# 概要

## 情報

サーバーレスに関する情報を広く浅く集める。

サーバー（管理）レス　という認識が正しいらしい。

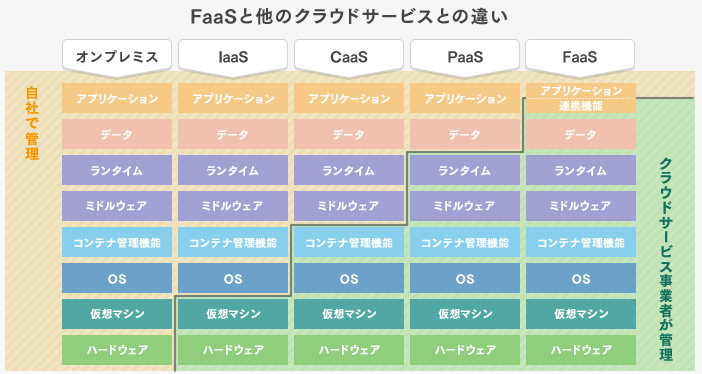
サーバーレスのデメリット

・提供サービスの仕様に縛られることになる。

・上に関連して，他サービスへの移行が難しくなる。

## クラウド製品の区分

|  |  |  |
| --- | --- | --- |
| 略語 | 説明 | 例 |
| aPaaS/hpaPaaS | PaaSの提供範囲に加えて、ローコード・ノーコードによるアプリケーション開発環境を提供。 | Kintone  Mendix  Salesforce Lightning Platform |
| BaaS/MBaaS | モバイルアプリ開発に必要な機能性を提供。  Backend≒サーバー　らしい | Amazon Cognito, Amplify  Firebase，Kii |
| CaaS  Containers-as-a-Service | コンテナオーケストレーションツールを構築した状態で提供。利用者はコンテナ上のソフトウェア構築・開発に専念できる。 | GKE，ECS，PKS，RedHat Open Shift |
| DaaS | PCのデスクトップ環境を提供。 | Amazon Workspaces  Citrix WorkSpace，VMware Horizon |
| FaaS  Function as a Service | アプリケーションの実行・管理環境を提供。利用者はアプリケーションの開発に専念できる。 | AWS Lamda  Cloud Functions  Azure Iaas |
| IaaS  Infrastructure as a Service | サーバー本体、ハウジングなどの物理インフラを構築した状態で提供。利用者はOS以上の階層のソフトウェア構築・開発に専念できる。 | EC2，GCE |
| iDaaS | IDパスワード管理、ID認証機能、シングルサインオン、アクセス制御機能などを提供。 | AWS Single Sign-On  Cloud Identity  Hennge One |
| iPaaS | オンプレとSaaS、または複数のSaaS間のシステム連携機能を提供。 | Anyflow，MuleSoft  Zapier |
| PaaS  Platform-as-a-Service | IaaSの提供範囲に加えてOSとミドルウェアまでを構築した状態で提供。  利用者はアプリケーション開発に専念できる。 | GAE  Heroku  Pivotal Application Service |
| SaaS  Software-as-a-Service | Webブラウザやネイティブアプリを通してシン・クライアントで利用するソフトウェア。 | Gmail  SAP  Salesforce |



# クラウドコンピューティング(PaaS)

## OpenStack

### 概要

AWS/AzureのようなIaaS（イアース）を構築するLinuxパッケージ。

機能

|  |  |
| --- | --- |
| Compute（Nova） | 仮想マシンを自動的に生成しサービスを提供 |
| Object Storage（Swift） | サーバーのデータ保存領域を提供 |
| Image Service（Glance） | Novaで生成する仮想マシンのVMディスクイメージを管理 |
| Dashboard（Horizon） | Web管理するためのインターフェース |
| Identity（Keystone） | ユーザー認証を管理 |
| Virtual Network Service（Quantum） | 仮想ネットワークを管理 |

flat

## AWS； Amazon Web Services

### 概要

対応言語

Java，C#，PHP，Python，Ruby，Go，C ++

### AWS共通

#### AWSリソースを操作する手段

|  |  |  |
| --- | --- | --- |
|  |  |  |
| AWSマネジメントコンソールManagement Console | Webブラウザ経由のGUI | Webブラウザ経由 |
| AWS CLI | コマンドラインツール | 開発機にインストールして実行 |
| AWS SDK | 開発キット |  |

・AWSアカウントによるAWSリソースの操作を行う運用は推奨されない。

⇒ IAMユーザーで行う

⇒ AWSアカウントでは契約関連のみ行うと考えた方が良い

#### アカウント

##### 概要

|  |  |  |  |
| --- | --- | --- | --- |
|  | | 概要 | 備考 |
| AWSアカウント | | 契約関連を行う  課金の対象者  サービス全体を管理 | ・非常に強力なので**取り扱いには最新の注意を払う**  ・別名ルートユーザー |
| IAM | ユーザー | AWSアカウントへのアクセスの認証単位 | ・アプリ層のユーザー  ・MFAの必須化ができる（推奨） |
| IAM | グループ | IAMユーザーをグループ単位で管理する機能 | ・命名規則はないが、英単語の複数形か \*-groupとされる事が多い  例：Admins、admin-groupなど |
| IAM | ロール | EC2やLambdaへ割り当てて操作権限の認可を行う |  |
| IAM | ポリシー | 操作権限の認可を行う仕組み | ・制御制限をまとめたもので、細かい制御ができるようになる |

大きく以下の４つの概念に分かれる

　ポリシー IAM全体の権限の大本になる要素

　ユーザー AWS内のサービスを操作するユーザー

　グループ ユーザーを束ねる

　ロール サービス側での操作権限

##### AWSアカウントとユーザーの違い

|  |  |  |
| --- | --- | --- |
|  | AWSアカウント | ユーザー |
| ログインの際のID | Eメール | 12桁のアカウントID /  アカウントエイリアス |
| パスワードポリシー | 変更できない |  |
| 備考 |  | アクセスポータルのURLからしかログインできない？  URLはIAM > ユーザーを選択 > セキュリティ認証情報　で確認 |

(AIM)ユーザーのアクセスポータルのURLはIAM Identity Centerの「アイデンティティソース」にある

AWSアカウント

AWSのサービスを利用するために必要なルートエンティティ

ルートユーザー

IAM→MFA（多要素認証）を追加→MFAを割り当てる

認証アプリケーション Authenticatorなどのアプリの事

　２つめのコード　は時間がたつと変化する

セキュリティキー USBメモリを用いて

ハードウェアトークン ？

ユーザー

ユーザーの追加

左メニュー：ユーザー → ユーザーを追加

ユーザーに許可セットを割り当てる

左メニュー：AWSアカウント → 管理アカウントのユーザーにチェック →  → ユーザー／グループのタブを選択 → ユーザーを選択（チェックボックス） → 次へ → 許可セット → 送信

アカウントエイリアス

12桁のIDの代わりに用いる文字列。割り当ては任意。

アカウントエイリアスの作成手順

ログイン

##### 重要事項

・アクセスキーの発行は極力避ける

・IAMユーザーは基本的に利用ユーザーごとに作る

共有ユーザーは厳禁。やるならグループで実現させる。

・リソースへのアクセス権限の付与はグループで行う

#### IAM

##### 概要

Identity and Access Management

「どのユーザーが」「どのリソースに」「どのような操作ができるか」を定義する認証／認可の仕組み。

ユーザーだけでなくアプリケーションに割り当てられる事もある

IAMロールは「アクセス元」に割り当てる。（例：Lambdaに割り当てる）

12桁のアカウントIDが割り当てられ、IAMではこれを使用する。（2023-0724多分）

許可ポリシーは

明示的なDeny > 明示的なAllow > 暗黙的なDeny（デフォルト）　となる

|  |  |
| --- | --- |
| コンポーネント | 説明 |
| IAMユーザー | AWSアカウントへのアクセスの認証単位 |
| IAMグループ | IAMユーザーをグループ単位で管理する機能 |
| IAMロール | AWSリソースへ操作権限の認可を行う仕組み |
| IAMポリシー | 操作権限の認可を行う仕組み |

##### チュートリアル

###### カスタマー管理ポリシーの作成

　IAM > ナビゲーションペイン：ポリシー > ポリシーを作成 >  でJSONを選択。

入力内容

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Deny",

"Action": "\*",

"Condition" : {

"NotIpAddress" : {

"aws:SourceIp" : [

"8.8.8.8/32"

]

}

},

"Resource": "\*"

}

]

}

ポリシー名「IpAddressRestriction」で作成

###### グループの作成

　操作：IAM > ナビゲーションペイン：ユーザーグループ > グループを作成

　名前を「ReadOnlyGroup」とする。

　ポリシーにReadOnlyAccess、また先ほど作成したIpAddressRestrictionを追加

###### ユーザーの作成

Identity Center でユーザーを指定する （推奨）

Identity Center を使用して、ユーザーにコンソールアクセスを提供することをお勧めします。Identity Center では、AWS アカウントおよびクラウドアプリケーションへのユーザーアクセスを一元管理できます。

IAM ユーザーを作成

[ 調査 ] アクセスキー、または AWS CodeCommit や Amazon Keyspaces のサービス固有の認証情報を使用してプログラムによるアクセスを作成する場合は、この IAM ユーザーの作成後に生成できます。

<https://docs.aws.amazon.com/console/iam/access-type>

###### アクセスキーの作成

　操作：IAM > ナビゲーションペイン：ユーザー > 任意のユーザーを選択

　> セキュリティ認証情報 > アクセスキー：アクセスキーを作成

メモ

・自分のパスワードだけ変えられるポリシーを作成してみる

##### 認証

別名：Authentication、ログイン

###### 概要

認証（ログイン）の方法は以下の２つ

|  |  |
| --- | --- |
| 方法 | 概要 |
| ・IDとパスワードの組み合わせ | ・MFAをオプションとしてつける事ができる |
| ・アクセスキーIDとシークレットアクセスキーの組み合わせ | ・できるだけ使用を避ける  理由：鍵ファイルの管理が難しい為  ・ |

補足：アクセスキーはコマンドやAPIがAWSリソースにアクセスする際に使用

###### アクセスキー

ユースケース

・コマンドラインインターフェイス

・ローカルコード

・AWSコンピューティングサービスで実行されるアプリケーション

・サードパーティーサービス

・AWSの外部で実行されるアプリケーション

・その他

アクセスキーの発行はできるだけ避ける

代替案１

ブラウザベースの CLI である AWS CloudShell を使用

> でも、重たいんでしょう？

代替案２

AWS CLI V2 を使用して

https://docs.aws.amazon.com/ja\_jp/cli/latest/userguide/cli-configure-sso.html

##### ロール

###### 概要

EC2やLambdaなどの各種AWSリソースに対して、他のAWSリソースの操作権限を管理する仕組み。

例：EC2インスタンスにS3バケットへアクセス許可するIAMロールをアタッチ

・うまく使うとIAMユーザーのアクセスキー運用がほぼ不要になる

・ログイン自体はIAMユーザーで行い、ログイン後に権限をIAMロールに切り替える、といった使い方をする。

補足：これをクロスアカウントロールという（らしい）

⇒ 操作：このロールを使用できるアカウントを指定する

###### 操作方法

IAM > ナビゲーションペイン： ロール > ロールを作成 >

信頼されたエンティティタイプで「AWSのサービス」を選択

　ユースケースで　他のAWSのサービスのユースケース　のドロップダウンで：Amplify　を選択

　ラジオボタンが表示されるので　Amplify – Backend Deploymentを選択　「次へ」

　「許可を追加」画面で許可ポリシー AdministratorAccess-Amplify が選択されていることを確認して「次へ」

　ロール名に「Amplify-Dev」（ロール名何でも良い）

　説明はデフォルトのままに（英数字しか使えない為）

　「ロールを作成」で作成される

###### 認証関連

MFA強制ポリシー

<https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/tutorial_users-self-manage-mfa-and-creds.html>

