AWS Migration Quick Assessment Tool

amqaプログラム仕様書

v0.93

as of 2019.1

目次

[目的 3](#_Toc516234503)

[使用するAWSサービス 3](#_Toc516234504)

[サイジング対象となるEC2インスタンス一覧 3](#_Toc516234505)

[サイジングルール 3](#_Toc516234506)

[金額算出 5](#_Toc516234507)

[事前準備 6](#_Toc516234508)

[プログラム実行環境条件 8](#_Toc516234509)

[プログラム実行方法 8](#_Toc516234510)

[プログラム実行時の注意点 8](#_Toc516234511)

[プログラム実行前のパラメータ設定 8](#_Toc516234512)

[プログラムファイル構成 9](#_Toc516234513)

[事後処理/保守 11](#_Toc516234514)

[今後の課題 11](#_Toc516234515)

[補足：全体プロセスフロー 12](#_Toc516234516)

## 目的

AWS移行簡易アセスにおいて、できるだけ手間を掛けずに顧客のサーバー一覧を分類し、EC2インスタンスへサイジングを行い、概算費用を算出する。

## ファイル一式のダウンロードサイト

https://github.com/kmorooka　　amqaツール

## 使用するAWSサービス

* QuickSight、Athena、S3、Glue、Cloud9（開発環境）、Pricing API
* QuickSightは、サーバー一覧であるCSVファイルを読み込んでカテゴライズの可視化を行う。
* Cloud9はPythonプログラムの開発・実行環境で、PythonのV3.6を利用しています。
* Athenaは、サーバー一覧のCSVファイルを読み込んでPythonプログラムからの問合せを処理
* S3、Glueは、AthenaやQuickSightへCSVファイルをロードするために利用
* 全体プロセス図は補足を参照してください。

## サイジング対象となるEC2インスタンス一覧

以下のインスタンスタイプのみがサイジングの対象です。これ以外のインスタンスタイプは現在サポートしていません。

c4, c5, m4, m5, r4, r5

上記タイプ以外のものは、例外処理用の幾つかのファイルに分類されます。（後述）

## サイジングルール

以下はサイジングルールとそれをAthenaに問い合わせるためのSQL文です。amqaプログラム実行時の第一引数は、予め準備されたこれらのルールを実行するSQL文のテキストファイルです。

サイジングの計算上、場合によってc4とc5でそれぞれ１台にカウントしてしまい紛らわしいため、c4とc5を同時に計算することは推奨しません。サイジングルールを記載するSQL文ファイルでは、必ずc４又はc５のどちらかだけの記載になるようにファイルを編集してからご利用ください。（他のインスタンスタイプm4/5, r5,6も同様です。）

c5、c5dともにコア数、メモリサイズが同じであるため、ここでは分けていません。利用時にc5、c5dを自由に選んで読み替えてください。

インスタンスタイプの選択ルール

|  |  |
| --- | --- |
| コア数に対するメモリ容量の比率（コア数：メモリ容量） | 選択するインスタンスタイプ |
| １：２≦（以下 | c　（コンピュート最適化） |
| １：２<かつ１：４≦（以下） | m （汎用） |
| １：４>（より大） | r　（メモリ最適化） |

※各タイプについては以下参照：https://aws.amazon.com/jp/ec2/instance-types/

インスタンスサイズの選択ルール

|  |  |  |
| --- | --- | --- |
| サイズ | コア数 | メモリサイズ |
| c4.Large | 1~2 | メモリ＜コア数ｘ２ |
| c4.xLarge | 3~4 | 同上 |
| c4.2xLarge | 5~8 | 同上 |
| c4.4xLarge | 9~16 | 同上 |
| c4.8xLarge | 17~36 | 同上 |
| c4.exception | 36より大 | 同上 |
| c5.large | 1~2 | メモリ≦コア数ｘ２ |
| c5.xlarge | 3~4 | 同上 |
| c5.2xlarge | 5~8 | 同上 |
| c5.4xlarge | 9~16 | 同上 |
| c5.9xlarge | 17~36 | 同上 |
| c5.18xlarge | 37~72 | 同上 |
| c5.exception | 73より大 | 同上 |
| m4.Large | 1~2 | コア数x２<メモリ≦コア数x4 |
| m4.xLarge | 3~4 | 同上 |
| m4.2xLarge | 5~8 | 同上 |
| m4.4xLarge | 9~16 | 同上 |
| m4.10xLarge | 17~40 | 同上 |
| m4.16xLarge | 41~64 | 同上 |
| m4.exception | 64より大 | 同上 |
| m5.Large | 1~2 | コア数x２<メモリ≦コア数x4 |
| m5.xLarge | 3~4 | 同上 |
| m5.2xLarge | 5~8 | 同上 |
| m5.4xLarge | 9~16 | 同上 |
| m5.12xLarge | 17~48 | 同上 |
| m5.24xLarge | 49~96 | 同上 |
| m5.exception | 96より大 | 同上 |
| r4.Large | 1~2 | コア数x4 < メモリ |
| r4.xLarge | 3~4 | 同上 |
| r4.2xLarge | 5~8 | 同上 |
| r4.4xLarge | 9~16 | 同上 |
| r4.8xLarge | 17~32 | 同上 |
| r4.16xLarge | 33~64 | 同上 |
| r4.exception | 64 より大 | 同上 |
| r5.Large | 1~2 | コア数x4 < メモリ |
| r5.xLarge | 3~4 | 同上 |
| r5.2xLarge | 5~8 | 同上 |
| r5.4xLarge | 9~16 | 同上 |
| r5.12xLarge | 17~48 | 同上 |
| r5.24xLarge | 49~96 | 同上 |
| r5.exception | 97より大 | 同上 |
| notype.nosize | 上記以外のコア数（０，65以上） | 上記以外（メモリ数が０など） |
| notype.small-mem | コア数はこだわらない | メモリサイズ＜コア数（メモリがコア数未満の場合） |

