMs。

```
namespace BedrockRuntime;

class GettingStartedWithBedrockRuntime
{
    protected BedrockRuntimeService $bedrockRuntimeService;
    public function runExample()
    {
        echo "\n";
        echo "----------------------------------------------------------------------
\n";
        echo "Welcome to the Amazon Bedrock Runtime getting started demo using PHP!
\n";
        echo "----------------------------------------------------------------------
\n";
        $bedrockRuntimeService = new BedrockRuntimeService();
        $prompt = 'In one paragraph, who are you?';
        echo "\nPrompt: " . $prompt;
        echo "\n\nAnthropic Claude:\n";
        echo $bedrockRuntimeService->invokeClaude($prompt);
        echo
 "\n----------------------------------------------------------------------\n";
        $image_prompt = 'stylized picture of a cute old steampunk robot';
        echo "\nImage prompt: " . $image_prompt;
        echo "\n\nStability.ai Stable Diffusion XL:\n";
        $diffusionSeed = rand(0, 4294967295);
        $style_preset = 'photographic';
        $base64 = $bedrockRuntimeService->invokeStableDiffusion($image_prompt,
 $diffusionSeed, $style_preset);
        $image_path = $this->saveImage($base64, 'stability.stable-diffusion-xl');
        echo "The generated image has been saved to $image_path";
        echo "\n\nAmazon Titan Image Generation:\n";
        $titanSeed = rand(0, 2147483647);
        $base64 = $bedrockRuntimeService->invokeTitanImage($image_prompt,
 $titanSeed);
        $image_path = $this->saveImage($base64, 'amazon.titan-image-generator-v1');
        echo "The generated image has been saved to $image_path";
    }

    private function saveImage($base64_image_data, $model_id): string
    {
        $output_dir = "output";
        if (!file_exists($output_dir)) {
```

```
            mkdir($output_dir);
        }

        $i = 1;
        while (file_exists("$output_dir/$model_id" . '_' . "$i.png")) {
            $i++;
        }

        $image_data = base64_decode($base64_image_data);
        $file_path = "$output_dir/$model_id" . '_' . "$i.png";
        $file = fopen($file_path, 'wb');
        fwrite($file, $image_data);
        fclose($file);
        return $file_path;
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - InvokeModel
  - InvokeModelWithResponseStream

Amazon Nova

Converse

下列程式碼範例示範如何使用 Bedrock 的 Converse API 將文字訊息傳送至 Amazon Nova。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

使用 Bedrock 的 Converse API 將文字訊息傳送至 Amazon Nova。

```
// Use the Conversation API to send a text message to Amazon Nova.

use Aws\BedrockRuntime\BedrockRuntimeClient;
```

```
use Aws\Exception\AwsException;
use RuntimeException;

class Converse
{
    public function converse(): string
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        $client = new BedrockRuntimeClient([
            'region' => 'us-east-1',
            'profile' => 'default'
        ]);

        // Set the model ID, e.g., Amazon Nova Lite.
        $modelId = 'amazon.nova-lite-v1:0';

        // Start a conversation with the user message.
        $userMessage = "Describe the purpose of a 'hello world' program in one
 line.";
        $conversation = [
            [
                "role" => "user",
                "content" => [["text" => $userMessage]]
            ]
        ];

        try {
            // Send the message to the model, using a basic inference configuration.
            $response = $client->converse([
                'modelId' => $modelId,
                'messages' => $conversation,
                'inferenceConfig' => [
                    'maxTokens' => 512,
                    'temperature' => 0.5
                ]
            ]);

            // Extract and return the response text.
            $responseText = $response['output']['message']['content'][0]['text'];
            return $responseText;
        } catch (AwsException $e) {
            echo "ERROR: Can't invoke {$modelId}. Reason: {$e-
>getAwsErrorMessage()}";
```

```
            throw new RuntimeException("Failed to invoke model: " . $e-
>getAwsErrorMessage(), 0, $e);
        }
    }
}

$demo = new Converse();
echo $demo->converse();
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 Converse。

Amazon Titan Image Generator

InvokeModel

下列程式碼範例示範如何在 Amazon Bedrock 上叫用 Amazon Titan Image 以產生映像。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

使用 Amazon Titan Image Generator 建立映像。

```
    public function invokeTitanImage(string $prompt, int $seed)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and default values for Titan Image models refer
 to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html

        $base64_image_data = "";
        try {
            $modelId = 'amazon.titan-image-generator-v1';
```

```
            $request = json_encode([
                'taskType' => 'TEXT_IMAGE',
                'textToImageParams' => [
                    'text' => $prompt
                ],
                'imageGenerationConfig' => [
                    'numberOfImages' => 1,
                    'quality' => 'standard',
                    'cfgScale' => 8.0,
                    'height' => 512,
                    'width' => 512,
                    'seed' => $seed
                ]
            ]);
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => $request,
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->images[0];
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 InvokeModel。

Anthropic Claude

InvokeModel

下列程式碼範例示範如何使用調用模型 API，將文字訊息傳送至 Anthropic Claude。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

叫用 Anthropic Claude 2 基礎模型來產生文字。

```
    public function invokeClaude($prompt)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and default values for Anthropic Claude, refer
 to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
claude.html

        $completion = "";
        try {
            $modelId = 'anthropic.claude-3-haiku-20240307-v1:0';
        // Claude requires you to enclose the prompt as follows:
            $body = [
                'anthropic_version' => 'bedrock-2023-05-31',
                'max_tokens' => 512,
                'temperature' => 0.5,
                'messages' => [[
                    'role' => 'user',
                    'content' => $prompt
                ]]
            ];
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $completion = $response_body->content[0]->text;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }
```

```
        return $completion;
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 InvokeModel。

Stable Diffusion

InvokeModel

下列程式碼範例示範如何在 Amazon Bedrock 上調用 Stability.ai 穩定擴散 XL 來產生映像。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

建立具有穩定擴散的影像。

```
    public function invokeStableDiffusion(string $prompt, int $seed, string
 $style_preset)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available style_presets of Stable Diffusion
 models refer to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html

        $base64_image_data = "";
        try {
            $modelId = 'stability.stable-diffusion-xl-v1';
            $body = [
                'text_prompts' => [
                    ['text' => $prompt]
                ],
                'seed' => $seed,
                'cfg_scale' => 10,
```

```
                'steps' => 30
            ];
            if ($style_preset) {
                $body['style_preset'] = $style_preset;
            }

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->artifacts[0]->base64;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 InvokeModel。

使用適用於 PHP 的 SDK 的 Amazon DocumentDB 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon DocumentDB 執行動作和實作常見案例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 無伺服器範例

無伺服器範例

使用 Amazon DocumentDB 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 DocumentDB 變更串流接收記錄所觸發的事件。函數會擷取 DocumentDB 承載並記下記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 使用 Amazon DocumentDB 事件。

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
```

```
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

使用適用於 PHP 的 SDK 的 DynamoDB 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 DynamoDB 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 基本概念
- 動作
- 案例
- 無伺服器範例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立可存放電影資料的資料表。
- 放入、取得和更新資料表中的單個電影。
- 將影片資料從範例 JSON 檔案寫入資料表。
- 查詢特定年份發表的電影。
- 掃描某個年份範圍內發表的電影。
- 從資料表刪除電影，然後刪除資料表。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
```

```
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];
```

```
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
 {$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
 $rating);

        $movie = $service->getItemByKey($tableName, $key);
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a {$movie['Item']
['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
```

```
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - 查詢
  - 掃描
  - UpdateItem

動作

`BatchExecuteStatement`

以下程式碼範例顯示如何使用 `BatchExecuteStatement`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
```

```
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 BatchExecuteStatement。

BatchWriteItem

以下程式碼範例顯示如何使用 `BatchWriteItem`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function writeBatch(string $TableName, array $Batch, int $depth = 2)
    {
        if (--$depth <= 0) {
            throw new Exception("Max depth exceeded. Please try with fewer batch
items or increase depth.");
        }

        $marshal = new Marshaler();
        $total = 0;
        foreach (array_chunk($Batch, 25) as $Items) {
            foreach ($Items as $Item) {
                $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' => ['Item'
=> $marshal->marshalItem($Item)]];
            }
            try {
                echo "Batching another " . count($Items) . " for a total of " .
($total += count($Items)) . " items!\n";
                $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
                $BatchWrite = [];
            } catch (Exception $e) {
                echo "uh oh...";
                echo $e->getMessage();
                die();
            }
```

```
            if ($total >= 250) {
                echo "250 movies is probably enough. Right? We can stop there.\n";
                break;
            }
        }
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 BatchWriteItem。

CreateTable

以下程式碼範例顯示如何使用 `CreateTable`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

建立 資料表。

```
        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
 'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
```

```
                    ['AttributeName' => $attribute->AttributeName, 'AttributeType'
=> $attribute->AttributeType];
            }
        }

        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
'WriteCapacityUnits' => 10],
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateTable。

`DeleteItem`

以下程式碼範例顯示如何使用 `DeleteItem`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
```

```
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
    {
        $this->dynamoDbClient->deleteItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteItem。

DeleteTable

以下程式碼範例顯示如何使用 DeleteTable。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function deleteTable(string $TableName)
    {
        $this->customWaiter(function () use ($TableName) {
            return $this->dynamoDbClient->deleteTable([
                'TableName' => $TableName,
            ]);
        });
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteTable。

ExecuteStatement

以下程式碼範例顯示如何使用 ExecuteStatement。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 ExecuteStatement。

GetItem

以下程式碼範例顯示如何使用 GetItem。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetItem。