ハードキー：Yubikey

<https://www.yubico.com/yubikey/?lang=ja>

###### 信頼されたエンティティタイプ

該当のロールに切り替える事ができるアカウントの事。

例えばログイン自体はIAMユーザーで行い、ログイン後に権限をIAMロールに切り替える事ができる。

（クロスアカウントロール）

|  |  |
| --- | --- |
| AWSのサービス | ユーザーに代わって AWS サービスがアクションを実行する |
| AWSアカウント | このアカウントでアクションを実行することを他の AWS アカウントのプリンシパルに許可する　？？？ |
| ウェブアイデンティティ | 指定した外部ウェブアイデンティティプロバイダーによってフェデレーションされたユーザー |
| SAML2.0 フェデレーション | SAML 2.0 でフェデレーションされたユーザー |
| カスタム信頼ポリシー |  |

##### ポリシー関係

###### 操作方法

アイデンティティー（ユーザー）ベースのポリシー

ユーザーまたはOU；Organization Unit（グループ）に対してアタッチする。

|  |  |  |
| --- | --- | --- |
| 種類 | 概要 | 基本的な使いかた |
| AWS管理ポリシー  AWS Managed Policies | AWSによってあらかじめ定義されたポリシー | 基本的な権限を付与 |
| カスタマー管理ポリシー  Customer Managed Policies | ユーザーが定義するポリシー。  世代管理が可能 | 不要な権限をはく奪 |

|  |  |
| --- | --- |
| 種類 | 概要 |
| 管理ポリシー | 1つのポリシーを複数の対象にアタッチできる |
| インラインポリシー | **特定のグループ、ユーザーに作成・付与**する。複数の対象へアタッチできない。  （管理ポリシーができる前の古い仕組みなので使用を避ける） |

IAM > ユーザー からアタッチしたいユーザー名を押下して、既存のポリシーを直接アタッチで選択するポリシーがアイデンティティベースのポリシー。

###### 信頼ポリシー

IAMロールがどのAWSサービス・リソースへアタッチできるかを指定

###### IAMポリシー

自身のAWSアカウントに対し、IAMユーザー単位でアクセス権限を設定。

・IAMユーザーやIAMグループなどにアタッチする事ではじめて利用できる

比較

|  |  |  |  |
| --- | --- | --- | --- |
|  | 適用範囲 | 対象 | 備考 |
| セキュリティーグループ | リソース | IPアドレスの範囲  セキュリティグループ | ・厳しく、細かく設定  ・通信の状態を管理する（ステートフル） |
| ネットワークACL | サブネット | IPアドレスの範囲 | ・緩めに設定（全開放でも良いくらい）  ・通信の状態を管理しない（ステートレス） |

AWS管理ポリシー

カスタマー管理ポリシー

インラインポリシー （管理ポリシーができる前の古い仕組みなので使用を避ける）

###### リソースベースのポリシー

S3バケットなど、AWSリソースや認証ポリシーに対してアタッチする

補足：IAM ロールもリソースなので、信頼ポリシーというポリシーが用意されている

ECRを例にすると

ECR > リポジトリ からアタッチしたいリポジトリ名を押下して、ナビゲーションペイン(左側のメニュー)でPermissionsを選択するとアクセス許可画面が開きます。右上のポリシーJSONの編集で編集出来るのがリソースベースのポリシーです。

###### IAM Policy Simulator

###### 資料（ポリシー）

https://docs.aws.amazon.com/ja\_jp/IAM/latest/UserGuide/reference\_policies.html

##### JSON形式でのポリシー記述例

###### 基本

|  |  |  |  |
| --- | --- | --- | --- |
| JSON | | 概要 | 具体例／備考 |
| Version | |  | 2008-10-17(def.)と2012-10-17 |
| Statement | Sid | ドキュメント内で一意の識別子 | 必須項目ではないらしい  "Statement1" |
| Statement | Principal |  |  |
| Statement | NotPrincipal |  |  |
| Statement | Action | 対象のサービス（操作） | "s3:\*" |
| Statement | NotAction | 制限するサービス（操作） |  |
| Statement | Resource | 機能（AWSサービス） | arn形式で記述する |
| Statement | NotResource |  |  |
| Statement | Effect | 許可／拒否 | Allow／Deny拒否 |

最もシンプルな例（すべての権限を持ったIAMポリシー）

{

"Version" : "",

"Statement" : [

{

"Effect" : "Allow",

"Action" : "\*",

"Resource" : "\*",

}

]

}

arn; Amazon Resource Name形式

arn:aws:<service>:<region>:<account>:<resource>

補足：あさりあれ　で覚える事にする。（Aws, SERvice, REgion, Account, REsource）2023-10-26

公式：<https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/reference_policies_elements.html>

###### 評価ロジック

公式：<https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/reference_policies_evaluation-logic.html>

・明示的なDeny > 明示的なAllow > 暗黙的なDeny（デフォルト）　となる

・Statementが複数ある場合にはOR（論理和）が適用される

⇒ どれか一つでもDenyに該当する場合はDenyとなる

公式：<https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/reference_policies_elements_statement.html>

・Condition ブロックでは、演算子やキーについては AND で、キーに対する Value は OR で評価

公式：<https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/reference_policies_condition-logic-multiple-context-keys-or-values.html>

###### Statement - Condition　条件

|  |  |  |  |
| --- | --- | --- | --- |
| JSON | | 概要 | 備考／具体例 |
| IpAddress | | アクセスを許可するIPアドレス（郡） | 送信元のパブリックIPアドレス（プライベートIPを指定しても意味が無い） |
| NotIpAddress | | アクセスを拒否するIPアドレス（郡） |  |
| Bool | MultiFactorAuthPresent | MFAを義務付ける |  |
|  |  |  |  |
|  |  |  |  |
| StringLike | | 文字列に指定文字が含まれる |  |
|  | |  |  |

"Bool": {"aws:MultiFactorAuthPresent": true }

"Condition": {"StringLike": {"s3:prefix": ["marketing/\*"]}}

SourceIp 抜粋

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Deny",

"Action": "\*",

"Condition" : {

"NotIpAddress" : {

"aws:SourceIp" : [

"8.8.8.8/32"

]

}

},

"Resource": "\*"

}

]

}

参考：<https://qiita.com/Shoma0210/items/17193f254180396fb8e1>

テンプレート

例

{

"VerCondition" : "",

"Statement" : [

{

"Effect" : "Allow",

"Action" : "\*",

"Resource" : "\*",

}

]

}

###### Statement - Principle

信頼するAWSのサービス。arn形式で指定する。

###### 資料

サービス

iam IAM

logs

##### Permissions Boundary

別名：許可の境界、アクセス許可境界

IAMユーザー（アイデンティティベースのポリシー）に対して個別に設定するアクセス境界。

IAMユーザーやIAMロールに対して設定できる操作可能な範囲の上限 from pingt　⇒上限？？

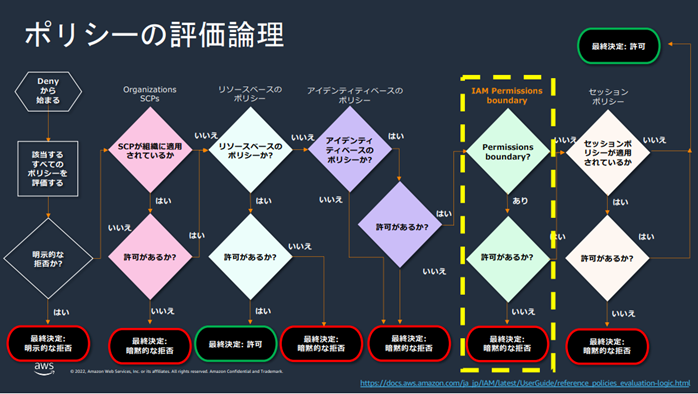
IAMポリシーとPermissions Boundary両方で許可されたアクションのみ許可される

注意：awsに慣れないうちは意識しなくてよい

・リソースベースのポリシーは関係がない（らしい）

・組織外の人間に権限を委任する場合などに使う

公式：<https://docs.aws.amazon.com/ja_jp/singlesignon/latest/APIReference/API_PermissionsBoundary.html>



[https://docs.aws.amazon.com/ja\_jp/IAM/latest/UserGuide/reference\_policies\_evaluation-logic.html#policy-eval-denyallow](https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/reference_policies_evaluation-logic.html" \l "policy-eval-denyallow)

##### 具体例

・グループのデザインパターン

運用として複数グループに登録する方法と、グループ自体に複数ポリシーを付与する２つのパターンがある。

基本は１ユーザーに対して１グループがわかりやすくて理想

###### ホワイトリストパターン

許可する権限のみ付与していくパターン

ユースケース

役割分担がはっきりとしているウォーターフォール式開発の場合

デメリット

実験的な案件で、運用パターンが決まっていないとき都度修正が必要

###### ブラックリストパターン

許可する権限のみ付与していくパターン

ユースケース

役割分担がはっきりとしているウォーターフォール式開発の場合

デメリット

AWSが新機能を搭載後に対応しないとそれが使えるままになってしまう

###### ハイブリッドパターン

AWSが新機能を搭載するペースが速く、AWSの定義済みポリシー＋自作のブラックリストを組み合わせた

ハイブリットパターンが現実的、との事。

###### IAMグループのデザインパターン

まずポリシーのパターン

|  |  |  |
| --- | --- | --- |
|  | １つのポリシー方式 | 複数ポリシー方式 |
| 概要 | １つのポリシーにすべて書く | 複数のポリシーをIAMグループに適用 |
| 長所 | 可視性が良い | 再利用性が高い |
| 短所 |  |  |
| 組織 | 少人数・小規模 | 大規模利用 |
|  |  |  |

IAMグループのパターン

|  |  |  |
| --- | --- | --- |
|  |  |  |
| IAMユーザー | 複数のグループに所属 | １つのグループに所属 |
| グループとポリシー | １，２このポリシーを割り当てる | グループごとに複数のポリシー |
| 概要 | ・必須の権限を全ユーザー向きグループに適用  ・ |  |
| 長所 | ・必須権限の漏れが発生しにくい  ・変更の際の適用範囲がわかりやすい | ・ユーザーから見るとポリシーの構造がシンプル |
| 短所 |  |  |

##### その他

インスタンスプロファイル

IAMロールに対する入れ物（参照先情報）。

例えばEC2インスタンスが起動時に設定されたIAMロール（現在のログインユーザーの情報）を読み込む為に使われる

ユースケース

EC2で動作するアプリから、S3バケット「mybucket」に格納されている機密ファイルを参照したい

⇒EC2インスタンスのインスタンスプロファイルに、「mybucket」への参照権のみを付与したIAMロールを関連づける

命名規則

+=,.@-\_ 。これを考慮して命名する。

環境情報の含有: 例えば「dev」と「prod」のように環境に関連する情報を含めることがある。

機能や役割: 「Admins」、「ReadOnly」、「S3FullAccess」のように、そのグループのメンバーが持つ権限や役割を示す名前にすることが多い。

例:

Prod-Admins: 本番環境の管理者グループ

Dev-ReadOnly: 開発環境の読み取り専用グループ

S3-DataTeam: S3に関連する権限を持つデータチームのグループ

#### Auto Scaling

##### 概要

EC2インスタンスやRDSなどのサービスにおいてリソースが枯渇した時に自動的にスケーリングする機能。

ELB；Elastic Load Balancingと組み合わせて負荷分散を実現する事が多い。

主な設定項目

|  |  |
| --- | --- |
| 項目 | 説明 |
| 希望する容量 | Auto Scalingグループで稼働するリソースの数 |
| 最小キャパシティ | Auto Scalingグループで稼働するリソースの最小数 |
| 最大キャパシティ | Auto Scalingグループで稼働するリソースの最大数 |
| ターゲットグループ | Auto Scalingグループに紐づけるELBのターゲットグループ |
| ヘルスチェックのタイプ | ELBのヘルスチェックと、 EC2のステータスチェックが利用できる |
| ヘルスチェックの猶予期間 | インスタンスが起動してから初回のヘルスチェックまでの待機時間 |
| インスタンスの保護 | スケールインが発生した時に削除対象から除外する |
| 終了ポリシー | スケールインが発生した時に、 リソースを削除する優先順位 |

複数のAZ（AZ-a、AZ-b、AZ-c）にわたるAuto Scalingグループで実行されており、常時6台のインスタンスの稼働が求められている場合の最もコスト効率が高いインスタンス配置は

AZ-a内にインスタンスを3台、AZ-b内にインスタンスを3台、AZ-c内にインスタンスを3台

⇒バックアップインスタンスは起動していない為費用が掛からない、という事？2023-09-13

補足

Auto Scalingグループ：自動スケーリング対象の管理単位

具体例

・ヘルスチェックに異常が発生した場合は新規にインスタンスを起動させる

・最も低いコストで最低限の可用性を確保する

⇒ インスタンスを2つ用意し、インスタンスの最小キャパシティ2、最大キャパシティ2で運用する

##### Auto Scalingグループ

Auto Scalingの管理単位で、１つのグループにはEC2インスタンスの最大数、最小数などが設定される。

ヘルスチェックで正常なリソース数がリソースの最小数より下回った場合、Auto Scalingの「起動テンプレート」で設定した新しいリソースが起動されます。

##### ヘルスチェック

Auto Scalingのヘルスチェックのタイプ

EC2インスタンスの場合 「EC2」と「ELB」

|  |  |
| --- | --- |
|  | 概要 |
| EC2 | インスタンスの「ステータスチェック」の結果を確認  無効にできない設定項目 |
| ELB | 指定した接続先への応答確認  ping、httpレスポンス、TCPハンドシェイク　の順に行う（らしい） |

・Auto Scalingのヘルスチェックで異常となったリソースは、自動的に終了

- タイプがELB無効でEC2のみの場合、EC2インスタンスが404エラー等を出してもステータスチェックの結果が良好であれば、Auto Scalingによる終了の対象外となる

⇒ 終了しなければ新規起動もできない為、最小キャパシティ未満のインスタンスで負荷分散する可能性が出る

・デフォルトでは「EC2」が有効、「ELB」は無効

（「EC2」「ELB」ともに有効にすることが推奨）

##### スケーリングポリシー

スケーリングを任意に発生させる場合の条件設定

（スケーリングは基本的にはリソースの負荷状況やスケジュールに従って自動的に実施される）

###### 動的スケーリング

CPUやネットワークなどのパフォーマンスの負荷状況に応じて、自動的にスケールアウト/インを実施。

動的スケーリングを利用するには、スケーリングの発生条件となるメトリクスと、スケールアウト/インのアクションを「動的スケーリングポリシー」に設定します。

タイプ

|  |  |
| --- | --- |
|  | 概要 |
| シンプルスケーリング | 特定のメトリクスに対する1つのしきい値に基づいて、インスタンスを増減 |
| ステップスケーリング | シンプルスケーリングのような増減を段階的に行う |
| ターゲット追跡スケーリング | 特定のメトリクスが指定した目標値になるように、インスタンスを増減 |

具体例

シンプルスケーリング

300秒間の平均CPU使用率が50％を超えたら、インスタンスを1台追加する

ステップスケーリング

300秒間の平均CPU使用率が50％を超えたらインスタンスを1台追加し、90％を超えたら2台追加する

ターゲット追跡スケーリング

平均CPU使用率を30％で維持

###### スケジュールに基づくスケーリング

特定の日時をトリガにして増減を行う

具体例

・毎日10時にインスタンス台数を4台にする

###### 手動スケーリング

文字通り、手動で増減を行う。バッチ処理などの高負荷処理の前に増やしたりする事ができる。

###### 予測スケーリングポリシー

今までのデータを元に増減スケジュールを自動で作成する。

・2021年5月に開始

・予測のみを行い、スケーリングを行わないモードが存在する

##### ライフサイクルフック

スケーリング発生によるEC2インスタンスの起動または終了時に任意の処理を実行する機能

例

・スケールアウトによって新たに追加したインスタンスに初期化スクリプトを実行

・スケールインによって終了するインスタンスのログを収集・退避させるといった処理を行う

##### 終了ポリシー

Auto Scalingグループでスケールインが発生した時、リソース数が最も多いAZ内のリソースが削除されるが、

具体的にどのリソースを優先的に削除するのか決める。

選択肢は以下：

・最も起動時刻が古いリソース

・最も起動時刻が新しいリソース

・最も作成時刻が古い起動テンプレートを利用して起動されたリソース(def.)

・次の課金タイミングが最も早いリソース

補足

起動テンプレート

スケールアウトが発生した時に起動するリソースを設定する。

#### Cognito

AWSの認証システム

ASWを利用して構築したシステム／サービスの認証(Authentication)と認可(Authorization)を担当

#### Identity Center

AWS Organizationsで複数アカウントを運用している環境で各ユーザを集約管理し、各アカウントへのログインを簡単に行えるようにするためのサービス

（2023年3月頃にスタート？）

・前提として、AWS Organizationsを使用していることが条件

・シングルサインオンの仕組みを簡単に提供する（らしい）

#### Organizations

##### 概要

**複数のAWSアカウント**をまとめて管理する機能

組織内のアカウントは「管理アカウント（マネジメントアカウント）」と「メンバーアカウント」に分けられる

対象

・個々のAWSアカウント

・OU（Organization Unit）

例：

・複数のAWSアカウントに対して権限を設定

・複数のAWSアカウントの請求を一括処理

##### SCP；Service Control Policy

別名：サービスコントロールポリシー

###### 概要

AWS Organizationsの機能で、複数のAWSアカウント、組織（OU）に対してざっくりとした権限制御を一元的に行うための仕組み

ユースケース

・大量のアカウントに一括で設定を行う

例：

・テスト用のアカウント（群）は特定のサービスにしかアクセスできないようする

・全てのアカウントに対してMFA（多要素認証）を強制するように設定する

#### ルーティング関連

##### 概要

ほとんどはVPC dashboardで設定される

##### リージョン

・グローバル

「全世界」を示す区分

（普段意識する必要は無いとおもわれる）

・リージョン

地域

データセンターが配置されている場所

参考<https://aws.amazon.com/jp/about-aws/global-infrastructure/>

##### AZ；Availability Zone

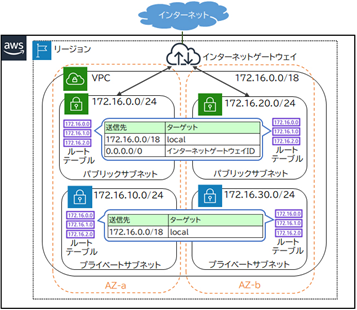
リージョン内の物理的および論理的に区切られたデータセンター（群）

１つのリージョン内に複数のアベイラビリティゾーン（AZ）が存在する事になる。

##### エッジロケーション

AZとは異なるAWSデータセンターで、AZよりも世界中に数多く配置されている

##### サブネット

VPCを論理的に分割したネットワーク空間。サブネットは一つのAZに属し、一つのAZ内に複数のサブネットが作成される。

・複数のVPCをまたいで作成することはできない

・複数のリージョンをまたいで作成することはできない

パブリックサブネット

インターネットから直接通信があるサブネット

プライベートサブネット

インターネットから直接通信がないサブネット

操作

EC2の場合

指定方法

インスタンス作成のときに選択。（それ以降は変更できない）

確認方法

EC2ダッシュボード > インスタンス > インスタンス選択 > ネットワーキング > サブネットID

##### IPアドレスの範囲（CIDR）

・VPCではCIDRブロックのプレフィックス長を**/16から/28の間で**指定

・サブネットのCIDRは、VPCのCIDRブロックの範囲内で指定する必要がある

言い換えると　サブネットのCIDR bit値 < VPCのCIDR bit値

補足：VPCの中にサブネット（CIDRブロック）が存在する

補足：CIDRブロック　≒１つのVPC（セグメント）の事？ 2023-10-16

・AWSは次のプライベートIPv4アドレス範囲から、CIDRブロックを指定することを推奨。

10.0.0.0 ～ 10.255.255.255、172.16.0.0 ～ 172.31.255.255、192.168.0.0 ～ 192.168.255.255

（通常のプライベートIPv4の範囲と同じ）

##### ルートテーブル

VPC内からVPC外へのデータパケットを**どこへ転送するか**を定義する機能。

（転送先に応じて次のノードを指示する）

・「local」は、送信元と同一VPC内でルーティングするという意味

・送信先「0.0.0.0/0」は「デフォルトルート」（デフォルト設定）

・igw-xxxxxxは、インターネットゲートウェイに付けられるID

・サブネット（パブリック／プライベート）１つにルートテーブルを１つ（だけ）紐付け

・ルートテーブルの紐付けがないサブネットは、メインルートテーブルに従ってルーティングされる

###### メインルートテーブル

VPC全体に適用されるルートテーブル

公式

ルートテーブルを設定する：<https://docs.aws.amazon.com/ja_jp/vpc/latest/userguide/VPC_Route_Tables.html>

##### ゲートウェイ

###### 概要

###### インターネットゲートウェイ

・複数のパブリックサブネットで共有して利用する

・これ自体にネットワークアクセス制御の機能は無い

###### Egress-Only

NATゲートウェイとインターネットゲートウェイの特徴を併せ持つIPv6専用の機能

・VPCからインターネットへ（Egress）の接続開始要求は通す

・インターネットからVPCへの接続開始要求は通さない

・IPv6専用のインターネットゲートウェイ。

###### NATゲートウェイ／インスタンス比較

**プライベートサブネットからインターネットへ**の通信を可能にする為の機能

・いずれも**パブリックサブネット**に作成する

|  |  |  |
| --- | --- | --- |
|  | NATゲートウェイ | NATインスタンス |
| 保守 | マネージドサービス | ユーザーが障害対応などの運用管理を実施する必要がある |
| ポート転送機能 | なし | あり |
| 作成方法 |  | EC2インスタンスから作成 |
| IPのバージョン | IPv6/IPv4 | IPv4 |

