JAWS DAYS 2024

E-4全方位でのAWSコスト管理

～最適化、削減、そしてガバナンス～

株式会社QUICK E-Solutions
松浦 翼

ハッシュタグ：#jawsdays2024 #jawsug #jawsdays2024_e

自己紹介

・名前：松浦 翼 （まつうら つばさ）

・所属：株式会社QUICK E-Solutions
　　クラウドプラットフォーム本部 AWSプラットフォームグループ

・業務歴：物理サーバー保守・運用(4年)
　　＞ AWSシステムの監視運用(2年)
　　＞ AWS設計・構築(3年)

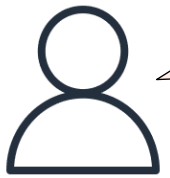
・趣味：食べ歩き（コロナ禍以降はお取り寄せグルメも）

・ひとこと：AWS re:Invent 2023 ラスベガス現地参戦してきました！

はじめに

AWSでのコスト管理での課題

とにかくコストは安く！
クラウドで従量課金なんだから簡単に安くできるんでしょ？



偉い人

・料金関連の参照権限が無い・・・
・共通リソースは部門合同だし放置でいいや・・・
・とりあえずRI/SP使っておけば納得するでしょ・・・

＼ハイワカリマシタ／

コスト管理者/
エンジニア

本セッションのポイント

本セッションでは以下の観点でのコスト管理方法をご案内します！
それぞれ、以下の方をターゲットとしています。

1.請求情報やAWS Cost Explorerが利用できない環境下での
Reserved Instance (RI)/ Savings Plans(SP)購入の判定方法
⇒ **RI/SPを購入する必要があるが、請求関連情報の参照権限が無い方向け**

2. 共通リソースから考えるコスト管理
⇒ **AWS全体のコスト管理を行う方向け**

3. 開発エンジニア目線で考えるコスト管理/削減
⇒ **AWSで設計/開発を行うエンジニアの方向け**

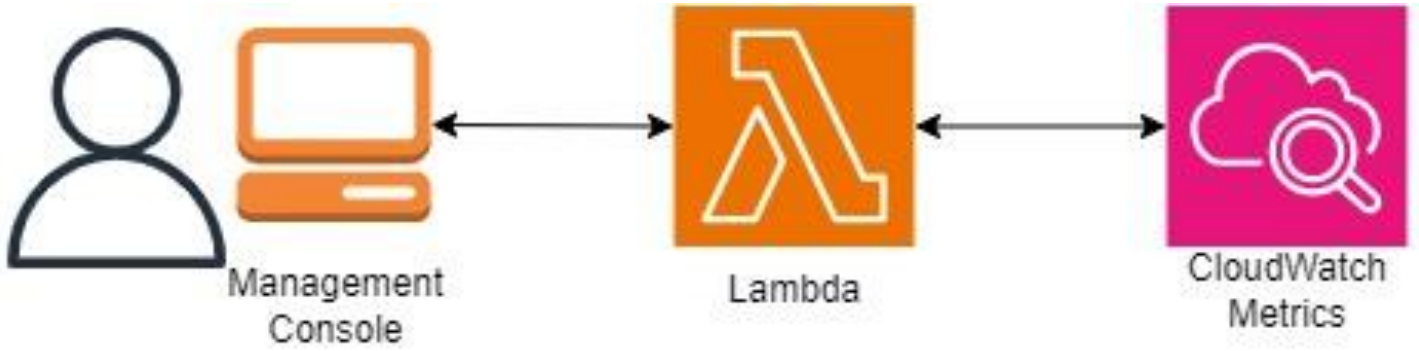
1.請求情報やAWS Cost Explorerが利用できない環境下でのRI/SP購入の判定方法