ListTables

以下程式碼範例顯示如何使用 ListTables。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function listTables($exclusiveStartTableName = "", $limit = 100)
{
    $this->dynamoDbClient->listTables([
        'ExclusiveStartTableName' => $exclusiveStartTableName,
        'Limit' => $limit,
    ]);
}
```

- 如需 API 的詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 ListTables。

`PutItem`

以下程式碼範例顯示如何使用 `PutItem`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    echo "What's the name of the last movie you watched?\n";
    while (empty($movieName)) {
        $movieName = testable_readline("Movie name: ");
    }
    echo "And what year was it released?\n";
    $movieYear = "year";
    while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
        $movieYear = testable_readline("Year released: ");
    }
```

```
        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

    public function putItem(array $array)
    {
        $this->dynamoDbClient->putItem($array);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 PutItem。

Query

以下程式碼範例顯示如何使用 `Query`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
```

```
    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 Query。

Scan

以下程式碼範例顯示如何使用 Scan。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $yearsKey = [
```

```
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

    public function scan(string $tableName, array $key, string $filters)
    {
        $query = [
            'ExpressionAttributeNames' => ['#year' => 'year'],
            'ExpressionAttributeValues' => [
                ":min" => ['N' => '1990'],
                ":max" => ['N' => '1999'],
            ],
            'FilterExpression' => "#year between :min and :max",
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->scan($query);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 Scan。

UpdateItem

以下程式碼範例顯示如何使用 UpdateItem。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

    public function updateItemAttributeByKey(
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
            ],
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 UpdateItem。

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for PHP

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

使用多批 PartiQL 陳述式查詢資料表

以下程式碼範例顯示做法：

- 透過執行多個 SELECT 陳述式取得一批項目。
- 透過執行多個 INSERT 陳述式新增一批項目。
- 透過執行多個 UPDATE 陳述式更新一批項目。
- 透過執行多個 DELETE 陳述式刪除一批項目。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
```

```
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";
```

```
        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']['S']}
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
```

```
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
```

```
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 BatchExecuteStatement。

使用 PartiQL 查詢資料表

以下程式碼範例顯示做法：

- 透過執行 SELECT 陳述式取得項目。
- 透過執行 INSERT 陳述式新增項目。
- 透過執行 UPDATE 陳述式更新項目。
- 透過執行 DELETE 陳述式刪除項目。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
```

```
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQL($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
```

```
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);

        $movie = $service->getItemByPartiQL($tableName, $key);
```

```
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
 {$movie['Items'][0]['rating']['N']}\n";

        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
```

```
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
 $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
```

```
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 ExecuteStatement。

無伺服器範例

使用 DynamoDB 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 DynamoDB 串流接收記錄所觸發的事件。函數會擷取 DynamoDB 承載並記下記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 DynamoDB 事件。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 DynamoDB 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為從 DynamoDB 串流接收事件的 Lambda 函數實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 報告 DynamoDB 批次項目失敗。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
```

```
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用適用於 PHP 的 SDK 的 Amazon EC2 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon EC2 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 動作

動作

CreateVpc

以下程式碼範例顯示如何使用 `CreateVpc`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /**
     * @param string $cidr
     * @return array
     */
    public function createVpc(string $cidr): array
    {
        try {
            $result = $this->ec2Client->createVpc([
                "CidrBlock" => $cidr,
            ]);
            return $result['Vpc'];
        }catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 CreateVpc。

CreateVpcEndpoint

以下程式碼範例顯示如何使用 `CreateVpcEndpoint`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /**
     * @param string $serviceName
     * @param string $vpcId
     * @param array $routeTableIds
     * @return array
     */
    public function createVpcEndpoint(string $serviceName, string $vpcId, array
 $routeTableIds): array
    {
        try {
            $result = $this->ec2Client->createVpcEndpoint([
                'ServiceName' => $serviceName,
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);

            return $result["VpcEndpoint"];
        } catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 CreateVpcEndpoint。

`DeleteVpc`

以下程式碼範例顯示如何使用 `DeleteVpc`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /**
     * @param string $vpcId
     * @return void
     */
    public function deleteVpc(string $vpcId)
    {
        try {
            $this->ec2Client->deleteVpc([
                "VpcId" => $vpcId,
            ]);
        }catch(Ec2Exception $caught){
            echo "There was a problem deleting the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 DeleteVpc。

`DeleteVpcEndpoints`

以下程式碼範例顯示如何使用 `DeleteVpcEndpoints`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /**
     * @param string $vpcEndpointId
     * @return void
     */
    public function deleteVpcEndpoint(string $vpcEndpointId)
    {
        try {
            $this->ec2Client->deleteVpcEndpoints([
                "VpcEndpointIds" => [$vpcEndpointId],
            ]);
        }catch (Ec2Exception $caught){
            echo "There was a problem deleting the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 DeleteVpcEndpoints。

`DescribeRouteTables`

以下程式碼範例顯示如何使用 `DescribeRouteTables`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /**
     * @param array $routeTableIds
     * @param array $filters
     * @return array
     */
```

```
    public function describeRouteTables(array $routeTableIds = [], array $filters =
[]): array
    {
        $parameters = [];
        if($routeTableIds){
            $parameters['RouteTableIds'] = $routeTableIds;
        }
        if($filters){
            $parameters['Filters'] = $filters;
        }
        try {
            $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
$parameters);
            $contents = [];
            foreach ($paginator as $result) {
                foreach ($result['RouteTables'] as $object) {
                    $contents[] = $object['RouteTableId'];
                }
            }
        }catch (Ec2Exception $caught){
            echo "There was a problem paginating the results of DescribeRouteTables:
{$caught->getAwsErrorMessage()}\n";
            throw $caught;
        }
        return $contents;
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 DescribeRouteTables。

AWS Glue 使用適用於 PHP 的 SDK 的範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 來執行動作和實作常見案例 AWS Glue。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 基本概念
- 動作

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立網路爬取公有 Amazon S3 儲存貯體的爬蟲程式，以及產生 CSV 格式中繼資料的資料庫。
- 列出 中資料庫和資料表的相關資訊 AWS Glue Data Catalog。
- 建立從 S3 儲存貯體中擷取 CSV 資料的任務、轉換資料，以及將 JSON 格式的輸出載入至另一個 S3 儲存貯體。
- 列出任務執行的相關資訊、檢視已轉換的資料以及清除資源。

如需詳細資訊，請參閱教學課程： AWS Glue Studio 入門。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
```

```
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
```

```
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
```

```
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
```

```
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
 $databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
```

```
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
```

```
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
```

```
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
 Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

動作

`CreateCrawler`

以下程式碼範例顯示如何使用 `CreateCrawler`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $crawlerName = "example-crawler-test-" . $uniqid;
```

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateCrawler。

`CreateJob`

以下程式碼範例顯示如何使用 `CreateJob`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");
```

```
        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateJob。

DeleteCrawler

以下程式碼範例顯示如何使用 `DeleteCrawler`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
```

```
    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteCrawler。

DeleteDatabase

以下程式碼範例顯示如何使用 DeleteDatabase。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteDatabase。

DeleteJob

以下程式碼範例顯示如何使用 DeleteJob。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteJob。

`DeleteTable`

以下程式碼範例顯示如何使用 `DeleteTable`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }
```

```
    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteTable。

GetCrawler

以下程式碼範例顯示如何使用 GetCrawler。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetCrawler。

GetDatabase

以下程式碼範例顯示如何使用 GetDatabase。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    $databaseName = "doc-example-database-$uniqid";

    $database = $glueService->getDatabase($databaseName);
    echo "Found a database named " . $database['Database']['Name'] . "\n";

public function getDatabase(string $databaseName): Result
{
    return $this->customWaiter(function () use ($databaseName) {
        return $this->glueClient->getDatabase([
            'Name' => $databaseName,
        ]);
    });
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetDatabase。

GetJobRun

以下程式碼範例顯示如何使用 GetJobRun。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetJobRun。

GetJobRuns

以下程式碼範例顯示如何使用 GetJobRuns。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    $jobName = 'test-job-' . $uniqid;

    $jobRuns = $glueService->getJobRuns($jobName);

public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
{
    $arguments = ['JobName' => $jobName];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    return $this->glueClient->getJobRuns($arguments);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetJobRuns。

`GetTables`

以下程式碼範例顯示如何使用 `GetTables`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    $databaseName = "doc-example-database-$uniqid";
```

```
        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetTables。

ListJobs

以下程式碼範例顯示如何使用 `ListJobs`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
```

```
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 ListJobs。

StartCrawler

以下程式碼範例顯示如何使用 `StartCrawler`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 StartCrawler。

StartJobRun

以下程式碼範例顯示如何使用 `StartJobRun`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 StartJobRun。

使用適用於 PHP 的 SDK 的 IAM 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 IAM 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 基本概念
- 動作

基本概念

了解基本概念

下列程式碼範例示範如何建立使用者並擔任角色。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立沒有許可的使用者。
- 建立一個可授予許可的角色，以列出帳戶的 Amazon S3 儲存貯體。
- 新增政策，讓使用者擔任該角色。
- 使用暫時憑證，擔任角色並列出 Amazon S3 儲存貯體，然後清理資源。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace Iam\Basics;
```

```
require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";
```

```
$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}
```

```
$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

動作

`AttachRolePolicy`

以下程式碼範例顯示如何使用 `AttachRolePolicy`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
```

```
                'RoleName' => $roleName,
            ]);
        });
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 AttachRolePolicy。

CreatePolicy

以下程式碼範例顯示如何使用 `CreatePolicy`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
```

```
        $result = $this->customWaiter(function () use ($policyName, $policyDocument)
 {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 CreatePolicy。

CreateRole

以下程式碼範例顯示如何使用 `CreateRole`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
```

```
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 CreateRole。

`CreateServiceLinkedRole`

以下程式碼範例顯示如何使用 `CreateServiceLinkedRole`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
```

```
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 CreateServiceLinkedRole。

`CreateUser`

以下程式碼範例顯示如何使用 `CreateUser`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);
```

```
        return $result['User'];
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 CreateUser。

GetAccountPasswordPolicy

以下程式碼範例顯示如何使用 GetAccountPasswordPolicy。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 GetAccountPasswordPolicy。

GetPolicy

以下程式碼範例顯示如何使用 GetPolicy。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 GetPolicy。

`GetRole`

以下程式碼範例顯示如何使用 `GetRole`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
```

```
        });
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 GetRole。

ListAttachedRolePolicies

以下程式碼範例顯示如何使用 `ListAttachedRolePolicies`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
 "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 ListAttachedRolePolicies。

ListGroups

以下程式碼範例顯示如何使用 ListGroups。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 ListGroups。

ListPolicies

以下程式碼範例顯示如何使用 ListPolicies。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 ListPolicies。

`ListRolePolicies`

以下程式碼範例顯示如何使用 `ListRolePolicies`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 ListRolePolicies。

ListRoles

以下程式碼範例顯示如何使用 `ListRoles`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
```

```
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 ListRoles。

`ListSAMLProviders`

以下程式碼範例顯示如何使用 `ListSAMLProviders`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 ListSAMLProviders。

ListUsers

以下程式碼範例顯示如何使用 `ListUsers`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API Reference 中的 ListUsers。

使用適用於 PHP 的 SDK 的 Kinesis 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Kinesis 執行動作和實作常見案例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 無伺服器範例

無伺服器範例

使用 Kinesis 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 Kinesis 串流接收記錄所觸發的事件。此函數會擷取 Kinesis 承載、從 Base64 解碼，並記錄記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 Kinesis 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 Kinesis 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為從 Kinesis 串流接收事件的 Lambda 函數實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來報告 Kinesis 批次項目失敗。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
```

```
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

AWS KMS 使用適用於 PHP 的 SDK 的範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 來執行動作和實作常見案例 AWS KMS。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

開始使用

您好 AWS Key Management Service

下列程式碼範例示範如何開始使用 AWS Key Management Service。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get the
 keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or to
 break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 ListKeys。

主題

- 基本概念
- 動作

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立 KMS 金鑰。
- 列出您帳戶的 KMS 金鑰，並取得其詳細資訊。
- 啟用和停用 KMS 金鑰。
- 產生可用於用戶端加密的對稱資料金鑰。
- 產生用於數位簽署資料的非對稱金鑰。
- 標籤索引鍵。
- 刪除 KMS 金鑰。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo "\n";
        echo "-------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using the
 AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service that
 allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys, making
 it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
```

```
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "--------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
        $this->kmsService = new KmsService($this->kmsClient);

        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
        echo "First, we will create a symmetric KMS key that is used to encrypt and
 decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code checks
 to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();
```

```
        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt data.
\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
```

```
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days) from
 the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key: {$key['KeyId']}\n";
        $this->pressEnter();

        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon Web
 Services accounts, IAM users, IAM roles, federated users, and assumed role users.
 For help with the ARN syntax for a principal, see IAM ARNs in the Identity and
 Access Management User Guide. \nTo skip this step, press enter without any other
 values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
```

```
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted earlier
 in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
 $encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are the
 primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in the
 key policy determine who has permission to use the KMS key and how they can use it.
\n";
        echo " You can also use IAM policies and grants to control access to the KMS
 key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
```

```
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
 screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();

        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
 make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
 compliance features provided by AWS, which can help you meet various regulatory
 requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
```

```
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text, $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
and governance of your KMS keys, making it easier to organize, track, and control
access to your encrypted data within your AWS environment.\n";
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
changes to PendingDeletion and the key can't be used in any cryptographic
operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
operation. When a KMS key is deleted, all data that was encrypted under the KMS key
is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources created
during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
```

```
            $this->cleanUp();
        }

        echo
 "--------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
        echo
 "--------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
```

```
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
"SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
 enabled.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
```

```
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
```

```
   public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
   {
       $args = [
           'KeyId' => $keyId,
           'GranteePrincipal' => $granteePrincipal,
           'Operations' => $operations,
       ];
       if($grantTokens){
           $args['GrantTokens'] = $grantTokens;
       }
       try{
           return $this->client->createGrant($args);
       }catch(KmsException $caught){
           if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
               echo "The request was rejected because the specified grant token is
not valid.\n";
           }
           throw $caught;
       }
   }



   /***
    * @param string $keyId
    * @return array
    */
   public function describeKey(string $keyId)
   {
       try {
           $result = $this->client->describeKey([
               "KeyId" => $keyId,
           ]);
           return $result['KeyMetadata'];
       }catch(KmsException $caught){
           if($caught->getAwsErrorMessage() == "NotFoundException"){
               echo "The request was rejected because the specified entity or
resource could not be found.\n";
           }
           throw $caught;
       }
   }
```

```
    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
```

```
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
```

```
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
```

```
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
```

```
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
 365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
```

```
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature, string
 $signingAlgorithm)
    {
        try {
```

```
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - CreateAlias
  - CreateGrant
  - CreateKey
  - 解密
  - DescribeKey
  - DisableKey
  - EnableKey
  - 加密
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - 符號
  - TagResource

動作

`CreateAlias`

以下程式碼範例顯示如何使用 `CreateAlias`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 CreateAlias。

CreateGrant

以下程式碼範例顯示如何使用 CreateGrant。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
   /***
    * @param string $keyId
    * @param string $granteePrincipal
    * @param array $operations
    * @param array $grantTokens
    * @return Result
    */
   public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
   {
       $args = [
           'KeyId' => $keyId,
           'GranteePrincipal' => $granteePrincipal,
           'Operations' => $operations,
       ];
       if($grantTokens){
           $args['GrantTokens'] = $grantTokens;
       }
       try{
           return $this->client->createGrant($args);
       }catch(KmsException $caught){
           if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
               echo "The request was rejected because the specified grant token is
not valid.\n";
           }
           throw $caught;
       }
   }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 CreateGrant。