###### NATゲートウェイ

VNC内のプライベートIP ⇔ EIP；Elastic IPの変換を行う

プライベートサブネット > インターネット（逆方向の通信は拒否）

冗長性

- 単一ゾーン内で冗長性を持たせて実装される仕様である為、トラフィック増加程度であれば対応が可能

- 複数AZに作成することでAZに障害が発生した場合の耐障害性を高める事ができる

###### NATインスタンス

NATゲートウェイに似ているが多少の違いがある。

・VPC外からプライベートサブネット内へ接続する際の踏み台サーバーとして利用できる

##### セキュリティサービス

###### 全体

概要

VPC内へのネットワークアクセスを、指定したIPアドレスを元に制御したりする

どちらもポート番号の制御を行う

比較

|  |  |  |
| --- | --- | --- |
|  | セキュリティーグループ | ネットワークACL |
| 適用範囲 | リソース（インスタンス） | サブネット |
| 操作 | 各リソースの「セキュリティ」  （EC2 | VPC dashboard |
| ステート | 通信の状態を管理する（ステートフル） | 通信の状態を管理しない（ステートレス） |
| デフォルト設定 | インバウンド　全て拒否  アウトバウンド　全て許可 | 全て許可  （インバウンド、アウトバウンド） |
| ルール記述 | 許可ルールを記述　拒否ルールは記述できない | 許可／拒否を両方記述 |
| ルールの適用 | すべてのルールを適用し、判断 | ルール番号の小さい順にチェックし、通信内容に合致するルールで通信可否を判断 |
| 送信元／宛先 | IPアドレスの範囲  セキュリティグループ | IPアドレスの範囲 |
| 推奨の方針 | 厳しく、細かく設定 | 緩めに設定（全開放でも良いくらい） |
|  |  |  |

補足

ステートフル セッション状態を保持。アウトバウンド設定は不要。

ステートレス イン／アウトバウンドそれぞれに通信許可／拒否の設定が必要

ACL；Access Control List

比較

IAMポリシー 自身のAWSアカウントに対し、IAMユーザー単位でアクセス権限を設定。

###### セキュリティグループ

VPC上でネットワークアクセスをインスタンスごとに制御するファイアウォール

関連付けられたリソースに到達するトラフィックおよびリソースから離れるトラフィックを制御する。

###### ネットワークACL

1つまたは複数のサブネットに関連付けてイン／アウトバウンドのトラフィックを許可または拒否する。

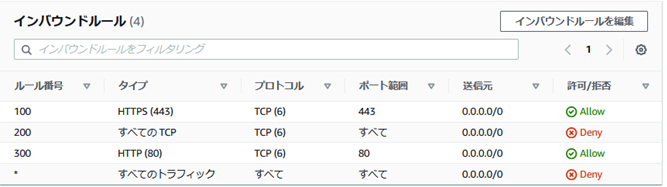
・ルール数が最大40

・「タイプ」「プロトコル」「ポート番号」を「許可／拒否」

・ルール番号の小さい順にルールが適用

ルール間で矛盾がある場合は、小さいルール番号のルールが優先

例：



タイプ： **単にポート番号**を意味する（プロトコルを意味する訳では無い）

プロトコル： 「タイプ」によってほぼ自動的に決まる

補足

「http(80) 、TCP(6)」という表現は「TCPプロトコルを使用してポート番号80での通信を制御する」の意。（非常に紛らわしい）

操作

VPC dashboard > ネットワークACL

メモ

　セキュリティーグループ EC2

　ネットワークACL

補足

　S3にもACLがあるが、これはバケットへのアクセス権限をAWSアカウント単位で設定する機能。

##### EIP；Elastic IP

###### 概要

AWSリソース（特にEC2）を使う為に用いられる静的な**グローバルIP(IPv4)**

（通常、AWSリソースは動的なグローバルIPで提供される）

次の条件がすべて満たされている限り、EIPに料金は発生しない。

・Elastic IP アドレスが EC2 インスタンスに関連付けられている。

・Elastic IP アドレスに関連付けられているインスタンスが実行中である。

・インスタンスには、1 つの Elastic IP アドレスしかアタッチされていない。

・Elastic IP アドレスが、アタッチされているネットワークインターフェイスに関連付けられている。

それ以外の場合は１時間単位で課金される

補足

・デタッチして別のEC2にアタッチする事が可能

・BYOIPアドレス（自分たちが持つパブリックIP）を利用する事もできる

比較

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 概要 | IP | 備考 |
| Elastic IP | インターネットと通信可能な**固定の**パブリックIP | 固定 | ・パブリックDNSホスト名も固定  ・AWSリソースが停止していたり、割り当てられていない時は料金が発生 |
| パブリックIP | AWSリソースを有効化すると割り付けられる**動的な**IPアドレス | AWSリソース削除時に解放 |  |
| プライベートIP | プライベートサブネットで利用。あとはパブリックIPと同条件 | AWSリソース削除時に解放 | パブリックサブネットにあるインスタンスに割り当てる事も可能  AWSリソースを停止して起動した後も、同一のIPアドレスが使われる |

公式：<https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html>

###### チュートリアル

EC2コンソールからEIPを払い出し

EC2 > Elastic IPアドレス > Elastic IPアドレスの割り当て

EC2インスタンスへの関連付け

EC2 > Elastic IPアドレス > アクション：Elastic IPの関連付け

Elastic IPの解放

EC2 > Elastic IPアドレス > アクション：Elastic IPアドレスの関連付けの解除

> アクション：Elastic IPの解放

##### ENI；Elastic Network Interface

AWSリソースで用いられる仮想的なNIC；Network Interface Cardの事。

ENI１つにつきIPアドレスが１つ割り振られる

・16進数文字16桁のIDが割り振られる（=MACアドレス？）

##### BYOIPアドレスプール

概要

自分たちが所有しているパブリックIPアドレスをクラウド環境で使う事ができる。

注意点

RIR；Regional Internet Registry、ISP；Internet Service Providerから適切な許可、書類等が必要。

##### 資料

はじめてのネットワーク構築：[https://mihono-bourbon.com/aws-network-1/#toc5](https://mihono-bourbon.com/aws-network-1/" \l "toc5)

#### VPC；Virtual Private Cloud

##### 概要

AWSアカウントに割り当てられる論理的な仮想ネットワーク空間、およびそのサービス。

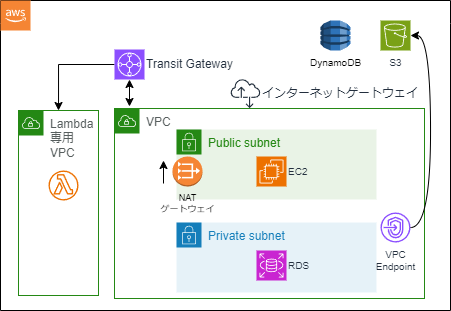
・VPCを複数のAZにまたがって構成することができる

・VPCではプレフィックス長を/16から/28のCIDRブロックを使用してIPアドレスの範囲を指定する

・セキュリティグループとネットワークACL（Access Control List）を使用してセキュリティを設定する

・デフォルトの状態では他のVPCやインターネットとは通信できない

・プライベートサブネット内のAWSリソースへアクセスしたい場合は「VPCアクセス」の設定を行う。もしくはNATゲートウェイ、NATインスタンスを経由する事でインターネットへ接続する。



参考

・ルートテーブルはサブネットに１つ（public/private問わず）に配置される

・

サービスの配置場所

|  |  |  |
| --- | --- | --- |
|  | サービス |  |
| VPC外  VPCエンドポイントのゲートウェイ型で接続 | S3，DynamoDB　など | S3はPrivateLinkでも接続可能 |
| VPC外  VPCエンドポイントのPrivateLinkで接続 | S3，CloudWatch Logs，Amazon SNS，AWS CloudFormation，Systems Manager，CodeBuild など |  |
| パブリック |  |  |
| プライベート | ELB配下のEC2、DB系 |  |

・Lambdaは専用のVPCがある

##### 操作

Management Console

検索で「VPC」 > VPC dashboardが開く

##### 各種ゲートウェイ

###### 概要

比較

|  |  |  |  |
| --- | --- | --- | --- |
|  | 配置場所 | 概要 | 備考 |
| Transit Gateway | VPC外 | 複数のVPCを繋ぐハブ |  |
| VPCピアリング | VPC内 | 2つのVPC間 |  |
| VPCエンドポイント | VPC内 | プライベートサブネット  => VPC外リソース |  |
| NATゲートウェイ | パブリックサブネット | プライベートサブネット  => インターネット | NATインスタンスも同等 |
| インターネットゲートウェイ | パブリックサブネット | インターネット<=> VPC |  |
| Egress-Onlyインターネットゲートウェイ | パブリックサブネット | NATゲートウェイ＋インターネットゲートウェイ | IPv6トラフィックのみ |
|  |  |  |  |

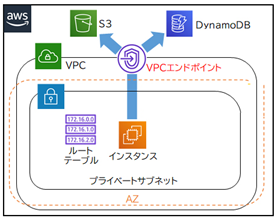
###### VPCエンドポイント

プライベートサブネットからVPC外のAWSサービスへ、インターネットゲートウェイを経由しなくてもアクセス可能にする機能。

ゲートウェイ型とAWS PrivateLink（インターフェイス型）の2種類ある

比較

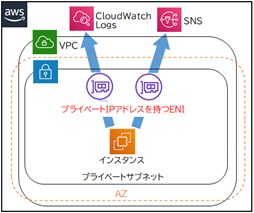
|  |  |  |
| --- | --- | --- |
|  | 対応サービス |  |
| ゲートウェイ型 | S3，DynamoDB | S3はPrivateLinkでも接続可能 |
| AWS PrivateLink | CloudWatch Logs，Amazon SNS，AWS CloudFormation，Systems Manager，CodeBuild など |  |

ゲートウェイ型

接続したいリソースが配置されているVPCにVPCエンドポイントを割り当て、ルートテーブルにターゲットがVPCエンドポイントのルーティングを設定。

補足

DynamoDBはゲートウェイ型のみ（S3はゲートウェイ型、PrivateLinkの両方で利用可能）

PrivateLink

サービスへ接続したいリソースが配置されているサブネットにプライベートIPアドレスを持つENIを作成し、ENIとサービスをリンクさせる。

・DynamoDB以外の多数のAWSサービスで利用できる

資料

公式（VPCクォータ）<https://docs.aws.amazon.com/ja_jp/vpc/latest/userguide/amazon-vpc-limits.html>

###### VPCピアリング

2 つの VPC 間でプライベートなトラフィックのルーティングを可能にするネットワーキング接続

VPC間のルーティングを可能にする機能

##### サブネット

###### 概要

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 外への接続 | 外からの接続 |  | 用途など |
| public | ○ | 可  （EIPなどが必要） |  | ・NATゲートウェイ／インスタンスを配置 |
| protected | × | 踏み台経由 |  |  |
| private | × | 踏み台経由 |  | ・プライベート接続でAWSリソースに接続 |

###### プライベートサブネット

補足

以下の理由で大体の場合はNATゲートウェイが必要となる

・apt, yumなどのパッケージマネージャーの動作の為

・NTPによる時刻同期

##### マルチAZ

プライマリインスタンスとスタンバイインスタンスを異なるAZに配置して可用性を高める事。

異なるAZへスタンバイ用のインスタンスを作成し、自動レプリケーション＋フェイルオーバー（切り替え）を実現するしくみ（）

|  |  |
| --- | --- |
|  |  |
| EC2 |  |
| RDS | 2つのAZまで |

比較

|  |  |
| --- | --- |
| プレイスメントグループ | アクセス速度を高めたりハードウェア障害を軽減する為の複数のEC2の論理的なグループ |