料金関連の参照権限が無い時の課題

・料金関連(請求コンソール/Cost Explorer)の参照権限が
与えられていない場合の課題

⇒ RI/SP購入を検討しているが、対象リソースの選定が困難

⇒ LambdaやAWS CLIを使用してCloudWatchのメトリクスから
稼働状況を確認し、判断基準とすることが可能です。
(以下はLambdaでの対応イメージ)



RI/SPおさらい

●RI(Reserved Instance)：
RIが購入可能なAWSサービスは以下のとおり。
・Amazon EC2
・Amazon RDS
・Amazon ElastiCache
・Amazon OpenSearch Service
・Amazon Redshift

●SP(Savings Plans)：
SPが適用可能なAWSサービスは以下のとおり。
・Amazon EC2
・AWS Fargate
・AWS Lambda
・Amazon SageMaker

RI/SPおさらい

・以下の様なリソースではRI/SPの活用を検討しましょう！
  - 長時間稼働されているもの
  - 長期的なコミットメントが予測されるもの

→ 長期的なコミットメントが予測されるかどうかはアプリケーション仕様にも依存するので、少なくとも長時間稼働しているリソースはRI/SPの適用候補になります。
請求情報が参照不可でもCloudWatchメトリクスから情報抽出して購入判断に活用しましょう！

RI(Reserved Instance)の購入判断

(EC2を例とした場合)

・Lambdaで各インスタンスごとの以下情報を取得/計算することでインスタンスが起動している時間の割合を把握し、RI購入の判断基準とすることができます。

- **直近24時間のCPU使用率のメトリクスを1時間ごとに取得**
- **CPU使用率が1%以上の時間帯の数を数え、24時間で割る**
  ※ 日々の停止/起動の運用次第で情報取得の期間は適宜調整が必要