CreateKey

以下程式碼範例顯示如何使用 `CreateKey`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
   /***
    * @param string $keySpec
    * @param string $keyUsage
    * @param string $description
    * @return array
    */
   public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
   {
       $parameters = ['Description' => $description];
       if($keySpec && $keyUsage){
           $parameters['KeySpec'] = $keySpec;
           $parameters['KeyUsage'] = $keyUsage;
       }
       try {
           $result = $this->client->createKey($parameters);
           return $result['KeyMetadata'];
       }catch(KmsException $caught){
           // Check for error specific to createKey operations
           if ($caught->getAwsErrorMessage() == "LimitExceededException"){
               echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
           }
           throw $caught;
```

```
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 CreateKey。

Decrypt

以下程式碼範例顯示如何使用 Decrypt。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
 "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的解密。

DeleteAlias

以下程式碼範例顯示如何使用 DeleteAlias。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 DeleteAlias。

DescribeKey

以下程式碼範例顯示如何使用 DescribeKey。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 DescribeKey。

DisableKey

以下程式碼範例顯示如何使用 DisableKey。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 DisableKey。

`EnableKey`

以下程式碼範例顯示如何使用 `EnableKey`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 EnableKey。

Encrypt

以下程式碼範例顯示如何使用 `Encrypt`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
```

```
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
enabled.\n";
            }
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的加密。

`ListAliases`

以下程式碼範例顯示如何使用 `ListAliases`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
```

```
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 ListAliases。

`ListGrants`

以下程式碼範例顯示如何使用 `ListGrants`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
```

```
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 ListGrants。

ListKeys

以下程式碼範例顯示如何使用 `ListKeys`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
```

```
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 ListKeys。

PutKeyPolicy

以下程式碼範例顯示如何使用 `PutKeyPolicy`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
```

```
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 PutKeyPolicy。

RevokeGrant

以下程式碼範例顯示如何使用 RevokeGrant。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《API 參考》中的 RevokeGrant。

ScheduleKeyDeletion

以下程式碼範例顯示如何使用 ScheduleKeyDeletion。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱適用於 PHP 的 AWS SDK 《 API 參考》中的 ScheduleKeyDeletion。

Sign

以下程式碼範例顯示如何使用 Sign。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱登入 適用於 PHP 的 AWS SDK API 參考。

`TagResource`

以下程式碼範例顯示如何使用 `TagResource`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 TagResource。

使用適用於 PHP 的 SDK 的 Lambda 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Lambda 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 基本概念
- 動作
- 案例
- 無伺服器範例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立 IAM 角色和 Lambda 函數，然後上傳處理常式程式碼。
- 調用具有單一參數的函數並取得結果。
- 更新函數程式碼並使用環境變數進行設定。
- 調用具有新參數的函數並取得結果。顯示傳回的執行日誌。
- 列出您帳戶的函數，然後清理相關資源。

如需詳細資訊，請參閱使用主控台建立 Lambda 函數。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;
```

```
class GettingStartedWithLambda
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
                        \"Service\": \"lambda.amazonaws.com\"
                    },
                    \"Action\": \"sts:AssumeRole\"
                }
            ]
        }";
        $role = $iamService->createRole($roleName, $rolePolicyDocument);
        echo "Created role {$role['RoleName']}.\n";

        $iamService->attachRolePolicy(
            $role['RoleName'],
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        );
        echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\n";
```

```
        echo "\nNow let's create an S3 bucket and upload our Lambda code there.\n";
        $bucketName = "test-example-bucket-$uniqid";
        $s3client->createBucket([
            'Bucket' => $bucketName,
        ]);
        echo "Created bucket $bucketName.\n";

        $functionName = "doc_example_lambda_$uniqid";
        $codeBasic = __DIR__ . "/lambda_handler_basic.zip";
        $handler = "lambda_handler_basic";
        $file = file_get_contents($codeBasic);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "Uploaded the Lambda code.\n";

        $createLambdaFunction = $lambdaService->createFunction($functionName, $role,
 $bucketName, $handler);
        // Wait until the function has finished being created.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['State'] == "Pending");
        echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";

        sleep(1);

        echo "\nOk, let's invoke that Lambda code.\n";
        $basicParams = [
            'action' => 'increment',
            'number' => 3,
        ];
        /** @var Stream $invokeFunction */
        $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
        $result = json_decode($invokeFunction->getContents())->result;
        echo "After invoking the Lambda code with the input of
 {$basicParams['number']} we received $result.\n";

        echo "\nSince that's working, let's update the Lambda code.\n";
        $codeCalculator = "lambda_handler_calculator.zip";
```

```
        $handlerCalculator = "lambda_handler_calculator";
        echo "First, put the new code into the S3 bucket.\n";
        $file = file_get_contents($codeCalculator);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "New code uploaded.\n";

        $lambdaService->updateFunctionCode($functionName, $bucketName,
 $functionName);
        // Wait for the Lambda code to finish updating.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "New Lambda code uploaded.\n";

        $environment = [
            'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
        ];
        $lambdaService->updateFunctionConfiguration($functionName,
 $handlerCalculator, $environment);
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for more
 information.\n";

        echo "Invoke the new code with some new data.\n";
        $calculatorParams = [
            'action' => 'plus',
            'x' => 5,
            'y' => 4,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
 "Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does equal
 $result.\n";
```

```
        echo "Here's the extra debug info: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nBut what happens if you try to divide by zero?\n";
        $divZeroParams = [
            'action' => 'divide',
            'x' => 5,
            'y' => 0,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "You get a |$result| result.\n";
        echo "And an error message: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nHere's all the Lambda functions you have in this Region:\n";
        $listLambdaFunctions = $lambdaService->listFunctions(5);
        $allLambdaFunctions = $listLambdaFunctions['Functions'];
        $next = $listLambdaFunctions->get('NextMarker');
        while ($next != false) {
            $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
            $next = $listLambdaFunctions->get('NextMarker');
            $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
        }
        foreach ($allLambdaFunctions as $function) {
            echo "{$function['FunctionName']}\n";
        }

        echo "\n\nAnd don't forget to clean up your data!\n";

        $lambdaService->deleteFunction($functionName);
        echo "Deleted Lambda function.\n";
        $iamService->deleteRole($role['RoleName']);
        echo "Deleted Role.\n";
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
```

```
        ]);
        echo "Deleted all objects from the S3 bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);
        echo "Deleted the bucket.\n";
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

動作

`CreateFunction`

以下程式碼範例顯示如何使用 `CreateFunction`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
        return $this->customWaiter(function () use ($functionName, $role,
 $bucketName, $handler) {
            return $this->lambdaClient->createFunction([
                'Code' => [
```

```
                    'S3Bucket' => $bucketName,
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的「CreateFunction」。

DeleteFunction

以下程式碼範例顯示如何使用 DeleteFunction。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteFunction。

GetFunction

以下程式碼範例顯示如何使用 GetFunction。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function getFunction($functionName)
{
    return $this->lambdaClient->getFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetFunction。

`Invoke`

以下程式碼範例顯示如何使用 `Invoke`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
        'LogType' => $logType,
    ]);
}
```

- 如需 API 的詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的「Invoke」。

ListFunctions

以下程式碼範例顯示如何使用 ListFunctions。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
public function listFunctions($maxItems = 50, $marker = null)
{
    if (is_null($marker)) {
        return $this->lambdaClient->listFunctions([
            'MaxItems' => $maxItems,
        ]);
    }

    return $this->lambdaClient->listFunctions([
        'Marker' => $marker,
        'MaxItems' => $maxItems,
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 ListFunctions。

UpdateFunctionCode

以下程式碼範例顯示如何使用 UpdateFunctionCode。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 UpdateFunctionCode。

`UpdateFunctionConfiguration`

以下程式碼範例顯示如何使用 `UpdateFunctionConfiguration`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
    public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
```

```
            'Environment' => $environment,
        ]);
    }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 UpdateFunctionConfiguration。

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for PHP

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

無伺服器範例

連線至 Lambda 函數中的 Amazon RDS 資料庫

下列程式碼範例示範如何實作連線至 RDS 資料庫的 Lambda 函數。該函數會提出簡單的資料庫請求並傳回結果。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 連線至 Lambda 函數中的 Amazon RDS 資料庫。

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
```

```
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
```

```
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 Kinesis 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 Kinesis 串流接收記錄所觸發的事件。此函數會擷取 Kinesis 承載、從 Base64 解碼，並記錄記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 Kinesis 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
```

```
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 DynamoDB 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 DynamoDB 串流接收記錄所觸發的事件。函數會擷取 DynamoDB 承載並記下記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 DynamoDB 事件。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
```

```
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 Amazon DocumentDB 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 DocumentDB 變更串流接收記錄所觸發的事件。函數會擷取 DocumentDB 承載並記下記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 使用 Amazon DocumentDB 事件。

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
```

```
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

使用 Amazon MSK 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 Amazon MSK 叢集接收記錄所觸發的事件。函數會擷取 MSK 承載並記下記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來取用 Amazon MSK 事件。

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
```

```
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 Amazon S3 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收透過將物件上傳至 S3 儲存貯體所觸發的事件。此函數會從事件參數擷取 S3 儲存貯體名稱和物件金鑰，並呼叫 Amazon S3 API 以擷取和記錄物件的內容類型。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 S3 事件。

```
<?php
```

```
use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
```

```
return new Handler($logger);
```

使用 Amazon SNS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 SNS 主題接收訊息所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 SNS 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
```

```
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

使用 Amazon SQS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 SQS 佇列接收訊息所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 SQS 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
```

```
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 Kinesis 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為從 Kinesis 串流接收事件的 Lambda 函數實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來報告 Kinesis 批次項目失敗。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
```

```
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 DynamoDB 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為從 DynamoDB 串流接收事件的 Lambda 函數實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 報告 DynamoDB 批次項目失敗。

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
```

```
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
```

```
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 Amazon SQS 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為從 SQS 佇列接收事件的 Lambda 函數實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 報告 SQS 批次項目失敗。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用適用於 PHP 的 SDK 的 Amazon MSK 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon MSK 執行動作和實作常見案例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 無伺服器範例

無伺服器範例

使用 Amazon MSK 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 Amazon MSK 叢集接收記錄所觸發的事件。函數會擷取 MSK 承載並記下記錄內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來取用 Amazon MSK 事件。

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
```

```
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用適用於 PHP 的 SDK 的 Amazon RDS 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon RDS 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 動作
- 案例
- 無伺服器範例

動作

`CreateDBInstance`

以下程式碼範例顯示如何使用 `CreateDBInstance`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
```

```
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateDBInstance。

`CreateDBSnapshot`

以下程式碼範例顯示如何使用 `CreateDBSnapshot`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';
```

```
try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateDBSnapshot。

DeleteDBInstance

以下程式碼範例顯示如何使用 DeleteDBInstance。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
```

```
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteDBInstance。

`DescribeDBInstances`

以下程式碼範例顯示如何使用 `DescribeDBInstances`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
```

```
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DescribeDBInstances。

案例

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼範例示範如何建立 Web 應用程式，追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES傳送報告。

SDK for PHP

示範如何使用 適用於 PHP 的 AWS SDK 建立 Web 應用程式，以使用 Amazon Simple Email Service (Amazon SES) 追蹤 Amazon RDS 資料庫中的工作項目和電子郵件報告。這個範例使用以 React.js 建置的前端與 RESTful PHP 後端互動。

- 將 React.js Web 應用程式與 AWS 服務整合。
- 列出、新增、更新和刪除 Amazon RDS 資料表中的項目。
- 使用 Amazon SES 傳送篩選工作項目的電子郵件報告。
- 使用隨附的 AWS CloudFormation 指令碼部署和管理範例資源。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

無伺服器範例

連線至 Lambda 函數中的 Amazon RDS 資料庫

下列程式碼範例示範如何實作連線至 RDS 資料庫的 Lambda 函數。該函數會提出簡單的資料庫請求並傳回結果。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 連線至 Lambda 函數中的 Amazon RDS 資料庫。

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
```

```
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
```

```
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用適用於 PHP 的 SDK 的 Amazon RDS Data Service 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon RDS Data Service 執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 案例

案例

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼範例示範如何建立 Web 應用程式，追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES傳送報告。

SDK for PHP

示範如何使用 適用於 PHP 的 AWS SDK 建立 Web 應用程式，以使用 Amazon Simple Email Service (Amazon SES) 追蹤 Amazon RDS 資料庫中的工作項目和電子郵件報告。這個範例使用以 React.js 建置的前端與 RESTful PHP 後端互動。

- 將 React.js Web 應用程式與 AWS 服務整合。
- 列出、新增、更新和刪除 Amazon RDS 資料表中的項目。
- 使用 Amazon SES 傳送篩選工作項目的電子郵件報告。
- 使用隨附的 AWS CloudFormation 指令碼部署和管理範例資源。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

使用適用於 PHP 的 SDK 的 Amazon Rekognition 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon Rekognition 執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 案例

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for PHP

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

使用適用於 PHP 的 SDK 的 Amazon S3 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon S3 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

開始使用

您好 Amazon S3

下列程式碼範例示範如何開始使用 Amazon S3。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 ListBuckets。

主題

- 基本概念
- 動作
- 案例
- 無伺服器範例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立儲存貯體並上傳檔案到該儲存貯體。
- 從儲存貯體下載物件。
- 將物件複製至儲存貯體中的子文件夾。
- 列出儲存貯體中的物件。

- 刪除儲存貯體物件和該儲存貯體。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
```

```
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
```

```
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

