DOL - 14

AWS SDK の機能をあらためて学び直そう

AWS SDK for PHP で学ぶ SDK の便利機能

Tomohiro Kamitani

アマゾンウェブサービスジャパン合同会社
技術統括本部メディアソリューション部 シニアソリューションアーキテクト

自己紹介

紙谷 知弘 (Tomohiro Kamitani)

技術統括本部 ストラテジックインダストリ技術本部
メディアソリューション部
ソリューションアーキテクト

- 普段は新聞社などのメディア系のお客様を担当
- 好きな AWS サービス :

AWS SDK　　AWS Cloud9

本セッションの内容

本日話すこと

- AWS SDK に用意されている機能を AWS SDK for PHP を通して紹介
- PHP 開発者だけでなく他言語で開発している方にも参考になる内容

本日話さないこと

- PHP を AWS 環境で扱う上のテクニックや Tips

本セッションの対象者

本セッションの対象者

- アプリケーション開発者の方
- プログラミングの基礎知識をお持ちの方

ゴール

- AWS SDK の設定オプション、機能を理解し活用方法を知る

アジェンダ

1. AWS SDK とは？
2. AWS SDK の設定オプション
3. AWS SDK で使える機能
4. Waiter デモ
5. まとめ

AWS SDK とは？

AWS SDK 使ってますか？

AWS SDK とは？

AWS SDKを学ぶと

- AWS SDK のオプションを理解することにより Timeout 値やリトライの調整などチューニングをすることができる
- AWS SDK で用意されている機能を理解することによりシンプルで効率の良いコードを書くことができる
- 一時的なリソースの作成などは AWS SDK で対応するなど Infrastructure as Code と使い分けることによって柔軟なリソース構築対応ができる

AWS SDK の設定オプション

設定オプションとは？

- 設定オプションを利用することによって、コネクションタイムアウトの秒数やリトライ回数、Proxy などを設定することができる
- クライアントによって指定できるオプションが異なる場合があるため、各クライアントのAPIドキュメントを確認
- 一部の設定オプションは環境変数またはAWS 設定ファイルに基づいてデフォルト値をチェックする

```php
$client = new S3Client([
    'http'    => [
        'connect_timeout' => 5,
        'timeout' => 5,
        'proxy' => 'http://192.168.16.1:10'
    ],
    'retries' => [
        'max_attempts' => 3,
        'mode' => 'standard'
    ]
]);
```

※AWS SDK for PHP 設定オプションの例

設定オプションの考慮事項

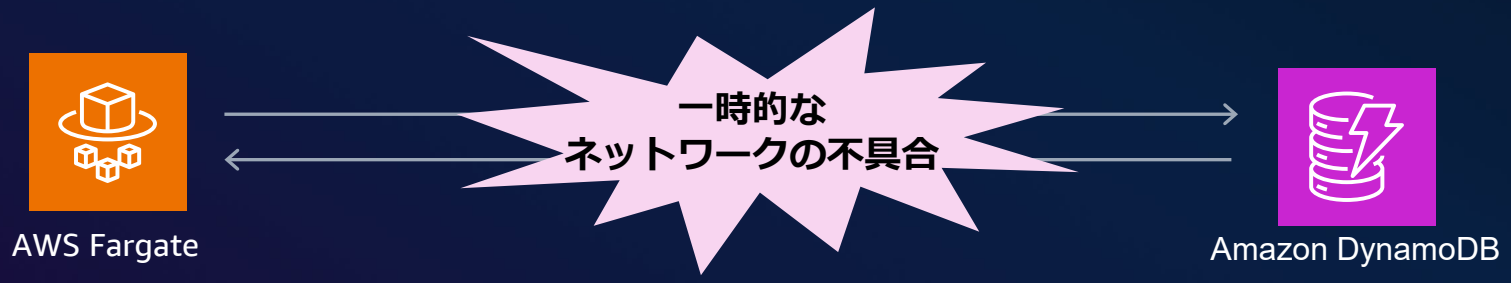
リクエストに関連する設定オプションはデフォルトから適切な値に変更

- 例えばデフォルトのコネクションタイムアウトの値は boto3 (Python) が 60秒、AWS SDK for PHP のデフォルト値が 無制限といったように言語によって異なる
- デフォルト値のままの場合、一時的なネットワークの問題が発生した際などに不要な待機時間が発生してしまう。