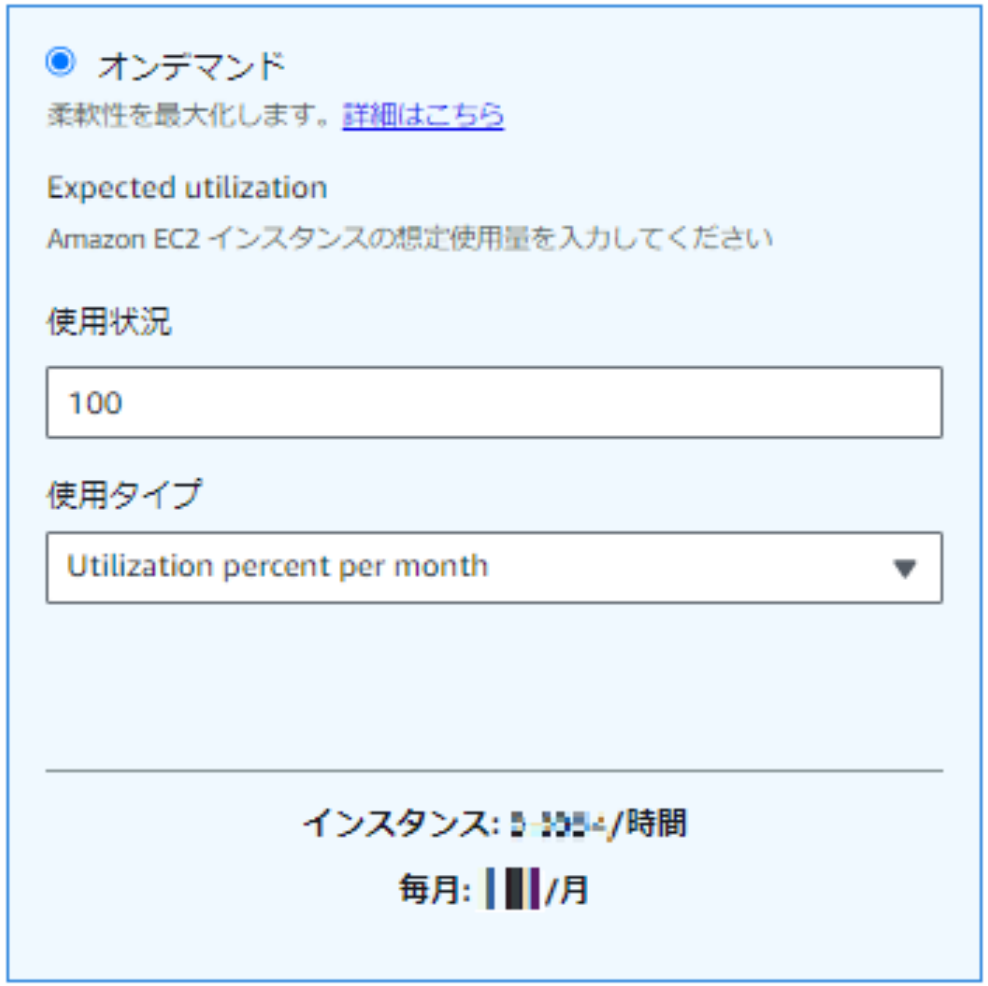
→以下の様な応答を返すことで、インスタンスごとの稼働状況を把握しましょう。

Instance ID: i-xxxxxxxxxxxxxxxxx, Running Time: 24 hours, Running Percentage: 100.0%
Instance ID: i-xxxxxxxxxxxxxxxxx, Running Time: 0 hours, Running Percentage: 0.0%

⇒上記結果を参考にAWS Pricing Calculatorで削減可否をシミュレーションし、RIの購入判断を行いましょう！

RI(Reserved Instance)の購入判断

＜参考：Pricing Calculatorでの稼働率を元にした料金計算＞



稼働率を元にオンデマンドでの料金を確認し、RI購入でコスト削減に繋がるかを確認可能！

SP(Savings Plans)の購入判断

(Fargateを例とした場合)

・Lambdaで各クラスター単位で予約されてる以下情報を取得することで
SP購入の判断基準とすることができます。

・予約されているメモリ
・予約されているvCPU数
・SP購入においては稼働率は影響が無いため、1時間ごとの使用率を確認

→以下の様な応答を返すことで、稼働状況を把握しましょう。

Time: 20xx-xx-xx xx:xx:xx+00:00, Total CPU Reserved: 4096.0, Total Memory Reserved: 8192.0

⇒上記結果を次ページでご紹介する料金計算に当てはめ、SPの購入判断を行いましょう！

SP(Savings Plans)の購入判断

＜参考：SP購入時の料金計算＞

・ 抽出した結果から1日当たり無駄にならない容量分のSPを計算して購入
  - FargateのCompute Savings Planにて該当リージョンの料金を確認
  - 1時間あたりのGB単位の料金 x 1時間あたりのvCPU単位の料金 を計算

※ AWS Pricing Calculatorで通常料金を算出することで減額量を把握することも可能

参考：ソースコードについて

今回の方法について、弊社技術ブログで詳細をご紹介しています。
ブログ内ではソースコードも載せていますので、ご興味のある方は是非ご覧ください！(以下QRコードよりご参照下さい！)

ブログタイトル：「AWSのRI、SPで購入した方がよいリソースを自力で抽出してみた」
※URL： https://www.qes.co.jp/media/aws/a363

2. 共通リソースから考えるコスト管理

はじめに - 共通リソースとは

・これからご説明する「共通リソース」は、
部門間/システム間で共用で使用するリソース
(DirectConnect,TransitGateway等のNWリソース、
セキュリティサービス系リソース、AWSサポート等)
を指します。

AWS環境を共通で使用することの課題

・複数部門/複数システムで共通のAWS環境を使用する場合、
一般的にコスト配分タグによりコスト配分をしているかと思います。
- Organizations配下のマルチアカウント
- 同アカウントで複数システム稼働 ・・・等