```
        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

動作

`CopyObject`

以下程式碼範例顯示如何使用 `CopyObject`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

簡單複製物件。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
```

```
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CopyObject。

`CreateBucket`

以下程式碼範例顯示如何使用 `CreateBucket`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

建立儲存貯體。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateBucket。

DeleteBucket

以下程式碼範例顯示如何使用 DeleteBucket。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

刪除空的儲存貯體。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteBucket。

DeleteObject

以下程式碼範例顯示如何使用 DeleteObject。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
   public function deleteObject(string $bucketName, string $fileName, array $args =
[])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
       try {
           $this->client->deleteObject($parameters);
           if ($this->verbose) {
               echo "Deleted the object named: $fileName from $bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object deletion before continuing.";
           }
           throw $exception;
       }
   }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteObject。

`DeleteObjects`

以下程式碼範例顯示如何使用 `DeleteObjects`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

刪除金鑰清單中一整組物件。

```
       $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteObjects。

GetObject

以下程式碼範例顯示如何使用 `GetObject`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

取得物件。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 GetObject。

ListObjectsV2

以下程式碼範例顯示如何使用 `ListObjectsV2`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出儲存貯體中的物件。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
```

```
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 ListObjectsV2。

PutObject

以下程式碼範例顯示如何使用 `PutObject`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

將物件上傳至儲存貯體。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 PutObject。

案例

建立預先簽章 URL

下列程式碼範例示範如何建立 Amazon S3 的預先簽章 URL 並上傳物件。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the S3
 bucket to use: ");
```

```
        $key = $this->testable_readline("Next, provide the key of an object in the
 given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();



namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
```

```
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
```

```
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
```

```
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
```

```
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
```

```
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
```

```
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
```

```
    }

}
```

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for PHP

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

無伺服器範例

使用 Amazon S3 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收透過將物件上傳至 S3 儲存貯體所觸發的事件。此函數會從事件參數擷取 S3 儲存貯體名稱和物件金鑰，並呼叫 Amazon S3 API 以擷取和記錄物件的內容類型。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 S3 事件。

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
```

```
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用適用於 PHP 的 SDK 的 S3 目錄儲存貯體範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 S3 Directory Buckets 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 基本概念

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 設定 VPC 和 VPC 端點。
- 設定政策、角色和使用者，以使用 S3 目錄儲存貯體和 S3 Express One Zone 儲存類別。
- 建立兩個 S3 用戶端。
- 建立兩個儲存貯體。
- 建立物件並將其複製。

- 示範效能差異。
- 填入儲存貯體以顯示語彙差異。
- 提示使用者查看他們是否想要清除資源。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

執行展示 Amazon S3 目錄儲存貯體和 S3 Express One Zone 基本概念的案例。

```
        echo "\n";
        echo "--------------------------------------\n";
        echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
        echo "--------------------------------------\n";

        // Change these both of these values to use a different region/availability
 zone.
        $region = "us-west-2";
        $az = "usw2-az1";

        $this->s3Service = new S3Service(new S3Client(['region' => $region]));
        $this->iamService = new IAMService(new IamClient(['region' => $region]));

        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results in
 this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute resources
 in the same AZ as your Directory
bucket.\n
INTRO;
        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
```

```
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program is
 running in an EC2 instance in the same AZ as your Directory buckets will be.\n";
        $ec2Choice = testable_readline("Are you running this in an EC2 instance
 located in the same AZ as your intended Directory buckets? Enter Y/y to setup a VPC
 Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
 $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
 ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
 $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
```

```
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();
        $file = file_get_contents(__DIR__ . "/../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
        $result = $this->cloudFormationClient->createStack([
            'StackName' => $stackName,
            'TemplateBody' => $file,
            'Capabilities' => ['CAPABILITY_IAM'],
        ]);
        $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
 ['StackName' => $stackName]);
        try {
            $waiter->promise()->wait();
        }catch(CloudFormationException $caught){
            echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
        $this->resources['stackName'] = $stackName;
        $stackInfo = $this->cloudFormationClient->describeStacks([
            'StackName' => $result['StackId'],
        ]);

        $expressUserName = "";
        $regularUserName = "";
        foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
            if ($output['OutputKey'] == "RegularUser") {
                $regularUserName = $output['OutputValue'];
            }
            if ($output['OutputKey'] == "ExpressUser") {
                $expressUserName = $output['OutputValue'];
            }
        }
        $regularKey = $this->iamService->createAccessKey($regularUserName);
        $regularCredentials = new Credentials($regularKey['AccessKeyId'],
 $regularKey['SecretAccessKey']);
        $expressKey = $this->iamService->createAccessKey($expressUserName);
        $expressCredentials = new Credentials($expressKey['AccessKeyId'],
 $expressKey['SecretAccessKey']);

        // 3. Create an additional client using the credentials with S3 Express
 permissions.
        echo "\n";
        echo "3. Create an additional client using the credentials with S3 Express
 permissions.\n";
```

```
        echo "This client is created with the credentials associated with the
user account with the S3 Express policy attached, so it can perform S3 Express
operations.\n";
        pressEnter();
        $s3RegularClient = new S3Client([
            'Region' => $region,
            'Credentials' => $regularCredentials,
        ]);
        $s3RegularService = new S3Service($s3RegularClient);
        $s3ExpressClient = new S3Client([
            'Region' => $region,
            'Credentials' => $expressCredentials,
        ]);
        $s3ExpressService = new S3Service($s3ExpressClient);
        echo "All the roles and policies were created an attached to the user. Then,
a new S3 Client and Service were created using that user's credentials.\n";
        echo "We can now use this client to make calls to S3 Express operations.
Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
Express.\n";
        pressEnter();

        // 4. Create two buckets.
        echo "\n";
        echo "3. Create two buckets.\n";
        echo "Now we will create a Directory bucket, which is the linchpin of the S3
Express One Zone service.\n";
        echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
        echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
        pressEnter();

        // Create a directory bucket. These are different from normal S3 buckets in
subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
```

```
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

        // Create a normal bucket.
        $normalBucketName = "normal-bucket-$uuid";
        $s3RegularService->createBucket($normalBucketName);
        $this->resources['normalBucketName'] = $normalBucketName;
        echo "Great! Both buckets were created.\n";
        pressEnter();

        // 5. Create an object and copy it over.
        echo "\n";
        echo "5. Create an object and copy it over.\n";
        echo "We'll create a basic object consisting of some text and upload it to
the normal bucket.\n";
        echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
        echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
        pressEnter();

        $objectKey = "basic-text-object";
        $s3RegularService->putObject($normalBucketName, $objectKey, $args = ['Body'
=> "Look Ma, I'm a bucket!"]);
        $this->resources['objectKey'] = $objectKey;

        // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
        $s3ExpressService->createSession($directoryBucketName);
        $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

        echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
        echo "Instead of validating permissions on every call as normal buckets do,
Directory buckets utilize the user credentials and session token to validate.\n";
```

```
        echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot of
time saved.\n";
        pressEnter();

        // 6. Demonstrate performance difference.
        echo "\n";
        echo "6. Demonstrate performance difference.\n";
        $downloads = 1000;
        echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
        $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
        if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
            $downloads = $downloadChoice;
        }

        // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
        $directoryStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3ExpressService->getObject($directoryBucketName, $objectKey);
        }
        $directoryEndTime = hrtime(true);
        $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

        $normalStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3RegularService->getObject($normalBucketName, $objectKey);
        }
        $normalEndTime = hrtime(true);
        $normalTimeDiff = $normalEndTime - $normalStartTime;

        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
" nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
seconds.\n";
        pressEnter();
```

```
        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a different
 manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how the
 output of ListObjects changes.\n";
        pressEnter();

        // Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService->listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
```

```
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical order,
 while the directory bucket does not. This is because the normal bucket considers
 the whole \"key\" to be the object identifies, while the directory bucket actually
 creates directories and uses the object \"key\" as a path to the object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
 Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
 created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }



namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
```

```
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }
```

```
    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
        }
    }
```

```
    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
            throw $exception;
        }
    }
```

```
   public function listObjects(string $bucketName, $start = 0, $max = 1000, array
$args = [])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
       try {
           $objects = $this->client->listObjectsV2($parameters);
           if ($this->verbose) {
               echo "Retrieved the list of objects from: $bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with list objects before continuing.";
           }
           throw $exception;
       }
       return $objects;
   }



   public function listAllObjects($bucketName, array $args = [])
   {
       $parameters = array_merge(['Bucket' => $bucketName], $args);

       $contents = [];
       $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

       foreach ($paginator as $result) {
           if($result['KeyCount'] == 0){
               break;
           }
           foreach ($result['Contents'] as $object) {
               $contents[] = $object;
           }
       }
       return $contents;
   }
```

```
    public function deleteObjects(string $bucketName, array $objects, array $args =
 [])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
 => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
```

```
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
```

```
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

使用適用於 PHP 的 SDK 的 Amazon SES 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon SES 執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 案例

案例

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼範例示範如何建立 Web 應用程式，追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES傳送報告。

SDK for PHP

示範如何使用 適用於 PHP 的 AWS SDK 建立 Web 應用程式，以使用 Amazon Simple Email Service (Amazon SES) 追蹤 Amazon RDS 資料庫中的工作項目和電子郵件報告。這個範例使用以 React.js 建置的前端與 RESTful PHP 後端互動。

- 將 React.js Web 應用程式與 AWS 服務整合。

- 列出、新增、更新和刪除 Amazon RDS 資料表中的項目。
- 使用 Amazon SES 傳送篩選工作項目的電子郵件報告。
- 使用隨附的 AWS CloudFormation 指令碼部署和管理範例資源。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

使用適用於 PHP 的 SDK 的 Amazon SNS 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon SNS 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 動作
- 案例
- 無伺服器範例

動作

`CheckIfPhoneNumberIsOptedOut`

以下程式碼範例顯示如何使用 `CheckIfPhoneNumberIsOptedOut`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#check-if-a-phone-number-has-opted-out。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 CheckIfPhoneNumberIsOptedOut。

`ConfirmSubscription`

以下程式碼範例顯示如何使用 `ConfirmSubscription`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';
```

```
try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 ConfirmSubscription。

`CreateTopic`

以下程式碼範例顯示如何使用 `CreateTopic`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the requested
 region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */
```

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-managing-topics.html#create-a-topic。
- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 CreateTopic。

`DeleteTopic`

以下程式碼範例顯示如何使用 `DeleteTopic`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 DeleteTopic。

GetSMSAttributes

以下程式碼範例顯示如何使用 GetSMSAttributes。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#get-sms-attributes。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 GetSMSAttributes。

GetTopicAttributes

以下程式碼範例顯示如何使用 GetTopicAttributes。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 GetTopicAttributes。

`ListPhoneNumbersOptedOut`

以下程式碼範例顯示如何使用 `ListPhoneNumbersOptedOut`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages from
 your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#list-opted-out-phone-numbers。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 ListPhoneNumbersOptedOut。

`ListSubscriptions`

以下程式碼範例顯示如何使用 `ListSubscriptions`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 ListSubscriptions。

ListTopics

以下程式碼範例顯示如何使用 ListTopics。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the region
 specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 ListTopics。

Publish

以下程式碼範例顯示如何使用 Publish。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-subscribing-unsubscribing-topics.html#publish-a-message-to-an-sns-topic。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的發佈。

`SetSMSAttributes`

以下程式碼範例顯示如何使用 `SetSMSAttributes`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#set-sms-attributes。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 SetSMSAttributes。

`SetTopicAttributes`

以下程式碼範例顯示如何使用 `SetTopicAttributes`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';
```

```
try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 SetTopicAttributes。

Subscribe

以下程式碼範例顯示如何使用 Subscribe。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

訂閱主題的電子郵件地址。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
```

```
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

訂閱 HTTP 端點至主題。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的訂閱。

`Unsubscribe`

以下程式碼範例顯示如何使用 `Unsubscribe`。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-subscribing-unsubscribing-topics.html#unsubscribe-from-a-topic。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的取消訂閱。

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for PHP

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

發布簡訊

下列程式碼範例示範如何使用 Amazon SNS 發佈簡訊。

SDK for PHP

> **Note**
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon SNS.
```

```
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#publish-to-a-text-message-sms-message。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的發佈。

無伺服器範例

使用 Amazon SNS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 SNS 主題接收訊息所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 SNS 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed
```