設定オプションの考慮事項

リクエストに関連する設定オプションはデフォルトから適切な値に変更

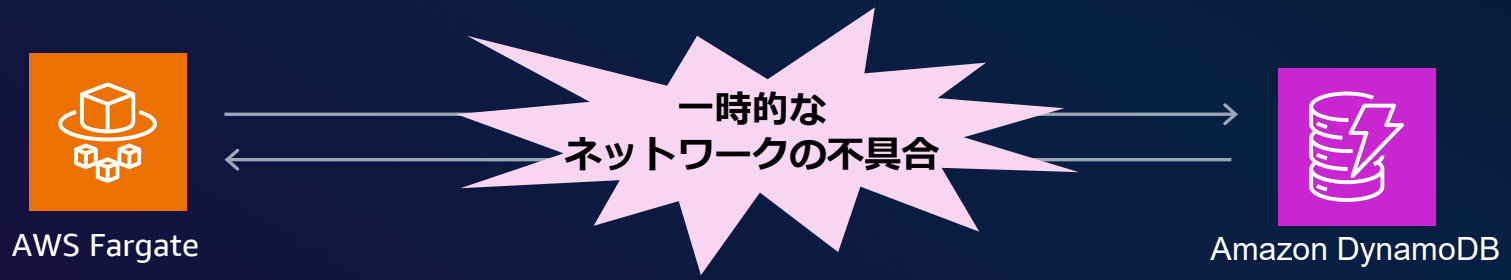
- 例えばデフォルトのコネクションタイムアウトの値は boto3 (Python) が 60秒、AWS SDK for PHP のデフォルト値が 無制限といったように言語によって異なる
- デフォルト値のままの場合、一時的なネットワークの問題が発生した際などに不要な待機時間が発生してしまう。



設定オプションの考慮事項

リクエストに関連する設定オプションはデフォルトから適切な値に変更

- 例えばデフォルトのコネクションタイムアウトの値は boto3 (Python) が 60秒、AWS SDK for PHP のデフォルト値が 無制限といったように言語によって異なる
- デフォルト値のままの場合、一時的なネットワークの問題が発生した際などに不要な待機時間が発生してしまう。