⇒以下についてはコスト配分タグでカバーしきれない範囲となり、
コスト管理が課題となります。
(1) AWS共通リソースの費用の按分
- DirectConnect、TransitGateway等NW要素に関する費用
- セキュリティガバナンスに対する費用
- AWSサポート費用
(2) 付与クレジットの適切な計算

AWS共通リソースの費用の按分

・ AWS共通リソースの費用の按分
  - DirectConnect、TransitGateway等NW要素に関する費用
  - セキュリティガバナンスに対する費用
  - AWSサポート費用

→ AWSの共通リソースとなり、明確なコスト配分が難しい部分となります。
各部門/利用者が正しくコスト管理/最適化を行うためには正しい費用感の把握が必要です。

⇒ コスト管理の第一歩として、各部門のAWS利用料に応じて割合を算出し、按分することをおすすめします。

付与クレジットの適切な計算

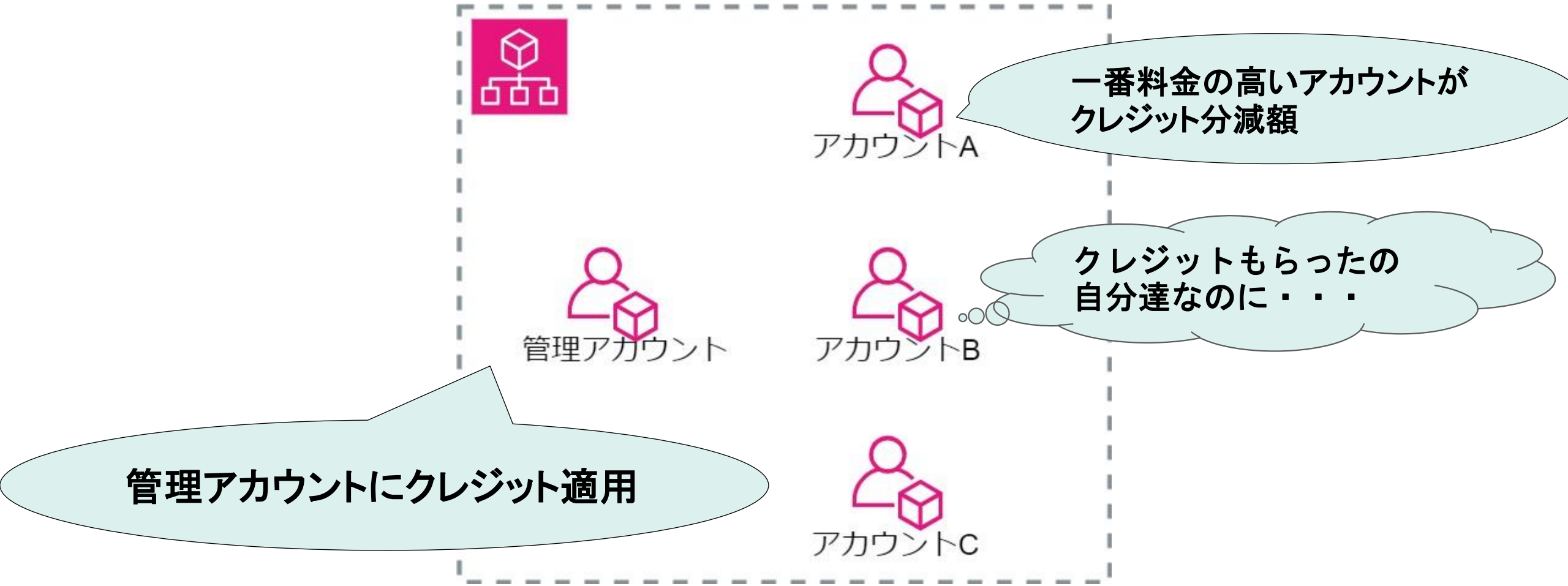
- AWSでは以下の様な場合にクレジットが発行されます。
  - POC申請を行った時
  - AWS側が起因の障害でSLAを下回った時
  - クーポン配布時(アンケート回答時等)
  - AWS Partner Network関連クレジット(APN所属企業のみ)