```
            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

使用適用於 PHP 的 SDK 的 Amazon SQS 範例

下列程式碼範例示範如何使用 適用於 PHP 的 AWS SDK 搭配 Amazon SQS 執行動作和實作常見案例。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- 無伺服器範例

無伺服器範例

使用 Amazon SQS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函數，以接收從 SQS 佇列接收訊息所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 SQS 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

使用 Amazon SQS 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為從 SQS 佇列接收事件的 Lambda 函數實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for PHP

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 搭配 Lambda 報告 SQS 批次項目失敗。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
```

```
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

從 第 2 版遷移 適用於 PHP 的 AWS SDK

本主題說明如何遷移程式碼以使用 適用於 PHP 的 AWS SDK 第 3 版，並說明新版本與開發套件第 2 版的差異。

> **Note**
>
> 開發套件第 2 版的基本使用模式 (如 `$result = $client->operation($params);`) 在第 3 版中並沒有變化，能讓使用者順利遷移版本。

簡介

第 3 版 適用於 PHP 的 AWS SDK 代表了改善 SDK 功能、納入超過兩年的客戶意見回饋、升級相依性、改善效能，以及採用最新 PHP 標準的重大努力。

第 3 版有哪些新功能？

第 3 版 適用於 PHP 的 AWS SDK 遵循 PSR-4 和 PSR-7 標準，並將遵循 SemVer 標準。

其他新功能包括：

- 推出中介軟體系統，可用來自訂服務用戶端行為
- 彈性化的分頁程式，可讓使用者逐一查看分頁結果
- 能夠使用 JMESPath，從結果與分頁程式物件中查詢資料
- 藉由 `'debug'` 組態選項，即可輕鬆偵錯

分離式 HTTP 層級

- 依據預設，系統會使用 Guzzle 6 傳送請求，但亦支援 Guzzle 5。
- 系統將在無法使用 cURL 的環境中運作此開發套件。
- 亦支援自訂 HTTP 處理常式。

非同步請求

- 使用者亦可採用非同步的方式來執行等待程式與分段上傳程式等功能。

- 可以透過 promises 與 coroutines，建立非同步的工作流程。
- 並行請求或批次請求的效能有所改善。

與第 2 版有何不同？

專案相依性已更新

此版本的開發套件相依性有所變更。

- 開發套件現在需要 PHP 8.1+。除此之外，開發套件程式碼中亦大量地使用 generators (產生器)。
- 我們已升級軟體開發套件以使用 Guzzle 6 （或 5)，提供軟體開發套件用來將請求傳送至 AWS 服務的基礎 HTTP 用戶端實作。Guzzle 的最新版本推出許多改良功能，包括非同步請求、可切換的 HTTP 處理常式、PSR-7 規範，以及更優異的效能等等。
- 來自 PHP-FIG (`psr/http-message`) 的 PSR-7 套件，成功定義了用來代表 HTTP 請求、HTTP 回應、URL 與串流的界面。這些界面可供開發套件與 Guzzle 使用，亦可與其他 PSR-7 合規套件保持互通性。
- Guzzle 的 PSR-7 實作套件 (`guzzlehttp/psr7`) 提供了 PSR-7 中界面的實作，以及數種頗具助益的類別與函數。因此，不論是開發套件或是 Guzzle 6，都極度倚賴此套件。
- 軟體開發套件與 Guzzle 皆使用 Guzzle 推出的 Promises/A+ 實作 (`guzzlehttp/promises`)，以提供管理非同步請求和 coroutine 的界面。最終，Guzzle 的多重 cURL HTTP 處理器會實作非封鎖式 I/O 模型，允許非同步請求，且此套件可讓使用者在該模式中進行程式設計。如需詳細資訊，請參閱第 3 適用於 PHP 的 AWS SDK 版中的 Promises。
- 開發套件會採用 PHP 推出的 JMESPath 實作 (`mtdowling/jmespath.php`)，以供使用者查詢 `Aws\Result::search()` 與 `Aws\ResultPaginator::search()` 方法的資料。如需詳細資訊，請參閱第 3 適用於 PHP 的 AWS SDK 版中的 JMESPath 表達式。

現在需要區域和版本選項

當您將任何服務的用戶端執行個體化時，請指定 `'region'` 與 `'version'` 選項。在 第 2 版中 適用於 PHP 的 AWS SDK， `'version'` 是完全選用的，有時`'region'`是選用的。然而在第 3 版中，上述兩個選項一律為必要選項。明確說明這兩個選項可讓您鎖定要為其編碼的 API 版本和 AWS 區域。建立新的 API 版本或新的 AWS 區域可用時，系統會將您與可能中斷的變更隔離，直到您準備好明確更新組態為止。

> **Note**
>
> 如果您對使用中的 API 版本沒有疑慮，則僅需將 `'version'` 選項設為 `'latest'`。不過，我們建議您明確地設定生產程式碼的 API 版本編號。
> 並非所有 服務都適用於所有 AWS 區域。如需查詢可用區域的清單，請參考區域與端點。
> 對於只能透過單一全域端點（例如 Amazon Route 53、 和 Amazon CloudFront) 提供的服務 AWS Identity and Access Management，請將用戶端的已設定區域設定為 來執行個體化`us-east-1`。

> **Important**
>
> SDK 也包含多區域用戶端，可根據做為命令參數提供的參數 (`@region`)，將請求分派至不同的 AWS 區域。這些用戶端預設使用的區域是經由提供給用戶端建構函式的 `region` 選項加以指定。

用戶端執行個體化使用 建構函數

在 第 3 版中 適用於 PHP 的 AWS SDK，您執行個體化用戶端的方式已變更。您僅需使用 `factory` 關鍵字，即可將用戶端執行個體化，不需再使用第 2 版的 new 方法。

```
use Aws\DynamoDb\DynamoDbClient;

// Version 2 style
$client = DynamoDbClient::factory([
    'region'  => 'us-east-2'
]);

// Version 3 style
$client = new DynamoDbClient([
    'region'  => 'us-east-2',
    'version' => '2012-08-10'
]);
```

> **Note**
>
> 您依然可以使用 `factory()` 方法來執行個體化用戶端。不過，系統會將該方式視為已遭取代。

用戶端組態已變更

第 3 版中的用戶端組態選項已從第 2 版稍微 適用於 PHP 的 AWS SDK 變更。如需所有支援選項的說明，請參閱第 3 適用於 PHP 的 AWS SDK 版的組態頁面。

> **Important**
>
> 在第 3 版的根層級中，`'key'` 與 `'secret'` 不再是有效的選項，但您可以將這兩個選項傳入 `'credentials'` 選項。我們這麼做的其中一個原因是阻止開發人員將 AWS 登入資料硬式編碼到專案中。

Sdk 物件

第 3 版會將 `Aws\Sdk` 物件做為替代項目 適用於 PHP 的 AWS SDK 引入 `Aws\Common\Aws`。系統會將 `Sdk` 物件做為用戶端 factory 使用，並採用該物件來管理多個用戶端的共用組態選項。

雖然軟體開發套件第 2 版的 `Aws` 類別是以類似服務定位器的方式進行運作 (該類別會一律傳回相同的用戶端執行個體)，但第 3 版的 `Sdk` 類別每次都會傳回新的用戶端執行個體。

`Sdk` 物件亦不支援與軟體開發套件第 2 版相同的組態檔案格式。該組態格式為 Guzzle 3 專屬格式，目前已淘汰。使用基本陣列將能更輕鬆地進行組態，其做法詳載於使用 Sdk 類別。

某些 API 結果已變更

為了提供 SDK 如何剖析 API 操作結果的一致性，Amazon ElastiCache、Amazon RDS 和 Amazon Redshift 現在在某些 API 回應上具有額外的包裝元素。

例如，呼叫 Amazon RDS DescribeEngineDefaultParameters 會導致版本 3 現在包含包裝 “EngineDefaults” 元素。然而在第 2 版中，此元素並不存在。

```
$client = new Aws\Rds\RdsClient([
    'region'  => 'us-west-1',
    'version' => '2014-09-01'
```

```
]);

// Version 2
$result = $client->describeEngineDefaultParameters();
$family = $result['DBParameterGroupFamily'];
$marker = $result['Marker'];

// Version 3
$result = $client->describeEngineDefaultParameters();
$family = $result['EngineDefaults']['DBParameterGroupFamily'];
$marker = $result['EngineDefaults']['Marker'];
```

下列操作在此版本中亦受到影響，因此這些操作目前在輸出結果中會納入包裝元素，如括號中所示：

- Amazon ElastiCache
  - AuthorizeCacheSecurityGroupIngress (CacheSecurityGroup)
  - CopySnapshot (Snapshot)
  - CreateCacheCluster (CacheCluster)
  - CreateCacheParameterGroup (CacheParameterGroup)
  - CreateCacheSecurityGroup (CacheSecurityGroup)
  - CreateCacheSubnetGroup (CacheSubnetGroup)
  - CreateReplicationGroup (ReplicationGroup)
  - CreateSnapshot (Snapshot)
  - DeleteCacheCluster (CacheCluster)
  - DeleteReplicationGroup (ReplicationGroup)
  - DeleteSnapshot (Snapshot)
  - DescribeEngineDefaultParameters (EngineDefaults)
  - ModifyCacheCluster (CacheCluster)
  - ModifyCacheSubnetGroup (CacheSubnetGroup)
  - ModifyReplicationGroup (ReplicationGroup)
  - PurchaseReservedCacheNodesOffering (ReservedCacheNode)
  - RebootCacheCluster (CacheCluster)
  - RevokeCacheSecurityGroupIngress (CacheSecurityGroup)
- Amazon RDS
  - AddSourceIdentifierToSubscription (EventSubscription)

- AuthorizeDBSecurityGroupIngress (DBSecurityGroup)
- CopyDBParameterGroup (DBParameterGroup)
- CopyDBSnapshot (DBSnapshot)
- CopyOptionGroup (OptionGroup)
- CreateDBInstance (DBInstance)
- CreateDBInstanceReadReplica (DBInstance)
- CreateDBParameterGroup (DBParameterGroup)
- CreateDBSecurityGroup (DBSecurityGroup)
- CreateDBSnapshot (DBSnapshot)
- CreateDBSubnetGroup (DBSubnetGroup)
- CreateEventSubscription (EventSubscription)
- CreateOptionGroup (OptionGroup)
- DeleteDBInstance (DBInstance)
- DeleteDBSnapshot (DBSnapshot)
- DeleteEventSubscription (EventSubscription)
- DescribeEngineDefaultParameters (EngineDefaults)
- ModifyDBInstance (DBInstance)
- ModifyDBSubnetGroup (DBSubnetGroup)
- ModifyEventSubscription (EventSubscription)
- ModifyOptionGroup (OptionGroup)
- PromoteReadReplica (DBInstance)
- PurchaseReservedDBInstancesOffering (ReservedDBInstance)
- RebootDBInstance (DBInstance)
- RemoveSourceIdentifierFromSubscription (EventSubscription)
- RestoreDBInstanceFromDBSnapshot (DBInstance)
- RestoreDBInstanceToPointInTime (DBInstance)
- RevokeDBSecurityGroupIngress (DBSecurityGroup)

- Amazon Redshift
  - AuthorizeClusterSecurityGroupIngress (ClusterSecurityGroup)

  - AuthorizeSnapshotAccess (Snapshot)

- CopyClusterSnapshot (Snapshot)
- CreateCluster (Cluster)
- CreateClusterParameterGroup (ClusterParameterGroup)
- CreateClusterSecurityGroup (ClusterSecurityGroup)
- CreateClusterSnapshot (Snapshot)
- CreateClusterSubnetGroup (ClusterSubnetGroup)
- CreateEventSubscription (EventSubscription)
- CreateHsmClientCertificate (HsmClientCertificate)
- CreateHsmConfiguration (HsmConfiguration)
- DeleteCluster (Cluster)
- DeleteClusterSnapshot (Snapshot)
- DescribeDefaultClusterParameters (DefaultClusterParameters)
- DisableSnapshotCopy (Cluster)
- EnableSnapshotCopy (Cluster)
- ModifyCluster (Cluster)
- ModifyClusterSubnetGroup (ClusterSubnetGroup)
- ModifyEventSubscription (EventSubscription)
- ModifySnapshotCopyRetentionPeriod (Cluster)
- PurchaseReservedNodeOffering (ReservedNode)
- RebootCluster (Cluster)
- RestoreFromClusterSnapshot (Cluster)
- RevokeClusterSecurityGroupIngress (ClusterSecurityGroup)
- RevokeSnapshotAccess (Snapshot)
- RotateEncryptionKey (Cluster)

列舉類別已移除

我們已經移除 適用於 PHP 的 AWS SDK第 2 版現存的 `Enum` 類別 (如 `Aws\S3\Enum\CannedAcl`)。Enum 屬於開發套件公有 API 中的具體類別，而該類別所含的常數可用來表示有效參數值群組。由於這些 enum 是 API 版本特有的類別，可能會隨著時間改變，或是與 PHP 保留字衝突，