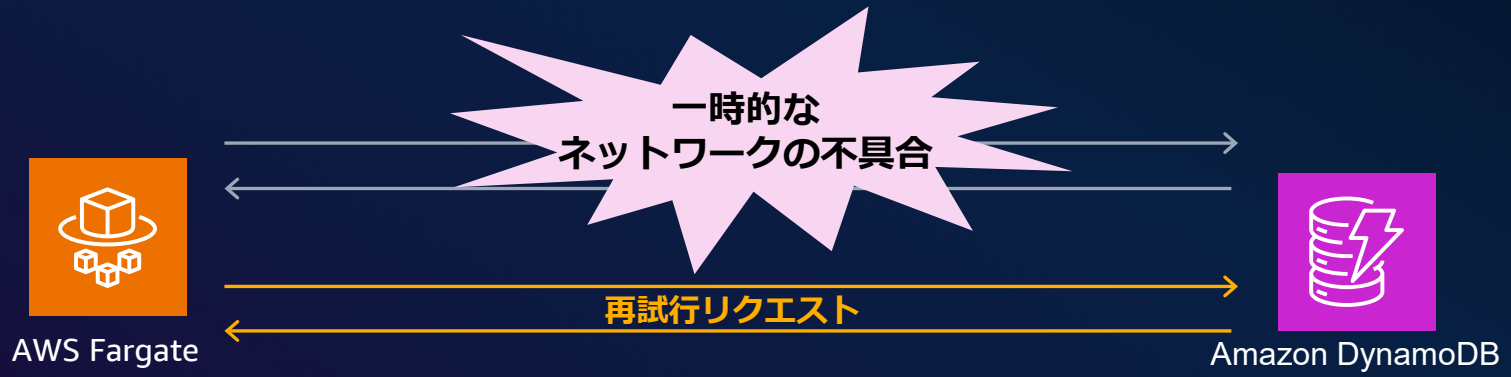


コネクションタイムアウトが設定されていない場合
デフォルトの秒数待機してからリトライ
不要な待機時間が発生！

設定オプションの考慮事項

リクエストに関連する設定オプションはデフォルトから適切な値に変更

- 例えばデフォルトのコネクションタイムアウトの値は boto3 (Python) が 60秒、AWS SDK for PHP のデフォルト値が 無制限といったように言語によって異なる
- デフォルト値のままの場合、一時的なネットワークの問題が発生した際などに不要な待機時間が発生してしまう。



適切なコネクションタイムアウトが設定されていれば
不要な待機をすることなくリトライ
一時的な不具合の影響を最小限に!

リクエストに関連する設定オプション

※ AWS SDK for PHP の例

http オプション

- connect_timeout
  - サーバーへの接続の試行時に待機する秒数
- timeout
  - リクエストのタイムアウトの秒数

retries オプション

- 最大試行回数とリトライモードが指定可能 (legacy, standard, adaptive(experimental))

```php
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'connect_timeout' => 5,
        'timeout' => 10
    ],
    'retries' => [
        'max_attempts' => 3,
        'mode' => 'standard'
    ]
]);
```

AWS SDK で使える機能

AWS SDK で用意されている機能

- AWS SDK には AWS のサービスに対するプログラミングを簡単にする機能が含まれており、利用することによってコードを簡素化できる。

- 一般的な機能に加えて特定のサービス (Amazon S3, DynamoDB等) では追加機能が提供されており、操作を簡素化することができる。

Paginator

レスポンスオブジェクトが大きく単一のリクエストで取得できない場合に

- 一部の AWS サービスオペレーションではページ分割され、結果が一部切り捨てられる
- Pagenator を利用すると簡単にページ分割されるリクエストの取得処理を実装できる

```php
$results = $s3Client->getPaginator(
    'ListObjects',
    ['Bucket' => 'my-bucket']
);

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

※AWS SDK for PHP のPagenatorの例

非同期リクエスト

並列処理で実行時間を短縮したい際に

- AWS SDK では言語によって非同期リクエストの実行をすることができる
- AWS SDK for PHP の場合は Guzzle のpromise を利用
- 非同期リクエストを利用して複数のリクエストを同時実行することができる

※ AWS SDK for PHP の場合は同時実行の実施のメカニズムとして CommandPool が用意されています