- 発行されたクレジットを適用した時、以下の優先順位で適用されます。
  1. クレジットを所有するアカウント
     (一部クレジットは管理アカウントに自動適用されるものあり)
  2. Organizations内で支出が最も高いアカウント
  3. そのアカウント内で最も高い費用を持つサービス
  4. そのサービス内で最も高い費用を持つ在庫保持単位(SKU)

  ※該当するクレジットを使い果たすまで、上記プロセスを繰り返します。

⇒こちらの仕様を考慮して料金計算を行った方が適切にコスト管理が行える場合があります。

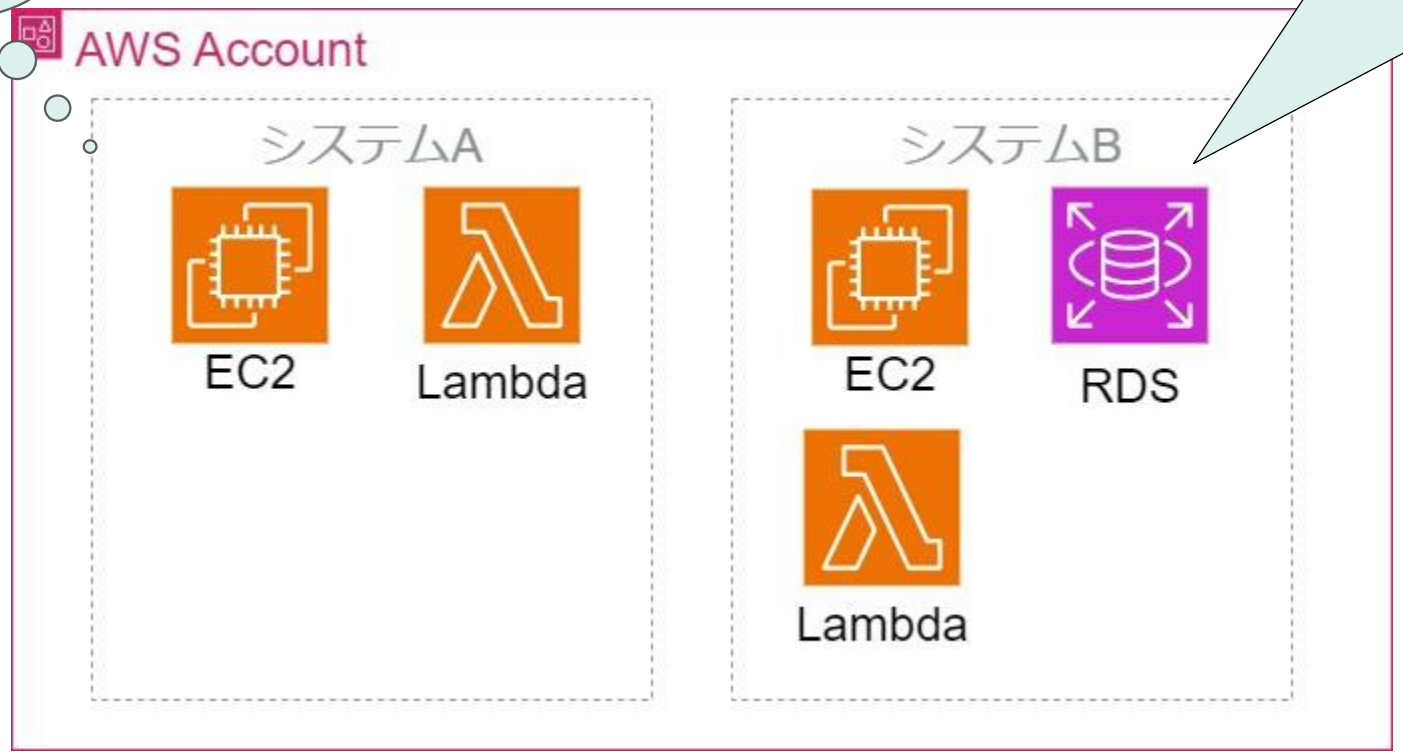
クレジット適用イメージ



クレジット適用イメージ

クレジットもらったの
自分達なのに・・・

一番料金の高いRDSが
クレジット分減額



クレジットの考え方

・適切なコスト管理を行うためには、このようなクレジットの仕様を理解の上で
以下のようにクレジット分を考慮した再計算を行うことをおすすめします。

- アカウント毎/システム毎のAWS利用料の割合に応じて、
クレジット分を按分して再計算する。（共通リソースと同じ考え方）

- クレジット減額分は代表部門(AWS基盤管理部門等)に割り当て、
その他部門についてはクレジット適用無しとして再計算する。

3.開発エンジニア目線で考えるコスト管理/削減

開発エンジニア目線でのコスト管理の課題

・「コスト削減して」と言われた時、RI/SPの購入で満足してしまいがち。
→ あくまで一例ですが、RI/SP以外にも以下の様な削減方法が考えられます！
- リージョン変更
- 使用プロセッサの変更
- スポットインスタンスの活用
- ログ出力方式の検討

リージョン変更

・構築を行う際、とりあえず東京リージョンとしていませんか？
リージョンの変更によりコスト削減の余地があるかもしれません。
次ページのチェックリストも活用して、リージョン変更が選択肢に入るか含め検討してみましょう！

＜参考：EC2インスタンスを例とした場合のリージョンごとの料金＞

リージョン	インスタンスタイプ	時間単価
東京	m7i.xlarge	$0.2604
オハイオ	m7i.xlarge	$0.2016
	差額：	$0.0588 (約22%削減)

※ Linux/オンデマンドインスタンスの場合(2024年2月時点)
※ 月額あたり約￥6,350の減額(24h x 30d x 150) ($1=¥150計算)

リージョン変更