##### VPCフローログ

VPC内のENI（Elastic Network Interface）で通信するネットワークトラフィック情報をキャプチャする機能。

VPC外のネットワークトラフィック情報は収集対象外。

##### エンドポイントポリシー

PCエンドポイントからの接続先を制限する

・ゲートウェイ型とAWS PrivateLinkの両方で利用できる

・許可/拒否する接続先のAmazonリソース名（ARN）を指定する

・ポリシーはフルアクセスとカスタムを選択

フルアクセス VPC内の全AWSリソース？

カスタム 特定のAWSリソースのみ

具体例



### 主力サービス

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| Amplify | アプリ開発環境（バックエンド） |
| DynamoDB | Key-Value型NoSQL。容量無制限。 |
| EBS；Elastic Block Store | ブロックストレージ。EC2用のローカルストレージとして使う。 |
| EC2;Elastic Compute Cloud | 仮想サーバー　旧サービス？ |
| ELB;Elastic Load Balancing | 仮想サーバー　旧サービス？ |
| Lambda | GraphQLのFaaS。 |
| Redis | クラウド上で動作するインメモリのデータストア。キーバリュー型。  pythonのpickleと思えば良い? |
| S3 | Simple Storage Service　オンラインストレージ |
| VPCs |  |

#### Amplify

##### 概要

・Firebaseに似てる

・Webブラウザ用のコンソールと呼ばれる画面上でも一部操作可能だが、基本的にはCLIで操作する

公式ドキュメント：<https://docs.amplify.aws/>

###### カテゴリ

amplify add カテゴリ

auth 認証機能

##### チュートリアル

amplify全体：<https://aws.amazon.com/jp/builders-flash/202103/amplify-app-development/flutter>：

https://docs.amplify.aws/start/q/integration/flutter/

ユーザーを作成

　IAM > ユーザー > ユーザーを追加

　ユーザー名：amplify-dev（名前は何でも良い） > 許可のオプション　でポリシーを直接アタッチする

　> 検索ボックスに：amplifyと入力 > AdministratorAccess-Amplifyにチェック > ユーザーの作成

　補足：デバイス毎にユーザーを作るのが推奨されているらしい

アクセスキーを作成

　> 今作ったユーザーを選択 > セキュリティ認証情報 > Command Line Interfaceを選択 > 次へ

　アクセスキーを作成、csvをダウンロード

? accessKeyId: # YOUR\_ACCESS\_KEY\_ID

? secretAccessKey: # YOUR\_SECRET\_ACCESS\_KEY

ローカルマシンでamplify cliを使う方法

npm init

npm install -g @aws-amplify/cli

sudo npm install -g @aws-amplify/cli --unsafe-perm=true # rootユーザーの場合

amplify init

amplify configure

　リージョンの選択、

　access key, secretkey の入力

[Security credentials] (セキュリティ認証情報)

###### React

インストールからアプリ作成まで

npm install -g @aws-amplify/cli

npx create-react-app amplifyapp

cd amplifyapp

npm start

・先にリモートリポジトリを作成

・ローカルリポジトリからアップロード

# GitHubでユーザー認証を使う場合

git remote add origin github.com/amplifyapp

# GitHubでsshを使う場合

git remote add origin git@github.com:*usr*/amplifyapp

# 管理項目の追加とコミット、プッシュ

git add .

git commit -m “initial commit”

git push origin master

・AWS Management ConsoleでAmplify > 使用を開始する(GET STARTED) > Amplify HostingのGet started

> リポジトリにGitHubを選択 > （GitHubの認証が成功） > リポジトリとブランチを選択

> ビルド設定（.yml）を編集（と詳細設定） > 確認画面で保存してデプロイ

###### Android Studio

　アクティビティ：Empty Activity

　プロジェクト名：Todo（仮）

　Minimum SDK：Android 7.0(Nougat)

cd ~/AndroidStudioProjects/Todo

amplify init

? Enter a name for the project

`Todo`

? Initialize the project with the above configuration?

`No`

? Enter a name for the environment

`dev`

? Choose your default editor:

`Android Studio`

? Choose the type of app that you're building

`android`

? Where is your Res directory:

`app/src/main/res`

? Select the authentication method you want to use:

`AWS profile`

? Please choose the profile you want to use

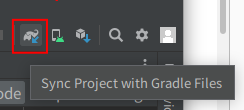
`default`

以下のものが作成される

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| amplifyのアプリ | amplify-todo-dev | CloudFormation::Stack | CREATE\_ |
| 一般IAMユーザー | UnauthRole | AWS::IAM::Role | CREATE\_ |
| S3のバケット | DeploymentBucket | AWS::S3::Bucket | CREATE\_ |
| IAM管理者ユーザー | AuthRole | AWS::IAM::Role |  |

最後に

Help improve Amplify CLI by sharing non sensitive configurations on failures (y/N)

Android Studioの方でGradleを使ってAWSと同期する。

##### GUI(Web)

AWSにログイン > リージョンを選択（IAMユーザーを作ったリージョンと同じにするこれが意外と重要！）

検索ボックスに Amplify > 使用を開始する

##### 詳細

###### Amplify　StudioとAmplify Hostingの違い

|  |  |
| --- | --- |
| Amplify Hosting | iOS/Android向き？ |
| Amplify Studio | React、Vue.js向き？ |

###### Amplify Hosting

ホスティングもととして選択できるもの（2023年11月現在）

GitHub

Bitbucket

GitLab

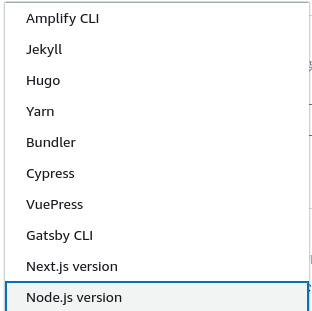
AWS CodeCommit

Gitプロバイダーなしでデプロイ

詳細設定

・構成イメージ（コンテナ？）https://docs.aws.amazon.com/amplify/latest/userguide/custom-build-image.html

・環境変数（Key-Value型）

・ライフパッケージ　右図参照

##### レファレンス

###### amplifyサブコマンド

|  |  |
| --- | --- |
| サブコマンド | 概要 |
| add api | バックエンドのAPIを作成 |
| configure | IAMユーザーなど、基本的な設定を行う |
| console | Amplify CLI（コンソール）にアクセス |
| console api | APIを確認 |
| delete | リソースをすべて削除 |
| mock | 開発テスト用のデータベースを構築する |
| mock api | ローカルでAPIのテスト |
| publish | 手動でホスティング |
| push | AWSにプッシュしてアプリを展開する |
| status |  |
| upgrade | amplifyをアップグレード |

Amplifyのライブラリをインストール

# Amplify CLI

npm install -g @aws-amplify/cli

sudo npm install -g @aws-amplify/cli --unsafe-perm=true # rootユーザーの場合

# React

npm install aws-amplify @aws-amplify/ui-react@1.x.x

###### aws-amplify

Auth

signOut

[ App.js ]

import { Auth } from ‘@aws-amplify/ui-react’;

function App(){

return (<a onClick={ Auth.signOut }>Sign Out</a>);

}

export default (App);

###### ui-react

withAuthenticator

使用例

[ App.js ]

import { withAuthenticator } from ‘@aws-amplify/ui-react’;

App = () => {}

export default withAuthenticator(App);

Authenticator

使用例

[ App.js ]

import { Authenticator } from ‘@aws-amplify/ui-react’;

function App(){

<Authenticator>

<!- ログインしないと表示できないコンテンツ ->

</Authenticator>

}

export default (App);

#### EBS；Elastic Block Store

ストレージ系サービスに記述

#### S3；Simple Storage Service

ストレージ系サービスに記述

#### EC2；Elastic Compute Cloud

##### 概要

・AMIに含まれないサーバー固有のOSの設定や、ミドルウェアの導入は個別に実施する

・ローカルストレージとしてアタッチできるのはEBS、EFS（Linuxなら）、インスタンスストア

アクション

|  |  |  |
| --- | --- | --- |
| Action | 概要 | 備考 |
| RebootInstances |  |  |
| StartInstances |  |  |
| StopInstances |  |  |
| TerminateInstances |  |  |

メトリクス

標準メトリクス

CPU使用率 (CPUUtilization)

ディスクの読み取り ・ 書き込み量　(DiskReadBytes /DiskWriteBytes)

ネットワーク使用率　(NetworkIn/NetworkOut)

カスタムメトリクス

メモリ使用量

ディスクの空き容量・使用状況

プロセス情報

##### チュートリアル

###### EC2インスタンス作成

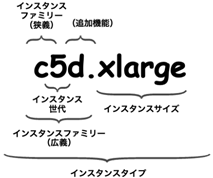
サービス検索 > EC2を検索 > インスタンスを起動

各種設定値の入力

・ AMI（マシンイメージ）

OS：Amazon Linux、macOS、Ubuntu、Windows、Red Hat、SUSE Linux、Debianなど

検索も可能

・インスタンスタイプ（スペック）

インスタンスファミリー、インスタンス世代、インスタンスサイズ

参考：初期値はt2.micro（無料利用枠）。m5.largeあたりが標準？

・キーペア

一応キーペアなしでの続行も可能（非推奨）

補足：予め作成したキーペアを利用する事もできる

・ ネットワーク設定

VPC、サブネット、ファイアウォール（セキュリティグループ）

・ ストレージのボリュームタイプを選択

参考：初期値gp2（汎用SSD）で良いと思う

8GB以上

・ 高度な設定 最初は初期値のままで

概要を確認して「インスタンスを起動」

メモ

SSHトラフィックを許可するIPに関連して

　接続もとが可変するグローバルIP

　curl inet-ip.info

　自宅PC　14.11.32.226 10/18

　自宅PC　14.11.32.226 10/19

###### 接続

AWS Consoleから

EC2 コンソール > ナビゲーションペイン：インスタンス > インスタンスを選択し、接続

> EC2 Instance Connectタグを選択 > ユーザー名を確認して接続

EC2 Instance Connect WebブラウザでConsole画面が開く

SSH/RDP接続のためのセキュアな認証・接続サービス

EC2 Instance Connectエンドポイント

AWS PrivateLinkを使ってVPCエンドポイントとして接続する

補足

・IPv6アドレスを使用した接続はできない

・sshでの接続方法もここの「SSHクライアント」タグで表示されている

公式： <https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/ec2-instance-connect-methods.html>

クライアント機から

鍵ファイルのpermissionを変更（これより厳しくてもゆるくてもだめ）

sudo chmod 400 /path/to/my\_key.pem # 秘密鍵のpermission変更

クライアントからの接続の場合、パブリックIPv4 DNS、パブリックIP、Elastic IPのいずれかが必要。

例：

ssh -i my\_key.pem ec2-usr@ec2-\*\*\*\*\*.amazonaws.com # DNSでログインする場合

ssh -i my\_key.pem ec2-usr@elastip\_ip\_address # Elastic IPまたはpublic IPv4

公式：<https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/connect-linux-inst-ssh.html>

memo

　EC2インスタンスの起動のたびにIPが変わるのかも

ubuntu@ec2-54-91-136-177.compute-1.amazonaws.com　10/19

###### 資料

参考；AWSでのEC2構築：<https://qiita.com/yo_dazy/items/3846915e4db21fe04537>

公式：<https://aws.amazon.com/jp/ec2/getting-started/>

10/18に試したときの「コマンドの確認」

{

"MaxCount": 1,

"MinCount": 1,

"ImageId": "ami-053b0d53c279acc90",

"InstanceType": "t2.micro",

"KeyName": "ec2\_test\_key",

"EbsOptimized": false,

"BlockDeviceMappings": [

{

"DeviceName": "/dev/sda1",

"Ebs": {

"Encrypted": false,

"DeleteOnTermination": true,

"SnapshotId": "snap-0d3283808e9f92122",

"VolumeSize": 8,

"VolumeType": "gp2"