## 金額算出

* 金額算出ルール

サマリー出力ファイルにおけるEC２インスタンス金額は、AWS Pricing APIによって取得できる価格一覧を基に、東京リージョン、3年間RI、All Upfront、Shared、Standard、Linuxで算出。

またディスク価格については、EBSを前提とした各インスタンスタイプの合計ディスク容量にgp2の１Gバイト/１ヶ月の単価をかけた、更に3年(36ヶ月）にした総額費用。

なお、金額はすべてUSD(＄)となっています。

* Pricing APIによってダウンロードされるCSVファイルの保守

CSVファイルのフォーマットが変更になった場合、プログラム中の特定部分のパラメータを修正する必要があります。該当するプログラムファイルは、amqa.pyファイルで、該当関数は、「create\_pricing()」です。この関数create\_pricing()内の大文字で書かれたパラメータ群を修正することで算出金額のモデルを変更可能です。各パラメータの設定は下記のとおりです。

|  |  |  |
| --- | --- | --- |
| パラメータ名 | 内容 | 具体例 |
| PRICE |  | 'priceperunit' |
| PRICE\_POS |  | 9 |
| INSTANCETYPE |  | 'instance type' |
| INSTANCETYPE\_POS |  | 18 |
| TERMTYPE |  | 'Reserved' |
| TERMTYPE\_POS |  | 3 |
| LEASECONTRACTLENGTH |  | '3yr' |
| LENGTH\_POS |  | 11 |
| PURCHASEOPTION |  | 'All Upfront' |
| PURCHASE\_POS |  | 12 |
| LOCATION |  | 'Asia Pacific(Tokyo)' |
| LOCATION\_POS |  | 16 |
| TENANCY |  | 'Shared' |
| TENANCY\_POS |  | 35 |
| OS |  | 'Linux' |
| OS\_POS |  | 37 |
| OFFERINGCLASS |  | 'standard' |
| OFFERINGCLASS\_POS |  | 13 |
| UNIT |  | 'Quantity' |
| UNIT\_POS |  | 8 |
|  |  |  |

## 事前準備

1. 顧客からサーバー一覧をCSVファイル又はエクセルファイルで入手します。

Athena/Glue用には、CSVファイルを1つ準備します。QuickSightは、2018年春頃にエクセルファイルを直接読み込めるように機能追加されたためエクセルのままインポート可能です。

1. EC2インスタンス金額表のダウンロード

amqa\_getprice.shを実行し、ファイルを準備します。

このシェルスクリプトは、AWSから値段表をCSV形式でダウンロードし、不要なヘッダーの削除、文章中の「,」（カンマ）のスペース文字への置き換えを行い、適正なCSVファイルに修正の上、amqa-price.csvファイルとしてローカルに保存します。

amqa-price.csvファイルは、amqa.pyの第二引数に指定します。

1. CSVファイルのフォーマット

CSVファイルのフォーマットはUTF-８形式でUnix形式を準備してください。(nkfやiconvコマンド等を利用してください。) これらは、QuickSightやAthena/Glueのファイル形式に関する制約です。

UTF-8への変換コマンド　$ nkf -w source.csv > source-utf8.csv

Unix形式への変換コマンド　$ nkf -Lu source-utf8.csv > target.csv

また各項目内（例えば文章中）で項目区切りでない意味の「,」カンマが入っていると、Athena/Glueで正しく認識できませんので、予めエディタ等で取除くなどの処理を行ってください。