致使降低效能且沒有任何助益；因此，我們決定在第 3 版中移除這些類別。如此一來，這項改變即可支援第 3 版的資料導向與 API 版本適用性。

您不該使用來自 Enum 物件的值，而是直接使用常值。例如：使用 `CannedAcl::PUBLIC_READ` 取代 `'public-read'`。

已移除精細的例外類別

出於與移除 Enums 類別相似的考量，我們也一併移除了各服務命名空間中的精細例外狀況類別 (例如 `Aws\Rds\Exception\{SpecificError}Exception`)。由於服務或操作會依據使用的 API 版本來擲出例外狀況，因此例外狀況會隨版本而異。不僅如此，系統亦無法使用指定操作所拋出的例外狀況完整清單，導致第 2 版的精細例外狀況類別無法完整呈現。

為了處理錯誤，您應該取得各項服務的例外根類別 (例如：`Aws\Rds\Exception\RdsException`)。您可以使用例外狀況的 `getAwsErrorCode()` 方法，藉此檢查特定的錯誤碼。此功能相當於取得不同的例外類別，但本版本所提供的功能不會增加開發套件的膨脹速度。

靜態外觀類別已移除

在 第 2 版中 適用於 PHP 的 AWS SDK，有一個受到 Laravel 啟發的隱藏功能，可讓您在 `Aws`類別`enableFacades()`上呼叫 ，以啟用對各種服務用戶端的靜態存取。然而，這項功能並不符合 PHP 最佳實務。因此，我們在一年前便已經停止記錄該功能，而我們更是在第 3 版中完全將之移除。從 `Aws\Sdk` 物件中擷取用戶端物件，並將這些物件做為物件執行個體來使用，而非靜態類別。

分頁程式取代疊代運算

第 2 版 適用於 PHP 的 AWS SDK 具有名為 * 迭代器* 的功能。系統會使用疊代運算物件來逐一查看分頁結果。使用者對這些物件感到不滿的原因之一便是「缺乏彈性」，因為疊代運算僅會發送每個結果的特定值。您只能透過事件接聽程式，才能擷取其他所需的結果值。

在第 3 版中，疊代運算已由分頁程式取代。兩種功能的用途十分相似，但分頁程式提供更多使用彈性。這是因為分頁程式會產生結果物件，而非擷取回應中的數值。

以下範例會示範第 2 版和第 3 版如何擷取 `S3 ListObjects` 操作的分頁結果，以便說明分頁程式與疊代運算之間的差異。

```
// Version 2
$objects = $s3Client->getIterator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($objects as $object) {
```

```
    echo $object['Key'] . "\n";
}
```

```
// Version 3
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($results as $result) {
    // You can extract any data that you want from the result.
    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
}
```

分頁程式物件的 `search()` 方法讓您能夠使用 JMESPath 表達式，更輕鬆地從結果集擷取資料。

```
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

> **Note**
>
> 為了讓使用者順利轉移至第 3 版，該版本仍然支援 `getIterator()` 方法，但我們建議您遷移程式碼以使用分頁程式。

許多更高層級的抽象概念已變更

整體而言，此版本改良或更新了許多高階抽象概念 (除用戶端之外的服務特定 helper 物件)；甚至移除部分高階抽象概念。

- 已更新:
  - Amazon S3 分段上傳的使用方式有所變更，Amazon S3 Glacier 分段上傳以類似方式變更。
  - Amazon S3 預先簽章的 URL 建立方式有所變更。
  - 採用 `Aws\S3\Sync` 類別來取代 `Aws\S3\Transfer` 命名空間。您仍可使用 `S3Client::uploadDirectory()` 與 `S3Client::downloadBucket()` 方法，但選項會有所不同。請參閱 Amazon S3 Transfer Manager 第 3 適用於 PHP 的 AWS SDK 版的文件。
  - 採用 `Aws\S3\Model\ClearBucket` 與 `Aws\S3\Model\DeleteObjectsBatch` 來取代 `Aws\S3\BatchDelete` 和 `S3Client::deleteMatchingObjects()`。

  - 搭配第 3 適用於 PHP 的 AWS SDK 版使用 DynamoDB 工作階段處理常式的選項和行為已稍微變更。
  - 採用 `Aws\DynamoDb\Model\BatchRequest` 來取代 `Aws\DynamoDb\WriteRequestBatch` 命名空間。請參閱 DynamoDB WriteRequestBatch 文件。
  - `Aws\Ses\SesClient` 現可在使用 `SendRawEmail` 操作時處理 base64 以對 `RawMessage` 進行編碼。
- 已移除：
  - Amazon DynamoDB`ItemAttribute`、 和 `ItemIterator`類別 - 這些先前已在 2.7.0 版中棄用。
  - Amazon SNS 訊息驗證程式 - 這是獨立的輕量型專案，不需要 SDK 做為相依性。然而，此專案會包含在開發套件的 Phar 與 ZIP 分發中。您可以在 AWS PHP Development 部落格中找到入門指南。
  - Amazon S3`AcpBuilder` 和相關物件已移除。

比較兩種 SDK 版本的程式碼範例

下列範例顯示使用 第 3 適用於 PHP 的 AWS SDK 版與第 2 版的一些不同之處。

範例：Amazon S3 **`ListObjects`**操作

從 SDK 第 2 版開始

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$s3 = S3Client::factory([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key'    => 'my-object-key'
```

```
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

從 SDK 第 3 版

主要差異：

- 使用 `new` 將用戶端執行個體化，而非使用 `factory()`。
- 需要指定 `'version'` 與 `'region'` 選項，才能執行個體化。

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$s3 = new S3Client([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1',
    'version' => '2006-03-01'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

範例：使用全域組態執行個體化用戶端

從 SDK 第 2 版開始

```
<?php return array(
    'includes' => array('_aws'),
    'services' => array(
        'default_settings' => array(
            'params' => array(
                'profile' => 'my_profile',
                'region'  => 'us-east-1'
            )
        ),
        'dynamodb' => array(
            'extends' => 'dynamodb',
            'params' => array(
                'region'  => 'us-west-2'
            )
        ),
    )
);
```

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\Common\Aws;

$aws = Aws::factory('path/to/my/config.php');

$sqs = $aws->get('sqs');
// Note: SQS client will be configured for us-east-1.

$dynamodb = $aws->get('dynamodb');
// Note: DynamoDB client will be configured for us-west-2.
```

從 SDK 第 3 版

主要差異：

- 使用 `Aws\Sdk` 類別，而不是 `Aws\Common\Aws`。
- 沒有組態檔案。但會使用組態陣列取而代之。

- 需要指定 'version' 選項，才能執行個體化。
- 使用 create<Service>() 方法，而不是 get('<service>')。

```
<?php

require '/path/to/vendor/autoload.php';

$sdk = new Aws\Sdk([
    'profile' => 'my_profile',
    'region' => 'us-east-1',
    'version' => 'latest',
    'DynamoDb' => [
        'region' => 'us-west-2',
    ],
]);

$sqs = $sdk->createSqs();
// Note: Amazon SQS client will be configured for us-east-1.

$dynamodb = $sdk->createDynamoDb();
// Note: DynamoDB client will be configured for us-west-2.
```

的安全性 適用於 PHP 的 AWS SDK

雲端安全是 Amazon Web Services (AWS) 最重視的一環。身為 AWS 客戶的您，將能從資料中心和網路架構的建置中獲益，以滿足組織最為敏感的安全要求。安全性是 AWS 與您之間的共同責任。共同責任模型 將此描述為雲端本身的安全和雲端內部的安全。

Cloud 的安全 – AWS 負責保護執行 AWS 雲端中提供的所有服務的基礎設施，並提供您可以安全使用的服務。我們的安全責任是 的最高優先順序 AWS，我們的安全有效性由第三方稽核人員定期測試和驗證，作為AWS 合規計劃的一部分。

雲端的安全性 – 您的責任取決於您使用 AWS 的服務，以及其他因素，包括資料的敏感度、組織的需求，以及適用的法律和法規。

主題

- 第 3 適用於 PHP 的 AWS SDK 版中的資料保護
- 身分和存取權管理
- 此 AWS 產品或服務的合規驗證
- 此 AWS 產品或服務的彈性
- 此 AWS 產品或服務的基礎設施安全
- 第 3 版中的 Amazon S3 加密用戶端遷移 適用於 PHP 的 AWS SDK

第 3 適用於 PHP 的 AWS SDK 版中的資料保護

AWS 共同責任模型適用於 中的資料保護。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。

- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 第 3 適用於 PHP 的 AWS SDK 版，或使用主控台、API AWS CLI、 AWS SDKs的其他 AWS 服務 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

身分和存取權管理

AWS Identity and Access Management (IAM) 是 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證 （登入） 和授權 （具有許可） 來使用 AWS 資源。IAM 是您可以免費使用 AWS 服務 的 。

主題

- 目標對象
- 使用身分驗證
- 使用政策管理存取權
- AWS 服務 如何使用 IAM
- 對 AWS 身分和存取進行故障診斷

目標對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在 中執行的工作 AWS。

服務使用者 – 如果您使用 AWS 服務 執行任務，管理員會為您提供所需的登入資料和許可。當您使用更多 AWS 功能來執行工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 中的功能 AWS，請參閱 對 AWS 身分和存取進行故障診斷或 AWS 服務 您正在使用的 使用者指南。

服務管理員 – 如果您負責公司 AWS 的資源，您可能擁有 的完整存取權 AWS。您的任務是判斷服務使用者應存取哪些 AWS 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用

者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何使用 IAM AWS，請參閱您正在使用的 使用者指南 AWS 服務 。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 AWS存取權的詳細資訊。若要檢視您可以在 IAM 中使用的以 AWS 身分為基礎的政策範例，請參閱 AWS 服務 您正在使用的 使用者指南。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證 （ 登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分登入 。 AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取 時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入 的詳細資訊 AWS，請參閱AWS 登入 《 使用者指南》中的如何登入您的 AWS 帳戶 。

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的 憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的適用於 API 請求的AWS Signature 第 4 版。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的 IAM 中的AWS 多重要素驗證。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳實務是， 要求人類使用者，包括需要管理員存取權的使用者，使用 聯合身分提供者 AWS 服務 來使用臨時憑證來存取 。

聯合身分是來自您企業使用者目錄、Web 身分提供者、 AWS Directory Service、 Identity Center 目錄或任何使用透過身分來源提供的登入資料 AWS 服務 存取的使用者。當聯合身分存取時 AWS 帳戶，它們會擔任 角色，而角色會提供臨時登入資料。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連接並同步到您自己的身分來源中的一組使用者 AWS 帳戶 和群組，以便在所有 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是 中的身分 AWS 帳戶 ，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從使用者切換至 IAM 角色（主控台）。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱《IAM 使用者指南》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。

- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以將政策直接連接到資源（而不是使用角色做為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可權給 AWS 服務。
  - 服務連結角色 – 服務連結角色是一種連結至 的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，當與身分或資源相關聯時， AWS 會定義其許可。當委託人（使用者、根使用者或角色工作階段）發出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console AWS CLI、 或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交

集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。

- 服務控制政策 SCPs) – SCPs是 JSON 政策，可指定 中組織或組織單位 (OU) 的最大許可 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶 之多個 的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs的詳細資訊，包括支援 RCPs AWS 服務 的 清單，請參閱AWS Organizations 《 使用者指南》中的資源控制政策 (RCPs)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

AWS 服務 如何使用 IAM

若要全面了解 如何使用 AWS 服務 大多數 IAM 功能，請參閱《AWS IAM 使用者指南》中的使用 IAM 的 服務。

若要了解如何 AWS 服務 搭配 IAM 使用特定 ，請參閱相關服務使用者指南的安全章節。

對 AWS 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 AWS 和 IAM 時可能遇到的常見問題。

主題

- 我無權在 中執行動作 AWS
- 我未獲得執行 iam:PassRole 的授權

- 我想要允許 以外的人員 AWS 帳戶 存取我的 AWS 資源

我無權在 中執行動作 AWS

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 awes:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 awes:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您未獲授權執行 iam:PassRole 動作，您的政策必須更新，允許您將角色傳遞給 AWS。

有些 AWS 服務 可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 marymajor 的 IAM 使用者嘗試使用主控台在 AWS中執行動作時，發生下列範例錯誤。但是，動作要求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許 以外的人員 AWS 帳戶 存取我的 AWS 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 是否 AWS 支援這些功能，請參閱 AWS 服務 如何使用 IAM。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 IAM 使用者中提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶 擁有。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

此 AWS 產品或服務的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃的範圍內，請參閱AWS 服務 合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱 AWS Compliance Programs。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱在 中下載報告 AWS Artifact。