}

}

],

"NetworkInterfaces": [

{

"AssociatePublicIpAddress": true,

"DeviceIndex": 0,

"Groups": [

"<groupId of the new security group created below>"

]

}

],

"TagSpecifications": [

{

"ResourceType": "instance",

"Tags": [

{

"Key": "Name",

"Value": "ec2\_test"

}

]

}

],

"PrivateDnsNameOptions": {

"HostnameType": "ip-name",

"EnableResourceNameDnsARecord": true,

"EnableResourceNameDnsAAAARecord": false

}

}



・２つ以上のボリュームを持つことができる

##### Management Consoleでの操作

###### 概要

良く使うものを抜粋

|  |  |  |
| --- | --- | --- |
| 大分類 | 項目 | 概要 |
|  | EC2ダッシュボード |  |
|  |  |  |
| インスタンス | インスタンス |  |
| インスタンスタイプ |  |
| ネットワーク&セキュリティ | Elastic IP | Elastic IPの割り当てなどを行う |

###### インスタンスの停止

ナビゲーションペイン：インスタンス（またはECダッシュボードの「リソース」 ：インスタンス）

> インスタンスを選択（チェック）> インスタンスの状態

##### AMI；Amazon Machine Image

OSやミドルウェアなどのテンプレート（≒DockerImage）

自分で作成する事も可能。（EC2インスタンスを停止した状態で作成する事を推奨）

OS：Amazon Linux、macOS、Ubuntu、Windows、Red Hat、SUSE Linux、Debianなど



|  |  |  |
| --- | --- | --- |
| OS | AMI | 備考 |
| Amazon Linux |  |  |
| macOS |  |  |
| Ubuntu | Server 22.04 LTS  Server 20.04 LTS  Server Pro 22.04 LTS  Server 22.04 LTS(SSD type)  Deep Learning AMI GPU  Deep Learning AMI GPU Pytorch  Deep Learning AMI Neuron Pytorchなど |  |
| Windows | Server 2022 Base  Server 2022 Core Base  Server 2019 Base  Server 2019 Core Baseなど |  |
| RedHat | Linux9  Linux 9 with High Availability  RHEL 8 with SQL Server 2022  Linux 9 SSD typeなど |  |
| SUSE |  |  |
| Debian |  |  |
|  |  |  |

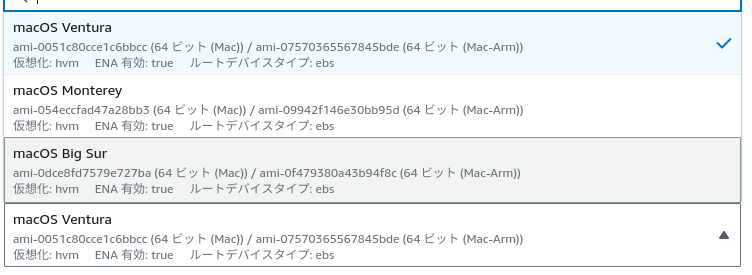
無料

・Amazon Linux 2023 AMI

・Ubuntu Server 22.04 LTS (HVM), SSD Volume Typeなど

・Microsoft Windows Server 2022 Base

・SUSE Linux Enterprise Server 12 SP5 (HVM), SSD Volume Type



##### キーペア

###### 概要

EC2へのSSHログインする際の公開鍵と秘密鍵のセットの事。

・ログインユーザー１人につき１つのキーペアを作成する

###### チュートリアル

1: EC2 > ネットワーク＆セキュリティ > キーペア

2: キーペアの作成ボタン

3: 各種設定値の入力　[キーペア名、キーペアのタイプ、プライベートキーファイル形式]

4: キーペアがつくられる

キーペア名 全角文字は使用不可

キーペアのタイプ 暗号方式の事

|  |  |  |
| --- | --- | --- |
| 暗号方式 | 特徴 | 備考 |
| RSA[def.] | ・2048, 3072, 4096bitの長さで使用される | ・長い歴史を持つ暗号方式  ・最も普及している  ・既存のシステムで使用する |
| ED25519 | ・256bitの固定長  ・処理速度が速い | ・安全面と性能面で最良  ・システムによる制限が無い限りこちらを使う |

プライベートキーファイル形式

|  |  |  |
| --- | --- | --- |
| 形式 | ソフト | 概要 |
| .pem | OpenSSL | X.509証明書、RSA秘密鍵、DSA秘密鍵など多種多様な情報を保存 |
| .ppk | PuTTY | WindowsユーザーかつPuTTYを使う場合のみ、と思って問題ない |

注意！鍵ファイルのダウンロードはこのタイミングしか無い。**ここを逃したら秘密鍵は二度と手に入らない**。

公式：[https://docs.aws.amazon.com/ja\_jp/AWSEC2/latest/UserGuide/ec2-key-pairs.html?icmpid=docs\_ec2\_console#having-ec2-create-your-key-pair](https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/ec2-key-pairs.html?icmpid=docs_ec2_console" \l "having-ec2-create-your-key-pair)

##### ネットワーク設定

サブネット

パブリックIPの自動割り当て

Firewall

・セキュリティグループを作成

・既存のセキュリティグループを選択

SSHからのトラフィックを許可

インターネットからのHTTP、HTTPSトラフィックを許可

##### インスタンスストア

無料で利用できるブロックストレージ

・高パフォーマンス。数百万を超えるIOPS。

・揮発性（EC2インスタンスを停止すると削除される）

> 使いどころは非常に難しい

・インスタンスストアは特定のEC2インスタンスにアタッチ（別インスタンスに紐づけることはできない）

・AMI作成の際もインスタンスストアは対象外

比較：EC2での利用の観点で

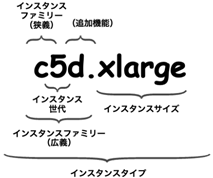
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 料金 | 容量上限 | データ永続化 | 備考 |
| EBS；Elastic Block Store | 有料？ | 16TiB | 設定で永続化可能 | ・スナップショットの作成が可能 |
| インスタンスストア | 無料 |  | 必ず削除  > S3などに都度コピーが良手 | ・高パフォーマンス。  ・暗号化非対応 |

比較：ファイル共有の観点で

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | EC2に直接  アタッチ | 保存単位 | 容量 | 同時  アクセス | その他／備考 |
| S3 | できない | オブジェクト | 無制限 | 可 | ・HTTP/HTTPS経由でアクセス |
| EBS | できる | ブロック | 16TB | 不可 | ・ローカルストレージとして使う。 |
| EFS | Linuxベースなら可能  （プライマリストレージとしては使えない） | ファイル | 無制限 | 可 | ・複数のEC2インスタンスで同じファイルシステムを利用したいようなケースで利用  ・コンテナ、オンプレでも使用可能  ・AWS Backupでバックアップ  ・単一のAZで動作  ・NFSに対応 |

##### インスタンスタイプ

###### 概要

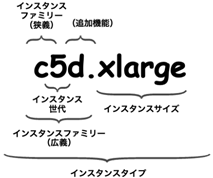
EC2インスタンスに適用するCPU、メモリ、ストレージ性能、ネットワークキャパシティーなどのスペックの組み合わせ。

逆に言うとCPU、メモリなどのスペック面での柔軟な変更はできない。

|  |  |
| --- | --- |
| 項目 | 概要 |
| インスタンスファミリー | 用途別の分類 |
| インスタンス世代 | 数字が大きいほど新しい世代 |
| インスタンスサイズ | CPU、メモリなどのスペック |

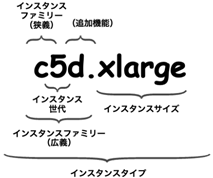
###### インスタンスファミリー

|  |  |  |
| --- | --- | --- |
|  | 分類 | 用途 |
| M  Most Scenarios | 汎用 | 特定のリソースに負荷が偏っていない、 多くの環境に適しています。 |
| T；Turbo | CPUバースト  （汎用） | CPU処理のバーストが可能  アイドル時間の多い小中規模のデータベースサーバー  開発環境に適しています。  t2.microは無料　<https://aws.amazon.com/jp/ec2/instance-types/t2/> |
| C；Compute | 高性能なCPUを提供 | 負荷の高いバッチ処理  アクセス頻度の高いWebサーバー  コンピューティング負荷の高い環境 |
| A；Arm |  |  |
| R；RAM  X；eXtra large memory z；hertZ | メモリ最適化 | 大容量のメモリを提供  リアルタイムでのビッグデータ解析を処理する  アクセス頻度の高いデータベースサーバ |
| P；general Purpose  G；Graphics  F；FPGA  VT；  Video Transcoding  Inf；Inference | 高速コンピューティング | 高性能なGPUを提供するインスタンスファミリーです。機械学習推論や3Dグラフィックスを大量に処理するサーバーなど、 高速演算が求められる環境に適しています。 |
| I；I/O Performance  D；Dense storage  H；High disk throuput | ストレージ最適化 | 高性能なローカルストレージを提供するインスタンスファミリーです。高いディスクスループットや、 大容量のローカルストレージが求められる環境に 適しています。 |



###### 追加機能

|  |  |  |  |
| --- | --- | --- | --- |
| 文字 | 概要 |  |  |
| a | AMD製CPU | Intel製と比較しコスパ良 |  |
| g | CPUは AWS Graviton2 を搭載 | Intel製と比較しコスパ良 |  |
| n | ネットワークを強化 |  |  |
| d | NVMe ベースのローカルSSD（高速）を追加 |  |  |
| e | メモリ搭載量を強化 (x1の場合) |  |  |



m5の場合の例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| インスタンスサイズ | vCPU | ECU | メモリ  [GB] | NW 帯域幅 [Gbps] | EBS 帯域幅 [Mbps] | 料金単価  (/h) | 備考 |
| large | 2 | 10 | 8 | 最大10 | 最大4750 | $0.12 |  |
| xlarge | 4 | 16 | 16 | 最大10 | 最大4750 | $0.25 |  |
| 2xlarge | 8 | 37 | 32 | 最大10 | 最大4750 | $0.50 |  |
| 4xlarge | 16 | 70 | 64 | 最大10 | 4750 | $0.99 |  |
| 8xlarge | 32 | 128 | 128 | 10 | 6800 | $1.98 |  |
| 12xlarge | 48 | 168 | 192 | 10 | 9500 | $2.98 |  |
| 16xlarge | 64 | 256 | 256 | 20 | 13600 | $3.97 |  |
| 24xlarge | 96 | 337 | 384 | 25 | 19000 | $5.95 |  |