1. CSVファイルのヘッダー

Athenaへupload(Glue経由)するCSVへファイルは、１行目をヘッダーとしてください。複数行の場合は１行にしてください。また必ずヘッダーは全て英語でかつ、「（」カッコや「”」ダブルクォーテーション等を含まないで変換してください。漢字を含むヘッダーは文字化けして表示されます。

1. CSVファイルのカラム設定

データとして必要になるカラム項目（CPU/Mem/ディスク容量等）は、プログラムコード内で別途設定しますので、カラム項目の並び順序は気にせずにGlue経由でAthenaへuploadします。

Cloud9のPython環境のセットアップ（オプション）  
ローカルでPython環境を思料する場合は、本項目の作業は不要です。  
ここでは、Cloud9上にPython3環境を構築し、AWSを操作するためのboto3導入を行います。  
最初にPython3へ変更するためにCloud9上で下記URLを参考にし、設定変更を行ってください。Cloud9は初期設定ではPython2.7となっています。  
<https://qiita.com/sho-hitomi/items/3ca8409d9f6dd0c6a658>  
  
次に、下記コマンドを実行します。  
- .bashrc　で最後の行に追記　alias python=python36  
- $ source ./.bashrc  
- sudo update-alternatives --config python  
- 番号をきかれるのでpython36である、「２」を選択  
- pythonのバージョンを確認　$ python -V  
  
次にboto3をインストールします。  
$ sudo pip install boto3

詳細参照先：<http://awesome-linus.tk/2018/01/15/boto3-install-initialsettings/>

1. Cloud9から実行する場合（aws configureコマンド）

Cloud9では自動的にAccess Key、Secret Keyなどが振られます。これらはそのままにしておいてください。リージョンはus-west-1がデフォルト設定されています。こちらは特に変更は必要ありません。「json」はデフォルトのままです。なお、aws configureを行った後、ちょっと待つとポップアップ画面が表示され、Credentialを「Force Update」するか、「Cancel」するかを聞かれますが、「Force Update」をクリックしてください。

## プログラム実行環境条件

pythonのバージョンは3.6を使っています。

AWS API用にboto3の導入が必要です。

## プログラム実行方法

$ amqa.py amqa-sql-ec2.txt amqa-price.csv [CR]

第一引数：Athenaに対するSQL文のテキストファイル

第二引数：AWSサイトからダウンロードしたインスタンス価格のCSVファイル

## プログラム実行時の注意点

* GlueのCrawler設定で特定バケットの特定CSVファイルを指定してCrawlerを実行します。CSVファイルを指定せず、バケット名で指定した場合、対象とするCSVファイル以外が残っているとプログラム実行時にSQL文を当てはめて正しく算出が行われない、Athena画面で他のファイルの一部が一緒に表示されるなどの問題が発生します。
* AthenaではたまにSQL処理表示で画面が崩れる場合があります。その場合は、「New Query」をクリックする、他のDBを参照表示させてから該当DBに戻ってSQL実行する、等で一旦SQL文をリセットしてから実行してください。

## プログラム実行前のパラメータ設定

パラメータ設定は、amqa.pyファイルの先頭部分で行う。

以下はパラメータの説明です。

|  |  |  |
| --- | --- | --- |
| パラメータ | 説明 | サンプル |
| S3BUCKET\_NAME | amqaで使用するS3バケット名を指定します。 | 'amqa-demo' |
| REGION\_NAME | プログラム実行リージョンを指定します。 | 'ap-northeast-1' |
| FN\_RESULT | 結果サマリーファイルの名前を指定。ファイルは現行ディレクトリに作成されます。 | 'amqa-summary.csv' |
| S3WAIT\_TIME | 単位は「秒」で、S3へのファイル書込み終了タイミングを指定します。長いほど確度が高くなりますが、処理時間がかかります。 | ５(整数値：５秒であれば東京リージョンで問題が出ない程度) |
| DB\_NAME | Athena側のDB名、テーブル名を「”」付きで「.」でつないで記載します。 | '"amqa-demo"."amqa\_demo"' |
| CPU\_CORE | 顧客のCSVファイルのカラム名を転記します。Pythonプログラムで処理するため、「'」で囲います。カッコやダブルクォーテーションが含まれない容易してください。 | 'cpu' |
| MEM\_SIZE | 同上 | 'mem' |
| DISK\_COL | 顧客CSVファイル内でディスク容量の欄が、何カラム目かを指定します。最初のカラムは０(ゼロ)です。 | 9 (数値：顧客CSVをファイルで11カラム目の場合) |
| CPU\_USAGE | CPU使用率を0.0〜1.0で指定します。仮想化率を30％と仮定した場合、0.3とします。ハイエンドUnixのCPUでEC2より処理能力が30％高い場合は、1.3といった指定をします。 | 1.0 (既存ITリソースそのままの場合) |