```php
$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);
$promise = $pool->promise();
$promise->wait();
```

※AWS SDK for PHP CommandPoolの例

Waiter

SDKで作成したリソースが希望する状態になるまで待機したい場合に

- Waiter を利用することによってリソースが特定の状態になるまで待機することができる
- AWS SDK for PHP では設定オプションを渡すことによって待機方法をカスタマイズすることができる

設定オプション

delay	ポーリング試行間隔の秒数
maxAttempts	発行するポーリング試行の最大回数
initDelay	最初のポーリング試行までの待機時間
before	ポーリング試行の前に呼び出す、PHPで呼び出し可能な関数を設定

AWS SDK for PHP で用意されている機能の例

Paginator	一部の AWS のサービスオペレーションではオブジェクトが大きく単一のレスポンスで返せない場合、ページ分割された結果を返す Paginator を利用することで分割された結果を簡単に取得することができる
非同期オペレーション	AWS SDK for PHP では promise を利用して非同期ワークフローを利用できる。 複数のオペレーションを同時に実行することができる
Waiter	リソースをポーリングして特定の状態になるまで待機することができる

上記の機能は他言語でも対応している機能もあるので、各種対応するドキュメントを参照してください

AWS SDK for PHP で用意されている機能の例

Paginator	一部の AWS のサービスオペレーションではオブジェクトが大きく単一のレスポンスで返せない場合、ページ分割された結果を返す Paginator を利用することで分割された結果を簡単に取得することができる
非同期オペレーション	AWS SDK for PHP では promise を利用して非同期ワークフローを利用できる。 複数のオペレーションを同時に実行することができる
Waiter	リソースをポーリングして特定の状態になるまで待機することができる

本日はこちらのデモをご紹介します

Waiter のデモ

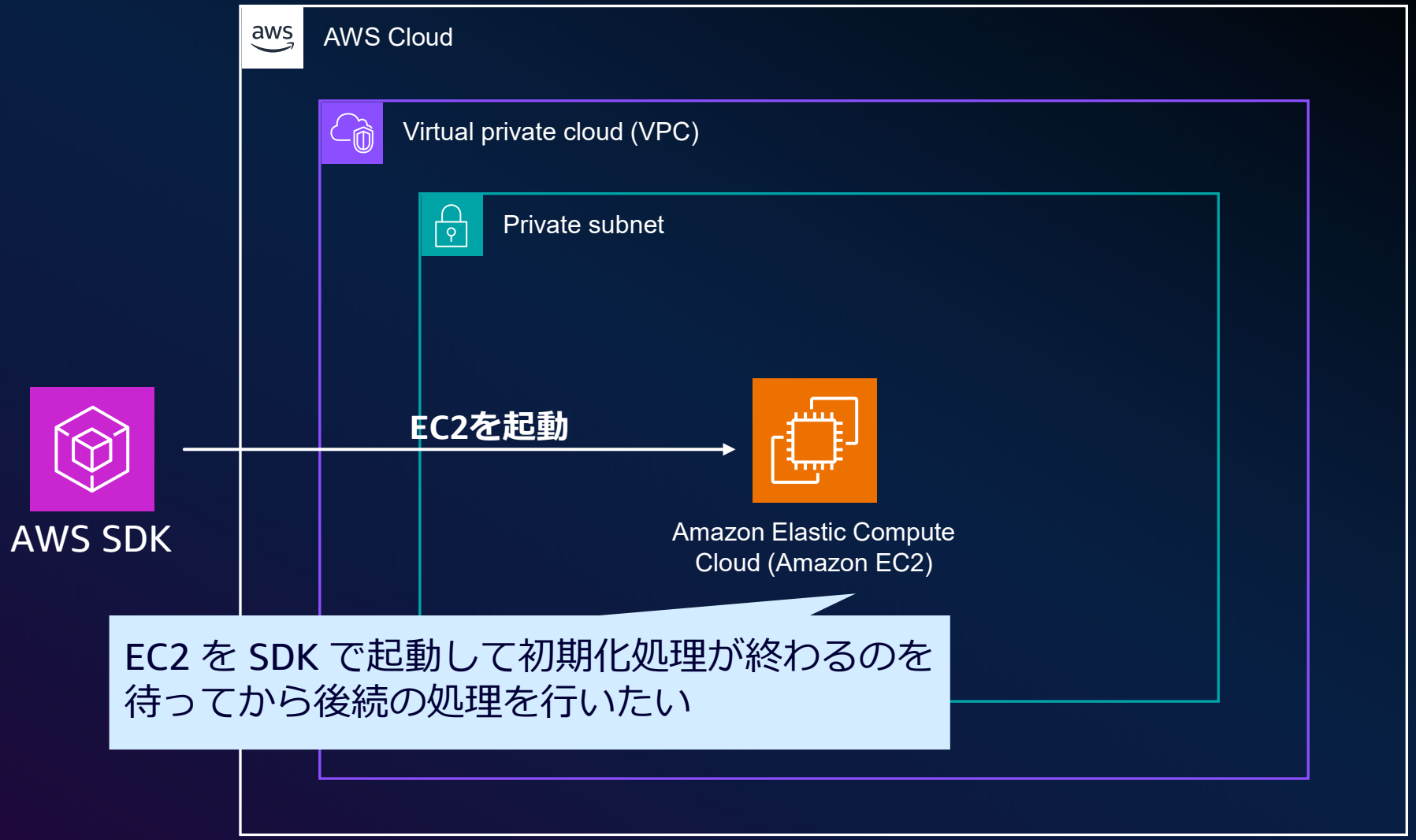
Waiter (再掲)

- Waiter を利用することによってリソースが特定の状態になるまで待機することができる
- AWS SDK for PHP では設定オプションを渡すことによって待機方法をカスタマイズすることができる