・以下チェックリストを元に、リージョンの変更が可能かを確認してみましょう。
変更が可能であればコスト削減に繋がるかもしれません！
また、新サービスが使用可能になるなどの副次的なメリットもあります。
(検証環境の様な閉じた環境だと特に有効な可能性あり！)

1. Direct Connect等、リージョン変更により接続不可が発生する懸念はないか？
   ⇒ Direct Connect使用の場合、リージョン変更により接続が保てなくなる可能性があります。
2. インスタンスの使用率が高いか？
   ⇒ インスタンスが常に高負荷で動作している場合、
   より費用対効果の高いリージョンに移行することでコスト削減が期待できます。
3. インスタンスがスポットインスタンスではないか？
   ⇒ スポットインスタンスは需要と供給に応じて価格が変動するため、
   リージョン移行が必ずしも効果的でない可能性があります。
4. インスタンスがRIではないか？
   ⇒ RIは契約期間中は特定のリージョンでの利用が前提となるため、リージョン変更が不可。

リージョン変更

5. データ転送費がリージョン変更により増加しないか？
   ⇒ リージョン間のデータ転送は追加費用が発生することが多いため、
   リージョン変更によるデータ転送費を考慮して全体コストを考える必要があります。
6. インスタンスのパフォーマンスにリージョン変更が影響を及ぼさないか？
   ⇒ ネットワーク遅延等によりパフォーマンスに影響が出る可能性があります。
7. ユーザーの接続速度にリージョン変更が影響を及ぼさないか?
   ⇒ リージョンが変わると接続速度が変化し、リージョンが遠くなると遅延が増加する可能性あり。
8. データのプライバシーと規制に影響を及ぼさないか？
   ⇒ データを異なるリージョンや国に移すことで、
   データ保護法や規制上の問題が発生する可能性があります。
9. データレプリケーションやバックアップ戦略に影響を及ぼさないか？
   ⇒ データが複数のリージョンにわたって保管されている場合、
   リージョンの変更がデータの管理やバックアップ戦略に影響を及ぼす可能性があります。
10. リージョン間でのデータ転送費用が増加しないか？
    ⇒ 異なるリージョン間で頻繁にデータ転送を行う場合、転送費用がかさむ可能性があります。
11. サポートまたはサービス契約に上記の変更が違反しないか？
    ⇒ サービス提供者との契約内でリージョンの変更が許可されているか確認が必要です。