## プログラムファイル構成

以下はプログラムファイル構成です。プログラム本体関連ファイルと出力ファイルの2種類があります。

プログラム本体関連

|  |  |
| --- | --- |
| amqa.py | プログラム本体。（Python3で開発テスト） |
| amqa\_getprice.sh | AWSから価格表をダウンロードするシェルスクリプト。ダウンロードしたCSVファイルは、文章中に「),」を含んでおり(例えば、”abc),defg"など）  、Glue等で読み込ませるとカラムがずれるため、このスクリプト内で「)」に置き換えています。将来ファイルのフォーマットが変更になった場合、このスクリプト及び、amqa.py内のcreate\_pricing()関数のパラメータを修正する必要があります。 |
| amqa-sql.txt | サイジング用AthenaへのSQL Query文 |

出力結果

* c4またはc5/c5dはどちらか選択した上でプログラム実行をするので、どちらかだけのファイル出力となるはずです。

|  |  |
| --- | --- |
| amqa-3year.csv | C4、m4、r4の全てのインスタスタイプのサイジング結果のサマリーファイル。各インスタンスタイプ毎に①インスタンス数、②インスタンスタイプの単価(USD)、③インスタンスタイプの総額、④ディスク容量(G)、⑤合計ディスク容量の総額(USD) 金額は3年合計USDです。ファイル名はamqa.py内で指定可能です。 |
| amqa-price.csv | amqa-getprice.shによってダウンロード、及びamqa.pyによって利用されるAWSインスタンスタイプの価格一覧CSVのファイル |
| c4.large.csv | 該当インスタンスタイプのサイジング結果。データがヘッダーだけの場合、該当するインスタンスタイプに対応する顧客サーバーはないことを表します。 |
| c4. large.csv | 同上 |
| c4. xlarge.csv | 同上 |
| c4. 2xlarge.csv | 同上 |
| c4. 4xlarge.csv | 同上 |
| c4. 8xlarge.csv | 同上 |
| c4.exception | サイジングレンジから外れたサーバーを集計 |
| c5.large | 該当インスタンスタイプのサイジング結果。データがヘッダーだけの場合、該当するインスタンスタイプに対応する顧客サーバーはないことを表します。 |
| c5.xlarge | 同上 |
| c5.2xlarge | 同上 |
| c5.4xlarge | 同上 |
| c5.9xlarge | 同上 |
| c5.18xlarge | 同上 |
| c5.exception | サイジングレンジから外れたサーバーを集計 |
| m4. large.csv | 該当インスタンスタイプのサイジング結果。データがヘッダーだけの場合、該当するインスタンスタイプに対応する顧客サーバーはないことを表します。 |
| m4. xlarge.csv | 同上 |
| m4. 2xlarge.csv | 同上 |
| m4. 4xlarge.csv | 同上 |
| m4. 10xlarge.csv | 同上 |
| m4. 16xlarge.csv | 同上 |
| m4.exception | サイジングレンジから外れたサーバーを集計 |
| r4. large.csv | 該当インスタンスタイプのサイジング結果。データがヘッダーだけの場合、該当するインスタンスタイプに対応する顧客サーバーはないことを表します。 |
| r4. xlarge.csv | 同上 |
| r4. 2xlarge.csv | 同上 |
| r4. 4xlarge.csv | 同上 |
| r4. 8large.csv | 同上 |
| r4. 16xlarge.csv | 同上 |
| r4.exception | サイジングレンジから外れたサーバーを集計 |
| notype.nosize.csv | コア数、メモリサイズがnullの場合 |
| notype.small-mem.csv | サイジングレンジから外れたサーバーを集計。メモリサイズ＜コア数となる場合。 |

## 事後処理/保守

1. S3側には、サイジングのSQL実行結果のファイルを一時的に保存し、amqa.py がローカルにダウンロード後、一時ファイルを削除しています。プログラム実行途中で割込などをかけたりすると、一時ファイルがS3に残る場合があります。その時は不要な一時ファイルを削除してください。

## 今後の課題

* 3/27実装済み：~~S3のアップロード先に残る処理後に不要なファイルの自動削除機能~~
* Web化し、ブラウザ経由で処理可能とする機能追加
* CPU／メモリ使用率への対応
* Athenaを使わずに直接CSVファイルからデータ集計する方式への変更
* S3へのファイル作成を確認対応し、sleep処理を削除
* QuickSightのAPIが2018/Q4で実装された場合、手動のAs−Is分析をAPIで自動化する機能

## 補足：全体プロセスフロー