```php
$s3Client->createBucket(['Bucket' => 'my-bucket']);
// バケットが存在するまで待機
$s3Client->waitUntil('BucketExists', ['Bucket' => 'my-bucket']);
```

※AWS SDK for PHP の Waiter の例

Waiter デモの想定構成



デモ：Waiter を使わない場合

```php
$result = $ec2Client->runInstances($instanceConfig);
$instanceId = $result['Instances'][0]['InstanceId'];
echo sprintf("\e[%dm %s \e[m", 32, "インスタンスが起動しました。InstanceId: ") . $instanceId . "\n";
```

インスタンスの起動処理

```php
$result = $ec2Client->describeInstanceStatus(
    ['InstanceIds' =>[$instanceId],
    'IncludeAllInstances' => true
]);
$status = $result['InstanceStatuses'][0]['InstanceStatus']['Status'];
```

インスタンスの
ステータス取得

```php
if ($status === 'ok') {
    echo sprintf("\e[%dm %s \e[m", 32, "インスタンスの初期化処理が完了しました。Status: ") . $status . "\n";
} else {
    echo sprintf("\e[%dm %s \e[m", 33, "インスタンスの初期化処理が完了していません。Status:") . $status . "\n";
}
```

ステータスの判定処理

どのような結果になるでしょうか？

デモ：Waiter を使った場合

```php
$result = $ec2Client->runInstances($instanceConfig);
$instanceId = $result['Instances'][0]['InstanceId'];
echo sprintf("\e[%dm %s \e[m", 32, "インスタンスが起動しました。InstanceId: ") . $instanceId . "\n";
```

インスタンスの起動処理

```php
$ec2Client->waitUntil('InstanceStatusOk', [
    'InstanceIds' => [$instanceId],
    'IncludeAllInstances' => true,
    '@waiter' => [
        'before' => function (CommandInterface $command, $attempts) {
            if ( $attempts === 1) {
                echo sprintf("\e[%dm %s \e[m", 34, "インスタンスの初期化待機中:");
            }
            echo "=";
        }
    ]
]);
echo "\n";
```

リソースが希望の状態になるまでの待機処理

```php
$result = $ec2Client->describeInstanceStatus(
    ['InstanceIds' =>[$instanceId],
    'IncludeAllInstances' => true
]);
$status = $result['InstanceStatuses'][0]['InstanceStatus']['Status'];
```

インスタンスのステータス取得

```php
if ($status === 'ok') {
    echo sprintf("\e[%dm %s \e[m", 32, "インスタンスの初期化処理が完了しました。Status: ") . $status . "\n";
} else {
    echo sprintf("\e[%dm %s \e[m", 33, "インスタンスの初期化処理が完了していません。Status: ") . $status . "\n";
}
```

ステータスの判定処理

デモ：Waiter を使った場合

```php
$ec2Client->waitUntil('InstanceStatusOk', [
    'InstanceIds' => [$instanceId],
    'IncludeAllInstances' => true,
    '@waiter' => [
        'before' => function (CommandInterface $command,
            $attempts) {
            if ( $attempts === 1) {
                echo sprintf("\e[%dm %s \e[m", 34,
                    "インスタンスの初期化待機中:");
            }
            echo "=";
        }
    ]
]);
```

Waiterのまとめ

- Waiter を利用することによって、
  リソースが特定の状態になるまで待機することができる

- SDK を通して作ったリソースの特定の処理が完了するまで
  次の処理の実行を待機したい場合に利用

まとめ

Key Takeaways

- AWS SDK では設定オプションが利用できる。
  リクエスト関連のオプションがデフォルトの値の場合だと無駄な待機時間などが発生するため、
  アプリケーション特性に応じて適切な値を設定する
- AWS SDK には便利な機能が提供されており利用することによってコードを簡素化することができる
- Waiter を利用することによってリソースを特定の状態になるまで待機することができ、特定の状態になるまで後続処理を開始させたくない場合などに利用することができる

Happy Coding !

Thank you!

Tomohiro Kamitani

アマゾンウェブサービスジャパン合同会社
技術統括本部メディアソリューション部
シニアソリューションアーキテクト