###### 資料

公式：<https://aws.amazon.com/jp/ec2/instance-types/>

参考；インスタンスタイプについて：<https://www.openupitengineer.co.jp/column/awsengineer/4585>

##### 高度な設定



##### 購入オプション

###### 概要

インスタンスの購入オプション

大きく「共用ハードウェアを使用するインスタンス」と「ハードウェアを専有するインスタンス」に分けられる。

###### 共有ハードウェアを使用するインスタンス

|  |  |  |
| --- | --- | --- |
| 購入オプション | 特徴 | 用途 |
| オンデマンドインスタンス | ・利用時間に対する従量課金  ・デフォルトの購入オプション | 開発環境など、 利用が短期間のシステム |
| リザーブドインスタンス | ・長期契約 (1年または3年)を結ぶことで、オンデマンド インスタンスより低価格で利用可能 | 本番環境など、長期間にわたって常時稼働することが決まっているシステム |
| スポットインスタンス | ・未使用のEC2インスタンスをスポット価格で契約。  ・オンデマンドインスタンスより低価格で利用可能  ・利用中にインスタンスが中断する可能性がある | 並列バッチ処理の一部のインスタンスなど、インスタンスが停止しても問題ないシステム |

設定方法

スポットインスタンス 作成時：高度な設定 > スポットインスタンスをリクエスト

ナビゲートペイン：スポットリクエスト

リザーブドインスタンス ナビゲートペイン：リザーブドインスタンス

補足

スポットフリート

スポットインスタンスのオプションの一つで、必要なインスタンス数を指定できる。

オンデマンド／スポットで購入したインスタンスでインスタンス数の維持を行う

スポットブロック

EC2インスタンスが終了しない時間帯を指定する

2021年7月に廃止

###### ハードウェアを専有するインスタンス

|  |  |  |
| --- | --- | --- |
| 購入オプション | 特徵 | 用途 |
| ハードウェア専有 インスタンス  (Dedicated Instances) | ・他の顧客とは分離された環境を用意し、インスタンス単位で契約  ・物理的なCPUソケット、 コア数、ホストIDは確認できない  ・物理ホスト単位で購入したソフトウェアライセンスの 持ち込み (BYOL)はサポートしない(※1) | 会社のコンプライアンス対応で、 他のAWSアカウントとは 分離された専用ハードウェアでEC2インスタンスを利用したい場合など |
| 専有ホスト  (Dedicated Hosts) | ・ホスト（物理サーバー）単位で契約  ・物理的なCPUソケット、 コア数、ホストIDが確認できる  ・BYOLをサポートする | ・利用するソフトのライセンス制約への対応。  ・ハードウェアの細かい制御が必要な場合 |

ハードウェア専有 インスタンス

別名：Dedicated Instances

EC2を1つ起動する度に、AWSからハードウェア（ホスト）が1つ割り当てられるイメージ

ユースケース

・コンプライアンス対応(他組織との同居を禁止する場合)などで利用

専有ホスト

別名：Dedicated Hosts

名前の通り、専有のホスト。

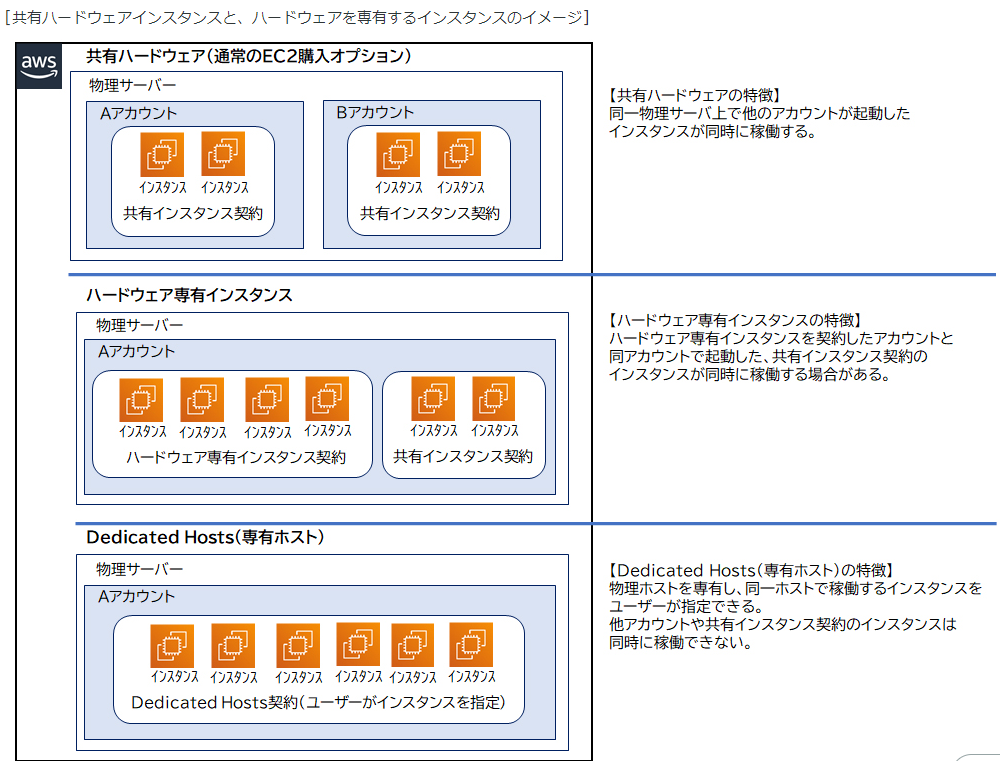
どのハードウェアで起動するかを固定することができる。

###### BYOL

別名：Bring Your Own Licence

利用者が予め購入済みのライセンスをAWSに持ち込んで使用すること

物理ホスト単位で購入したソフトウェアライセンスの 持ち込み



##### プレイスメントグループ

###### 概要

複数のEC2インスタンスを論理的にグループ化し、インスタンス間での低遅延な通信や、ハードウェア障害による影響を軽減できる機能

まとめ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 目的 | 別AZのインスタンス配置 | 概要 | 図 |
| クラスター  プレイスメントグループ | 通信遅延を少なくする | 不可 | 同一のAZ内に配置する。  高速な通信が可能 |  |
| パーティション  プレイスメントグループ | ハードウェア障害による影響を低減 | 可 | ・「ラック」を共有するインスタンスグループ（これをパーティションと呼ぶ）を形成  ・グループ内のEC2インスタンスが同一のラックを共有することがある |  |
| スプレッド  プレイスメントグループ | ハードウェア障害による影響を最小限にする | 可 | 別々の「ラック」にインスタンスを配置 |  |

類似機能の比較

|  |  |
| --- | --- |
| マルチAZ | 複数のAZに配置されたプライマリとバックアップインスタンス。 |

###### ラック

独立した電源とネットワークが配備されている

###### クラスタープレイスメントグループ

通信遅延は少ないが、物理的な障害等が発生した時の可用性は落ちる。

グループ内のインスタンスがハードウェアを共有することもある

###### パーティションプレイスメントグループ

ハードウェア障害による影響を低減

グループをパーティションとよばれる論理グループに分割し、それぞれのパーティションに1つ以上のEC2インスタンスを所属させる。

各パーティションは1つのラックが対応しており、ラックには独立した電源とネットワークが配備されている。

###### スプレッドプレイスメントグループ

グループ内のEC2インスタンスを異なるハードウェア（ラック）に配置

それぞれのラックに1つだけEC2インスタンスを配置。

比較

ハードウェア専有インスタンス

他のAWSアカウントとは分離された専用ハードウェアでEC2インスタンスを起動

##### 詳細

###### Action（ポリシー）

|  |  |  |
| --- | --- | --- |
| Action | 概要 | 備考／具体例 |
| Describe |  |  |
| StartInstance |  |  |
| StopInstance |  |  |

##### その他機能など

###### メタデータ

実行中のインスタンスに関するデータ

http://url/latest/meta-data/ でアクセスできる

instance-id

public-ipv4

host name

IAMロールをEC2インスタンスにアタッチ　するとどうなる？

##### 具体例

バックアップ

基本的には

|  |  |  |
| --- | --- | --- |
| EBSのスナップショット |  |  |
| 自作のAMIを作成 |  |  |

#### ELB；Elastic Load Balancing

##### 概要

負荷分散装置

例えば複数のECインスタンスのうちリソース使用率が高いものがある場合に利用する。

⇒ コンテンツを配信にのみもんだいがある場合はCloudFront

ALBでHTTPリスナーを作成？[調査] 2023-10-24

##### ロードバランス方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ALB | NLB | CLB | GLB |
| 動作するレイヤー | レイヤー7  （アプリケーション層） | レイヤー4  （トランスポート層） | レイヤー4,7 |  |
| サポートするプロトコル | HTTP、HTTPS | TCP, UDP, TLS | HTTP、HTTPS、TCP、 SSL/TLS |  |
| WebSocketへの対応 | 〇 | 〇 | × |  |
| SNI；Server Name Indication  に対応 | 〇 | 〇 | × |  |
| 同一インスタンス内の複数ポートへの負荷分散 | 〇 | 〇 | × |  |
| ホスト、パス、URLクエリ文字列でのルーティング | 〇 | × | × |  |
| 固定IPアドレスの割当て | × | 〇 | × |  |
| 備考 | ・WAFの割り当てが可能  ・リダイレクト機能 | EIP（固定IP）と関連付け、ポート番号による制御 |  | 2021年にリリース |

ALB；Application Load Balancer

・HTTP、HTTPSをサポート

・WebSocketや、同一インスタンス内の複数ポートへの負荷分散

・リクエストを別のURLにリダイレクトする機能がある

・ホストベースやパスベース、URLクエリ文字列でのルーティングに対応。

（コンテンツごとにELBを用意する必要がなく、１つのALBで複数コンテンツを管理できる）

|  |  |
| --- | --- |
| ホストベース | リクエスト元のURLのFQDNでルーティング |
| パスベース | ドメイン名より後のURLでルーティング |
| URLクエリ文字列 | ?lang=en, ?lang=jp などでルーティング |

補足

ホスト FQDN前半部

パス ドメイン名移行の文字列

NLB ；Network Load Balancer

・WebSocketや、同一インスタンス内の複数ポートへの負荷分散に対応。

CLB ；Classic Load Balancer

・ALBやNLBより古いサービスでパフォーマンスが劣る。（ALBやNLBが推奨）

##### リダイレクト機能

ALBにはリクエストを別のURLにリダイレクトする機能がある。

IF、THENの方式で設定を行う



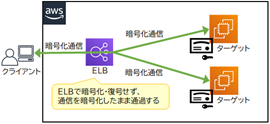
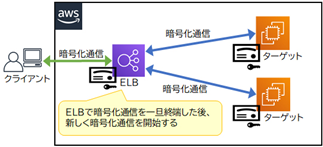
##### アクセスログ

ELB配下のインスタンスにアクセスしてきたクライアントの接続情報を記録する

##### 暗号化

ELB経由でクライアントからWebサーバーまでのすべての通信を暗号化する２つの方法

ELBとターゲットにサーバー証明書を導入する ターゲットにサーバー証明書を導入する



##### SNI；Server Name Indication

ELBからドメイン名が複数あるインスタンスへ暗号化通信を転送する

言い換えると、サーバー証明書をパブリックIPアドレスではなくドメイン名によって判断する

##### ホストベースのルーティング

ユーザーがリクエストした接続先URLのFQDNに従ってルーティングできる機能

ALBではホストベースのルーティングを利用しなくても、HTTPリスナーとHTTPSリスナーを両方作成することで、それぞれ専用のWebサイトに転送できる。

##### ヘルスチェック

主な設定項目

|  |  |
| --- | --- |
| 項目 | 説明 |
| 正常のしきい値 | 「非正常」状態のリソースを 「正常」 と判断するまでのヘルスチェック連続 「成功」 回数 |
| 非正常のしきい値 | 「正常」状態のリソースを 「非正常」 と判断するまでのヘルスチェック連続 「失敗」 回数 |
| タイムアウト | ヘルスチェックを失敗と見なす、 応答がない時間 |
| 間隔 | ヘルスチェックの間隔 |