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。AWS 提供下列資源來協助合規：

- 安全合規與治理 - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- HIPAA 合格服務參考 – 列出 HIPAA 合格服務。並非所有 AWS 服務 都符合 HIPAA 資格。
- AWS 合規資源 – 此工作手冊和指南集合可能適用於您的產業和位置。
- AWS 客戶合規指南 – 透過合規的角度了解共同責任模型。本指南摘要說明跨多個架構 （包括國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 保護 AWS 服務 和映射指南至安全控制的最佳實務。
- 《 AWS Config 開發人員指南》中的使用規則評估資源 – AWS Config 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub – 這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱「Security Hub 控制參考」。
- Amazon GuardDuty – 這會監控您的環境是否有可疑和惡意活動，以 AWS 服務 偵測對您 AWS 帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。

- AWS Audit Manager – 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險的方式，以及符合法規和業界標準的方式。

此 AWS 產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循共同責任模型。如需 AWS 服務安全資訊，請參閱AWS 服務安全文件頁面AWS ，以及合規計劃在 AWS 合規工作範圍內的服務。

此 AWS 產品或服務的彈性

AWS 全球基礎設施是以 AWS 區域 和可用區域為基礎建置。

AWS 區域 提供多個實體隔離和隔離的可用區域，這些可用區域與低延遲、高輸送量和高備援聯網連接。

透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

此 AWS 產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循共同責任模型。如需 AWS 服務安全資訊，請參閱AWS 服務安全文件頁面，以及AWS 合規計劃在 AWS 合規工作範圍內的服務。

此 AWS 產品或服務的基礎設施安全

此 AWS 產品或服務使用 受管服務，因此受到 全球網路安全的 AWS 保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取此 AWS 產品或服務。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

此 AWS 產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循共同責任模型。如需 AWS 服務安全資訊，請參閱AWS 服務安全文件頁面AWS ，以及合規計劃在 AWS 合規工作範圍內的服務。

第 3 版中的 Amazon S3 加密用戶端遷移 適用於 PHP 的 AWS SDK

本主題說明如何將應用程式從 Amazon Simple Storage Service (Amazon S3) 加密用戶端的第 1 版 (V1) 遷移到第 2 版 (V2)，並確保在整個遷移過程中的應用程式可用性。

遷移概觀

此遷移分為兩個階段：

1. 更新現有用戶端以讀取新格式。首先，將更新的 適用於 PHP 的 AWS SDK 版本部署到您的應用程式。這可讓現有的 V1 加密用戶端解密新 V2 用戶端寫入的物件。如果您的應用程式使用多個 AWS SDKs，您必須分別升級每個 SDK。

2. 將加密和解密用戶端遷移至 V2。一旦所有 V1 加密用戶端都可以讀取新的格式，您就可以將現有的加密和解密用戶端遷移到各自的 V2 版本。

更新現有用戶端以讀取新格式

V2 加密用戶端使用舊版用戶端不支援的加密演算法。遷移的第一步是將 V1 解密用戶端更新為最新的 SDK 版本。完成此步驟後，應用程式的 V1 用戶端將能夠解密 V2 加密用戶端加密的物件。請參閱以下每個 主要版本的詳細資訊 適用於 PHP 的 AWS SDK。

升級第 3 適用於 PHP 的 AWS SDK 版

第 3 版是 的最新版本 適用於 PHP 的 AWS SDK。若要完成此遷移，您必須使用 3.148.0 版或更新版本的`aws/aws-sdk-php`套件。

從命令列安裝

對於使用 Composer 安裝的專案，在 Composer 檔案中，將 SDK 套件更新至 SDK 的 3.148.0 版，然後執行下列命令。

```
composer update aws/aws-sdk-php
```

使用 Phar 或 Zip 檔案安裝

使用下列其中一種方法。請務必將更新的 SDK 檔案放在程式碼所需的位置，此位置是由要求陳述式決定。

對於使用 Phar 檔案安裝的專案，請下載更新的 檔案：aws.phar。

```
<?php
  require '/path/to/aws.phar';
?>
```

對於使用 Zip 檔案安裝的專案，請下載更新的 檔案：。

```
<?php
  require '/path/to/aws-autoloader.php';
?>
```

將加密和解密用戶端遷移至 V2

更新用戶端以讀取新的加密格式後，您可以將應用程式更新為 V2 加密和解密用戶端。下列步驟說明如何成功將程式碼從 V1 遷移至 V2。

更新至 V2 用戶端的需求

1. AWS KMS 加密內容必須傳遞至 `S3EncryptionClientV2::putObject`和 `S3EncryptionClientV2::putObjectAsync`方法。 AWS KMS 加密內容是金鑰值對的關聯陣列，您必須將其新增至 AWS KMS 金鑰加密的加密內容。如果不需要其他內容，您可以傳遞空陣列。

2. `@SecurityProfile` 必須傳入 中的 `getObject`和 `getObjectAsync`方法`S3EncryptionClientV2`。 `@SecurityProfile` 是`getObject...`方法的新必要參數。如果設定為 `'V2'`，則只能解密以 V2-compatible格式加密的物件。將此參數設定為 `'V2_AND_LEGACY'` 也允許解密以 V1-compatible格式加密的物件。若要支援遷移，請將 `@SecurityProfile`設定為 `'V2_AND_LEGACY'`。`'V2'` 僅用於新的應用程式開發。

3. （選用）在 中包含 `@KmsAllowDecryptWithAnyCmk` 參數`S3EncryptionClientV2::getObjectAsync*` `methods.`，`S3EncryptionClientV2::getObject`並已新增名為 的新參數`@KmsAllowDecryptWithAnyCmk`。將此參數設定為 可`true`啟用解密，而無需提供 KMS 金鑰。預設值為 `false`。

4. 對於使用 V2 用戶端進行解密，如果“getObject...”方法呼叫的 @KmsAllowDecryptWithAnyCmk 參數未設定為 true ，kms-key-id則必須將 KmsMaterialsProviderV2 提供給建構函數。

遷移範例

範例 1：遷移至 V2 用戶端

預遷移

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

遷移後

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

範例 2：使用 AWS KMS 搭配 kms-key-id

> **Note**
>
> 這些範例使用範例 1 中定義的匯入和變數。例如 $encryptionClient。

預遷移

```
use Aws\Crypto\KmsMaterialsProvider;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProvider(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

遷移後

```
use Aws\Crypto\KmsMaterialsProviderV2;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
```

```
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

第 3 適用於 PHP 的 AWS SDK 版常見問答集

在用戶端上可使用哪些方法？

適用於 PHP 的 AWS SDK 使用服務描述和動態魔術 __call() 方法來執行 API 操作。在用戶端的 API 文件中，提供了 web 服務用戶端可使用的方法完整清單。

我該如何處理 cURL SSL 憑證的錯誤？

如果使用過時的 CA 套件和 cURL 與 SSL，可能會發生此問題。您可以藉由更新伺服器上的 CA 套件，或是從 cURL 網站直接下載更新的 CA 套件，來解決此項問題。

根據預設， 適用於 PHP 的 AWS SDK 將使用編譯 PHP 時設定的 CA 套件。您可透過修改 `openssl.cafile` PHP .ini 組態設定 (其將設為磁碟上 CA 檔案的路徑) 以變更 PHP 所使用的預設 CA 套件。

用戶端可使用哪些 API 版本？

在建立用戶端時必須設定 `version` 選項。在每個用戶端的 API 文件頁面 (::aws-php-class:<index.html>) 上，皆提供了可用 API 版本的清單。如果您無法載入特定的 API 版本，您可能需要更新您 適用於 PHP 的 AWS SDK複本。

您可針對 “version” 組態值提供 `latest` 字串，以使用用戶端 API 提供者所能找到的最新可用 API 版本 (預設的 api_provider 將會掃描軟體開發套件的 `src/data` 目錄來尋找 API 模型)。

> **Warning**
>
> 我們不建議在生產應用程式中使用 `latest`，因為導入包含 API 更新的軟體開發套件小型更新版本，可能會中斷您的生產應用程式。

用戶端可使用哪些區域版本？

在建立用戶端時需要 `region` 選項，而此選項是使用字串值來指定。如需可用 AWS 區域和端點的清單，請參閱《》中的AWS 區域和端點 AWS 一般參考。

```
// Set the Region to the EU (Frankfurt) Region.
$s3 = new Aws\S3\S3Client([
    'region'  => 'eu-central-1',
    'version' => '2006-03-01'
]);
```

為什麼我無法上傳或下載大於 2 GB 的檔案？

由於 PHP 的整數類型帶有正負號，而許多平台使用 32 位元整數，在 32 位元的堆疊上， 適用於 PHP 的 AWS SDK 無法正確地處理大小超過 2 GB 的檔案 (「堆疊」包含 CPU、作業系統、web 伺服器和 PHP 二進位檔)。這是一項眾所皆知的 PHP 問題。如果是 Microsoft Windows，則只有 PHP 7 的版本支援 64 位元整數。

建議的解決方法是使用 64 位元 Linux 堆疊，例如 64 位元 Amazon Linux AMI，並且安裝最新版本的 PHP。

如需詳細資訊，請參閱 PHP 檔案大小：傳回的值。

我要如何查看透過線路傳輸的資料內容？

您可以在用戶端建構函式中使用 `debug` 選項取得偵錯資訊，包括透過線路傳輸的資料內容。當此選項設為 `true` 時，正在執行的命令的所有不同變異版本、傳送的請求、接收的回應和處理的結果，都會發送到 STDOUT。這包括透過線路傳送和接收的資料。

```
$s3Client = new Aws\S3\S3Client([
    'region'  => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

我要如何在請求中設定任意標頭？

透過在 `Aws\HandlerList` 或 `Aws\CommandInterface` 的 `Aws\ClientInterface` 中加入自訂的中介軟體，您可以在服務操作中加入任何任意的標頭。下列範例示範如何使用 `Aws\Middleware::mapRequest`協助程式方法將 `X-Foo-Baz`標頭新增至特定 Amazon `S3PutObject` 操作。

如需詳細資訊，請參閱 mapRequest。

我要如何簽署任意請求？

您可以使用 SDK 的 ：aws-php-class： SignatureV4 class <class-Aws.Signature.SignatureV4.html>，簽署任意 ：aws-php-class：PSR-7 請求 <class-Psr.Http.Message.RequestInterfaceRequestInterface.html>.html>。 aws-php-class SignatureV4 SignatureV4.html>

如需如何執行此操作的完整範例，請參閱使用第 3 適用於 PHP 的 AWS SDK 版簽署自訂 Amazon CloudSearch 網域請求。

我要如何在傳送命令之前先加以修改？

您可以藉由在 `Aws\CommandInterface` 或 `Aws\ClientInterface` 的 `Aws\HandlerList` 中加入自訂的中介軟體，在傳送前修改命令。以下範例示範如何在傳送命令之前，在命令中新增自訂的命令參數，這基本上就是新增預設的選項。此範例使用 `Aws\Middleware::mapCommand` helper 方法。

如需詳細資訊，請參閱 mapCommand。

什麼是 CredentialsException？

如果您在使用 `Aws\Exception\CredentialsException`時看到 適用於 PHP 的 AWS SDK，表示軟體開發套件未提供任何登入資料，而且無法在環境中找到登入資料。

如果您在沒有登入資料的情況下進行用戶端的執行個體化，則當您第一次執行服務操作時，軟體開發套件將會試著找出登入資料。它會先檢查某些特定環境變數，然後尋找執行個體描述檔登入資料，這些登入資料僅適用於設定的 Amazon EC2 執行個體。如果完全找不到或未提供登入資料，會丟出 `Aws\Exception\CredentialsException` 例外。

如果您看到此錯誤，且打算使用執行個體描述檔登入資料，則需要確保 SDK 正在執行的 Amazon EC2 執行個體已設定適當的 IAM 角色。

如果出現此項錯誤，而且您不打算使用執行個體描述檔的登入資料，則您必須向軟體開發套件提供適當的登入資料。

如需詳細資訊，請參閱第 3 適用於 PHP 的 AWS SDK 版的登入資料。

是否 適用於 PHP 的 AWS SDK 適用於 HHVM？

目前 適用於 PHP 的 AWS SDK 不在 HHVM 上執行，而且在 HHVM 中產生語意的問題解決之前， 將無法執行。

我要如何停用 SSL？

您可以藉由將用戶端工廠方法中的 `scheme` 參數設為 'http'，來停用 SSL。請務必注意，並非所有服務皆支援 `http` 存取。如需AWS 區域、端點和支援方案的 AWS 一般參考 清單，請參閱 中的區域和端點。

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

> **Warning**
>
> 由於 SSL 要求所有資料進行加密，而且需要更多的 TCP 封包來完成連線交握 (而不只是 TCP)，因此停用 SSL 可能會小幅改善效能。不過，當 SSL 停用時，透過線路傳送的所有資料都不會加密。在停用 SSL 之前，您必須仔細地考慮潛在的安全問題，以及透過網路竊聽的可能性。

我該如何處理「剖析錯誤」？

在遇到不了解的語法時，PHP 引擎將會拋出剖析錯誤。在試圖執行針對不同版本 PHP 撰寫的程式碼時，幾乎都會遇到這種狀況。

如果您遇到剖析錯誤，請檢查您的系統，並確定其符合 SDK 對第 3 適用於 PHP 的 AWS SDK 版的要求和建議。

為什麼 Amazon S3 用戶端解壓縮 gziped 檔案？

某些 HTTP 處理常式 (包括預設的 Guzzle 6 HTTP 處理常式) 預設會將壓縮的回應內容解壓縮。您可以將 decode_content HTTP 選項設定為 `false` 覆寫此行為。基於回溯相容性，這項預設無法變更，但我們建議您在 S3 用戶端層級停用內容解碼。

如需如何停用自動內容解碼的範例，請參閱 decode_content。

如何在 Amazon S3 中停用內文簽署？

若要停用內文簽署，您可以將命令物件的 `ContentSHA256` 參數設為 `Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD`。然後， 適用於 PHP 的 AWS SDK 會將其用作正式請求中的「x-amz-content-sha-256」標頭和內文檢查總和。