使用プロセッサの変更

・プロセッサ観点でもコスト削減の余地があるかもしれません。
次ページのチェックリストも活用して、プロセッサ変更が選択肢に入るか含め検討してみましょう！

＜参考：プロセッサ種別ごとのEC2インスタンス料金＞

プロセッサ種別	インスタンスタイプ	時間単価
AMD	m7a.xlarge	$0.29946
Intel	m7i.xlarge	$0.2604
Arm(Graviton3)	m7g.xlarge	$0.2108
	最大差額:	**$0.08866 (約29%削減)**

※東京リージョン/Linux/オンデマンドインスタンスの場合(2024年2月時点)
※ 月額あたり約￥9,575の減額(24h x 30d x 150) ($1=¥150計算)

使用プロセッサの変更

・以下チェックリストを元に、プロセッサの変更が可能かを確認してみましょう。
変更が可能であればコスト削減に繋がるかもしれません！
※ Intel-AMD間の変更は比較的容易に可能ですが、
Armへの変更は考慮点も多いので注意しましょう。
※ EC2、Lambda、FargateなどコンピュートリソースでのArm採用はハードルが
高いですが、RDS、OpenSearch等のマネージドサービスでのArm採用は
比較的ハードルが低いのでArmの採用を検討しましょう。

1. 特定のプロセッサアーキテクチャに依存していないか？
⇒ 特定のアーキテクチャに依存している場合、サポートされているプロセッサを
選択することが必要です。
特に、Armプロセッサはx86プロセッサとは異なる命令セットと機能を持つ
可能性があるため、Armへの変更を検討している場合は、
検証環境等でArmに変更しても問題無く動作することを確認してから変更する
必要があることに注意が必要です。
2. 使用しているOSはWindwosか？
⇒ Windowsの場合、Armプロセッサが非対応となります。

スポットインスタンスの活用

・稼働に不安が残るという理由で、スポットインスタンスを選択肢から除外していませんか？
次ページのチェックリストも活用して、スポットインスタンスの使用が選択肢に入るか含め検討してみましょう！

＜参考：スポットインスタンスの料金＞

種別	インスタンスタイプ	時間単価
通常オンデマンドインスタンス	m7i.xlarge	$0.2604
スポットインスタンス	m7i.xlarge	$0.0721
	差額:	$0.1883 (約72%削減)

※東京リージョン/Linuxの場合(2024年2月時点)
※ 月額あたり約￥20,336の減額(24h x 30d x 150) ($1=¥150計算)

スポットインスタンスの活用

・以下チェックリストを元に、スポットインスタンスの使用が可能かを確認してみましょう。使用可能であればコスト削減に繋がるかもしれません！

1. インスタンスのダウンタイムが許容されるか？
   ⇒ スポットインスタンスは突然終了される可能性があります。
2. インスタンスの使用率が高いか？
   ⇒ スポットインスタンスはEC2の余剰リソースを利用するため、
   使用率が高いほどコストパフォーマンスが良くなります。
3. メンテナンスのスケジューリングが柔軟であるか？
   ⇒ スポットインスタンスは提供時刻や期間が不定なため、
   スケジューリングの柔軟性が求められます。
4. アプリケーションがスポットインスタンスの終了2分間の通知を適切に処理できるか？
   ⇒ スポットインスタンスは終了の2分前に通知が行われます。
5. 内部データの一部が失われても許容できるか？
   また、EBSにアプリケーションの動作に必要な一時データを保持しないか？
   ⇒ スポットインスタンスは突如終了する可能性があるため、
   重要なデータを保持しないことが必要です。