```
$s3Client = new Aws\S3\S3Client([
    'version' => '2006-03-01',
    'region'  => 'us-standard'
]);

$params = [
    'Bucket' => 'foo',
    'Key'    => 'baz',
    'ContentSHA256' => Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD
];

// Using operation methods creates command implicitly
$result = $s3Client->putObject($params);

// Using commands explicitly.
$command = $s3Client->getCommand('PutObject', $params);
$result = $s3Client->execute($command);
```

在 適用於 PHP 的 AWS SDK中如何處理重試機制？

適用於 PHP 的 AWS SDK 具有處理重試行為`RetryMiddleware`的 。如果是伺服器錯誤的 5xx 系列 HTTP 狀態碼，軟體開發套件會針對 500、502、503 和 504 來進行重試。

調節例外狀況 (包括 `RequestLimitExceeded`、`Throttling`、`ProvisionedThroughputExceededException`、`Throttl` 和 `BandwidthLimitExceeded`) 也會重試。

適用於 PHP 的 AWS SDK 也會將指數延遲與重試配置中的退避和抖動演算法整合。此外，除了 Amazon DynamoDB 之外3，所有服務的預設重試行為都會設定為 ，也就是 `10`。

我要如何處理具有錯誤碼的例外狀況？

除了 適用於 PHP 的 AWS SDK自訂`Exception`類別之外，每個 AWS 服務用戶端都有自己的例外類別，繼承自 AwsExceptionAwsException。根據在每項方法的 `Errors` 部分所列出的 API 特定錯誤，您可以判斷更多要捕捉 (catch) 的特定錯誤類型。

利用 的 getAwsErrorCode()`Aws\Exception\AwsException`，可取得錯誤代碼的資訊。

```
$sns = new \Aws\Sns\SnsClient([
    'region' => 'us-west-2',
    'version' => 'latest',
]);

try {
    $sns->publish([
        // parameters
        ...
    ]);
    // Do something
} catch (SnsException $e) {
    switch ($e->getAwsErrorCode()) {
        case 'EndpointDisabled':
        case 'NotFound':
            // Do something
            break;
    }
}
```

詞彙表

API 版本

服務具有一個以上的 API 版本，您使用的版本會指出哪些操作和參數有效。API 版本格式類似日期。例如，Amazon S3 的最新 API 版本為 `2006-03-01`。設定用戶端物件時，請指定一個版本。

用戶端

用戶端物件是用來為服務執行操作。開發套件中支援的每項服務都有對應的用戶端物件。用戶端物件具有與服務操作一對一的對應方法。請參閱 the section called “建立基本服務用戶端” 以開始使用用戶端。

Command

命令物件封裝了一個操作執行。當您執行提出請求區段中所述的服務操作時，您不會直接處理命令物件。命令物件可使用用戶端的 `getCommand()` 方法存取，以使用開發套件的進階功能，如並行請求和批次處理。如需詳細資訊，請參閱第 3 版指南中的 適用於 PHP 的 AWS SDK 命令物件。

處理常式

透過處理常式函數，使用者可以將命令與請求實際轉換為結果；且處理常式通常會傳送 HTTP 請求。為了增強自身行為，處理常式可以由中介軟體組成。處理常式是一個函數，它接受 `Aws\CommandInterface` 和 `Psr\Http\Message\RequestInterface` 並回傳一個以 `Aws\ResultInterface` 履行或以 `Aws\Exception\AwsException` 理由拒絕的 promise。

JMESPath

JMESPath 是一種 JSON 類似資料的查詢語言。 適用於 PHP 的 AWS SDK 使用 JMESPath 表達式來查詢 PHP 資料結構。JMESPath 表達式可經由 `Aws\Result` 方法直接用在 `Aws\ResultPaginator` 和 `search($expression)` 物件。

中介軟體

中介軟體是一種特殊類型的高階函數，可增強傳輸命令和委派給「下一個」處理常式的行為。中介軟體函數可接受 `Aws\CommandInterface` 和 `Psr\Http\Message\RequestInterface`，並回傳以 `Aws\ResultInterface` 履行或以 `Aws\Exception\AwsException` 理由拒絕的 promise。

作業

是指服務 API 中的單一操作 （例如，`CreateTable`適用於 DynamoDB，`RunInstances`適用於 Amazon EC2)。在軟體開發套件中，透過呼叫對應服務用戶端物件上的相同名稱方法來執行操作。

執行操作牽涉準備和傳送 HTTP 請求到服務並剖析回應。這個執行操作的過程是由軟體開發套件透過命令物件以抽象化。

分頁程式

有些 AWS 服務操作會分頁，並以截斷的結果回應。例如，Amazon S3 `ListObjects`的操作一次最多只會傳回 1000 個物件。諸如此類的操作需要透過字符 (或標記) 參數進行後續請求，以擷取整組結果。分頁程式是軟體開發套件的一項功能，可做為此程序的抽象表示，讓開發人員更輕鬆地使用分頁 API。它們透過用戶端的 `getPaginator()` 方法存取。如需詳細資訊，請參閱第 3 適用於 PHP 的 AWS SDK 版指南中的分頁程式。

Promise

Promise 代表非同步操作的最終結果。與 promise 互動的主要方式是透過其 then 方法，該方法會註冊回呼以接收 promise 的最終值或 promise 無法履行的理由。

區域

服務在一個以上的地理區域受支援。服務在每個區域可能有不同的端點/URL，這些端點/URL 的存在可以減少應用程式中的資料延遲。設定用戶端物件時，請提供一個區域，以便軟體開發套件可以決定將哪個端點用於該服務。

SDK

「SDK」一詞可以指整個 適用於 PHP 的 AWS SDK 程式庫，但也可以指 `Aws\Sdk`類別 (docs)，其做為每個服務之用戶端物件的工廠。`Sdk` 類別也可讓您提供一組全域組態值，套用到其建立的所有用戶端物件。

服務

參考任何 AWS 服務的一般方式 （例如 Amazon S3、Amazon DynamoDB、 AWS OpsWorks 等）。各項服務在開發套件中都有對應的用戶端物件，可支援一個以上的 API 版本。各服務亦由一項或多項操作構成其 API。服務在一個以上的區域受支援。

簽章

執行操作時，此軟體開發套件會使用您的登入資料建立您的請求數位簽章。此服務會在處理您的請求之前驗證簽章。簽章程序是由軟體開發套件封裝，並在使用您為用戶端設定的登入資料時自動發生。

等待程式

等待程式是開發套件的一項功能，可以更輕鬆地處理變更資源狀態並最終一致或非同步的操作。例如，Amazon DynamoDB`CreateTable` 操作會立即傳回回應，但資料表可能幾秒鐘都無法存取。執行等待程式允許您透過睡眠和輪詢資源狀態來等待資源進入特定狀態。等待程式使用用戶端的

`waitUntil()` 方法存取。如需詳細資訊，請參閱第 3 適用於 PHP 的 AWS SDK 版指南中的等待程式。

如需最新的 AWS 術語，請參閱 中的AWS 詞彙表 AWS 一般參考。

文件歷史紀錄

下表說明自上次發行 適用於 PHP 的 AWS SDK 開發人員指南以來的重要變更。

最近的變更：

變更	描述	日期
目錄更新	修訂了目錄，以與其他 SDK 指南更一致。	2025 年 7 月 28 日
登入資料主題修訂	主題已重新組織。為預設登入資料提供者鏈結提供更多詳細資訊。	2025 年 1 月 10 日
Amazon S3 分段上傳	記錄 'params' 陣列，可用於設定 `ObjectUploader` 和 的子命令 `MultipartUploader`	2024 年 11 月 6 日
物件上傳程式	釐清 `ObjectUploader` S3 上傳中可用回呼的使用	2024 年 10 月 11 日
更新 Amazon S3 儲存貯體名稱	已更新整個指南中的 S3 儲存貯體名稱。	2024 年 9 月 30 日
Amazon EventBridge 全域端點	新增程式碼範例，示範如何使用 Amazon EventBridge 全域端點	2023 年 12 月 22 日
AWS 一般執行期 (AWS CRT)	新增主題，討論適用於 PHP 的 SDK 使用 AWS Common Runtime (AWS CRT)。	2023 年 11 月 17 日
StreamWrapper mkdir() 更新	使用 新增使用 儲存貯體和資料夾物件的相關資訊`mkdir()`。	2023 年 11 月 2 日
建立服務用戶端	移除 'version' 參數以更新程式碼片段，因為 'latest' 是預設值。	2023 年 8 月 31 日

建立基本用戶端	移除 'version' 參數以更新程式碼片段，因為 'latest' 是預設值。	2023 年 8 月 31 日
目錄	更新了目錄，讓程式碼範例更容易存取。	2023 年 6 月 1 日
IAM 最佳實務更新	更新了指南以符合 IAM 最佳實務。如需更多詳細資訊，請參閱 IAM 中的安全最佳實務。入門的更新。	2023 年 5 月 20 日
Amazon S3 傳輸管理員	新增`add_content_md5` 轉移選項。	2023 年 4 月 13 日
Amazon S3 分段上傳	包含同步上傳的組態資訊。新增非同步`add_content_md5` 上傳的上傳選項。	2023 年 4 月 13 日
參考資訊	新增了 AWS SDKs和工具參考指南中相關詳細資訊內容的多個連結。已更新指南格式。	2022 年 9 月 14 日
一般清除	新增對 AWS SDKs和工具參考指南的參考。更新 AWS Key Management Service 章節以反映術語更新。	2022 年 8 月 23 日
使用 AWS 服務	包含 GitHub 上提供的程式碼範例清單。	2022 年 4 月 1 日
啟用 SDK 指標	移除有關啟用 SDK 指標的資訊，該指標已於 2021 年 12 月 20 日棄用。	2022 年 1 月 27 日
Amazon S3 加密用戶端遷移	新增有關 Amazon S3 加密用戶端遷移的主題	2020 年 8 月 7 日

較舊的變更：

變更	描述	發行日期
Secrets Manager 範例	新增更多服務範例	2019 年 3 月 27 日
端點探索	端點探索的組態	2019 年 2 月 15 日
Amazon CloudFront	新增更多服務範例	2019 年 1 月 25 日
服務功能	軟體開發套件指標	2018 年 1 月 11 日
Amazon Kinesis、Amazon SNS	新增更多服務範例	2018 年 12 月 14 日
Amazon SES 範例	新增更多服務範例	2018 年 10 月 5 日
AWS KMS 範例	新增更多服務範例	2018 年 8 月 8 日
登入資料	釐清和簡化登入資料指南	2018 年 6 月 30 日
MediaConvert 範例	新增更多服務範例	2018 年 6 月 15 日
新的 Web 配置	文件轉換為 AWS 樣式	2018 年 5 月 9 日
Amazon S3 加密	用戶端加密	2017 年 11 月 17 日
Amazon S3、Amazon SQS	新增更多服務範例	2017 年 3 月 26 日
Amazon S3、IAM、Amazon EC2	新增更多服務範例	2017 年 3 月 17 日
新增登入資料	新增支援 AssumeRole 和 ini	2017 年 1 月 17 日
S3 範例	S3 多區域和預先簽章的 POST	2016 年 3 月 18 日
OpenSearch Service 和 Amazon CloudSearch	新增更多服務範例	2015 年 12 月 28 日
命令列	新增命令參數	2015 年 8 月 13 日

變更	描述	發行日期
服務功能	新增 S3 和 的服務功能 AWS	2015 年 4 月 30 日
新的開發套件版本	適用於 PHP 的 AWS SDK 發行的 第 3 版。	2015 年 5 月 26 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。