ログ出力方式の検討

・どんなログでもCloudWatchLogsに出力していませんか？
- CloudWatchLogsは意外と高額！
ログ監視を行うログだけをCloudWatchLogsに送信し、
それ以外のログについてはS3等に直接保管する様にしましょう！
加えてCloudWatchLogsも一定期間でS3に出力するようにしましょう！
※ S3でもIntelligent Tieringを活用するなど、コスト最適化を図りましょう！

- EC2のログを監視する場合は、CloudWatchAgentでフィルタ設定を行って
必要なログのみをCloudWatchLogsに送信しましょう！

- ECS Fargateは、awslogsドライバーを使用するのではなくFireLensを使用
することで、ログの投げ分けを行うことが可能です。

⇒上記のようにCloudWatchLogs上で保管するログを最小限に抑えることでコスト削減が可能となります！

(参考)CloudWatchAgent設定例

CloudWatch Agentでのログ出力フィルタ設定例は以下の通りです。(該当箇所を抜粋)

```
"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "/var/log/test.log",
          "log_group_name": "test.log",
          "log_stream_name": "test.log",
          "timezone": "Local",
          "filters": [
            {
              "type": "exclude",
              "expression": "INFO"
            }
          ]
        }
      ]
    }
  },
  "log_stream_name": "example_stream",
  "force_flush_interval": 30
}
```

(参考) FireLens設定例

FireLensでのログ投げ分けの設定例は以下の通りです。(該当箇所を抜粋)

```
[OUTPUT]
    Name s3
    Match info*
    region ap-northeast-1
    bucket sample-firelens-bucket/error-log
    total_file_size 1M
    upload_timeout 1m
    use_put_object On
    s3_key_format  /%Y%m%d_%H:%M:%S

[OUTPUT]
    Name cloudwatch
    Match critical*
    region ap-northeast-1
    log_group_name /ecs/firelens/
    log_stream_prefix test/
    auto_create_group true
    log_key log
```

まとめ

まとめ

- 請求関連情報が見れなくてもCloudWatchメトリクスからRI/SPの購入検討は可能！正しく使用状況を把握して、コスト最適化しましょう！
- AWS共通リソースやクレジットもしっかりコスト管理しましょう！
- RI/SPの使用以外でもコスト削減は可能！色々な方法を活用しましょう！

当社の取り組み（AWSソリューションのご紹介）

"Leap Beyond"

当社では本日ご紹介したご案内を活用したコスト削減サービスをはじめとして、
ご要件に応じたAWSソリューションをご提供いたします。
まずはご相談下さい！

分類	項目	内容	備考
基盤構築	**AWS 基盤構築サービス**	AWSアカウントのセキュリティや、フルマネージドサービスを活用したAWS基盤をご提供いたします。	数々の導入実績をもとに高度なAWS技術を持つエンジニアを中心とした複数名のチームで対応いたします。 (↓サイトQRコード)
運用	**AWS 運用支援サービス**	AWS基盤での運用開始後発生するシステム運用業務について、お客様のニーズに合わせ運用の支援、作業代行を時間単位で行います。	
コスト削減	**AWS コスト削減サービス**	お客様のAWS導入後の状態を調査させていただきAWSコスト最適化のアドバイスをレポート形式でご提供させていただきます。	

株式会社QUICK E-Solutions
クラウドプラットフォーム本部
AWSプラットフォームグループ(村上・舛岡)

TEL ： 070-4271-1842
MAIL： info_apg@qes.co.jp

JAWS DAYS 2024
"Leap Beyond"

アンケート

セッションの
アンケート

全体アンケート

ご視聴ありがとう
ございました！

LEAP
BEYOND