説明

「正常」状態のリソースを 「非正常」 と判断するまでのヘルスチェック連続 「失敗」 回数 |

##### 管理画面

CloudWatchメトリクス

正常/非正常な状態のターゲット数

#### Lambda

##### 概要

FaaS（Function as a Service）。Apexなどを使う？

Java、Node.js、Pythonなどに対応

・１回につき最長15分

・Lamba専用のVPCに配置される

比較

|  |  |
| --- | --- |
| API Gateway | あくまでゲートウェイ。他のAWSサービスと連携するAPIの窓口 |
|  |  |

##### トリガ

|  |  |
| --- | --- |
|  |  |
| S3バケットのオブジェクトの更新 |  |
| DynamoDBのレコードの更新 |  |
| AmazonSNS？にメッセージ発行 |  |
| CloudWatch Eventsのイベント発生 |  |
| CloudFrontのイベント発生 |  |
|  |  |
|  |  |

##### 実行

###### EC2インスタンスの起動／停止

起動の例：

import boto3

def lambda\_handler(event, context):

ec2 = boto3.client(‘ec2’)

response = ec2.run\_instances(

ImageId=‘AMI\_ID’,

InstanceType=‘INSTANCE\_TYPE’,

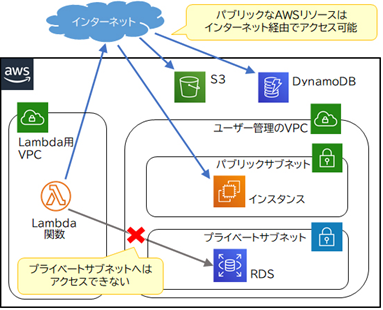
MinCount=1,

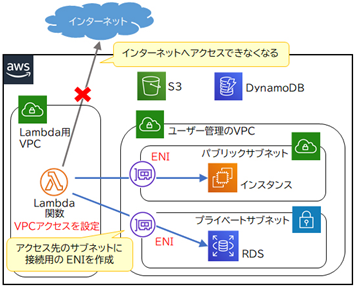
MaxCount=1

)

return ‘EC2インスタンスが起動しました。’

##### Lamba専用VPC

Lambdaは専用のVPCに配置され、インターネットや、インターネットを経由してパブリックサブネット内のAWSリソースにはアクセスできるが、プライベートサブネット内のAWSリソースへはアクセスできない。

Lambda関数からプライベートサブネット内のAWSリソースへアクセスさせたい場合は「VPCアクセス」の設定する

Lambda関数がサブネット毎に接続用のENI（Elastic Network Interface）が作成される

・VPCアクセスを設定したLambda関数は、ENIを作成したサブネットへアクセスできるようになる代わりに、インターネットへアクセスできなくなる。

⇒ VPCアクセスを設定したLambda関数からインターネットへアクセスしたい場合は、プライベートサブネットからインターネットへアクセスする方法と同様に、**NATゲートウェイまたはNATインスタンスを経由**する。

#### Lightsail

3.5$から始められる仮想サーバー。

Dockerのように、「OS+アプリ」のイメージが利用可能

WordPress, LAMP, Node.js, Joomla, Magento, MEAN, Drupal, GitLab, Redmine, Nginxなど

##### 具体的な用途

###### MQTTのブローカー

普通にmosquittoが入るっぽい

MQTTは一回最大128kb

lightsailの一番安いプラン（$3.50）が1TB/月 ⇒ 7,812,500回/月

補足：T=1012、G=109

#### Redis

読み方：レディス

##### 概要

クラウド上で動作するインメモリのデータストア。pythonのpickleなどをイメージすると良い。

具体的には、離れた場所で動いている２つ以上のアプリケーション用の共通のキャッシュ、といった使われ方をする。

・スナップショットを使用したバックアップ／リストアが可能

・SSLやパスワード認証による機密性

参考：Memcached

#### RDS

DB系サービスに記述

### ストレージ系サービス

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| EBS；Elastic Block Store | **ブロック**ストレージ。EC2用のローカルストレージとして使う。 |
| EFS；Elastic File System | **ファイル**ストレージ。NFSをサポートする。 |
| FSx | **ファイル**ストレージ。高パフォーマンス向き。 |
| Glacier | **アーカイブ**保存専用の**ファイル**ストレージ |
| Instance Store | EC2の揮発性**ブロック**ストレージ |
| S3；Simple Storage Service | **オブジェクト単位**のオンラインストレージ。 |
| Snowball | オンプレミス環境からS3へ大容量のデータをオフラインで移行するサービス |
| Snowcone | 8TBストレージ付きの小型デバイスでデータ移行を行うサービス |
| Storage Gateway | オンプレからNFS/SMBを介してS3へ高速かつセキュアにアクセスするゲートウェイ |

似たサービスの比較

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | EC2に直接  アタッチ | 保存単位 | 容量 | 同時  アクセス | その他／備考 |
| S3 | できない | オブジェクト | 無制限 | 可 | ・HTTP/HTTPS経由でアクセス |
| EBS | できる | ブロック | 16TB | 不可 | ・ローカルストレージとして使う。 |
| EFS | Linuxベースなら可能  （プライマリストレージとしては使えない） | ファイル | 無制限 | 可 | ・複数のEC2インスタンスで同じファイルシステムを利用したいようなケースで利用  ・NFS対応  ・コンテナ、オンプレでも使用可能 |

Storage Gatewayとの比較

オンプレからNFS/SMBを介してS3へ高速かつセキュアにアクセスするゲートウェイ

|  |  |
| --- | --- |
| ファイルゲートウェイ | NFSまたはSMBを介してS3バケットへアクセス |
| ボリュームゲートウェイ | S3にデータを保存しつつ、オンプレをキャッシュとして  S3バケットをバックアップとして使う |

補足

オブジェクト単位 階層／入れ子などの構造がなくフラットに管理できる

#### 共通事項

EC2 AMIやLambda関数（※）をデバイスに設定しておくことができます。転送するデータに特殊な処理を行いたい場合や分析を行いたいようなケースで活用できる。

#### FSx

##### 概要

ファイルストレージサービス。

・S3バケットとの連携（インポート／エクスポート）が可能

⇒ 高頻度アクセスのデータをFSx For Lustre上に配置し、アクセスしないデータはAmazon S3へ保存する と言った使い方ができる

補足：S3はオブジェクトストレージ

・SMB対応

比較

EFS

Linuxベースで利用可能なファイルストレージ。

EC2にアタッチ可能だがプライマリストレージとしては使えない。

FSxにはファイルシステムのタイプが以下の2種類ある。

Amazon FSx for Windows File Server

Windowsベース 比較：EFS（Linuxベース）

Active Directoryとの連携が可能

SMBプロトコルを介してアクセス（対応させればLinuxOSからもアクセスできる）

Amazon FSx for Lustre

機械学習や動画処理など速度が必要な高性能ファイルシステム

大量のクライアントから低レンテンシー、最大1秒あたり数百ギガバイトのスループットで処理が可能

補足

- Lustre(ラスター)はOSSのファイルシステム

- Lustreという名前はLinuxとClusterのかばん語

次ページへ続く

～FSx　つづき

##### FSx for Windows File Server

Windowsファイルサーバーが提供する機能の内、以下の機能がFSx for Windows File Serverで提供される

（Windowsが提供する機能そのもの；Windowsカーネルがインストールされている？）

|  |  |
| --- | --- |
|  | 説明 |
| NTFS  (NT File System) | Windowsにおける標準のファイルシステム。 |
| SMB  (Server Message Block) | 主にWindowsコンピュータ間において、ファイル共有やプリンタ共有に使用されるプロトコル。 |
| AD  (Active Directory) | Windowsのサーバーに搭載されている機能で、ユーザーやコンピューターを "ドメイン"という単位で認証を統合したり、アクセス権限を一括管理する機能。 |
| Windows ACL  (Access Control List) | ACLはADによって実現されるアクセス制御。ファイルやフォルダなどのリソースに対し、 ユーザーごとにアクセス許可/拒否を設定できる。 |
| DFS 名前空間  (Distributed File System  Namespace) | DFS とは、 Windowsコンピュータ間で複数のコンピュータに存在する 共有フォルダなどを一括管理する機能。  DFS 名前空間は異なるコンピュータ上の共有フォルダへ統一された名称でアクセスできるようにしたもの(例:"¥¥ADドメイン名¥共有名¥フォルダ名")。 |
| DFSR  (DFS Replication) | DFSRは共有フォルダ間を関連付けて自動的に複製する機能。  特定のコンピュータへのアクセス集中を防いだり(負荷分散)、意図しない削除からファイルを保護する (可用性の向上) ことができる。 |

HPC（High Performance Computing）環境に対応したストレージサービス

##### FSx For Lustre

Linuxベースのファイルシステム

主にスーパーコンピューターなどのHPC（High Performance Computing）や機械学習の分野で利用

・コンテナ（FargateやECSで実行されるコンテナなど）からLustreへアクセスする際は、IAMロールによる権限設定が必要

・POSIXに準拠したファイルシステム

補足

POSIX（ポジックス） UNIX系OSに共通する機能などの標準を定めた規格。

#### EBS；Elastic Block Store

##### 概要

EC2インスタンスにアタッチして使われるAWSのストレージサービス（EC2以外では使えない）

・１つのECインスタンスに１つのEBSボリュームをアタッチするのが基本。

・１つのECインスタンスには複数のEBSボリュームをアタッチできる。

・AZごとに独立している。（別のAZからのアタッチ、別のAZへのスナップショットの作成などは不可）

・上限は16TiB

・AZ内の異なるハードウェアに複数回コピーされて保存されることで、冗長化を実現している

・KMS；Key Management Serviceと併用した暗号化が可能（すべてのボリュームタイプが対象）

⇒ KMSで事前にキーを作成しEBSの「マスターキー」欄へ入力

一度暗号化したら変更はできない。変更したい場合は新たにEBSボリュームを作り直す必要がある。

・複数のECインスタンスからのアタッチに対応したマルチアタッチ機能があるボリュームも存在する

・デフォルト設定ではEC2インスタンス終了でデータが削除されてしまう

関連：DeleteOnTermination

※スナップショットによるバックアップは可能

・DLM；Data Lifecycle Managerでのスケジューリングされたスナップショットの作成が可能

・EC2インスタンス終了時にデータ破棄をする事ができる

・複数ボリュームのRAID 0（アクセス速度改善）を組む事ができる

比較

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 料金 | 容量 | 永続化 | 備考 |
| EFS；Elastic File System |  | 無制限 | ・複数AZにレプリケート  ・AWS Backup | あくまでファイルサーバー  NFSに対応 |
| EBS；Elastic Block Store | 有料？ | 16TiB | 設定で永続化可能  又はスナップショットの作成 |  |
| インスタンスストア | 無料 | ？ | 必ず削除  > S3などに都度コピーが良手 | 高パフォーマンス。  暗号化非対応 |

##### 設定項目

|  |  |  |
| --- | --- | --- |
|  | 概要 | def. |
| DeleteOnTermination | EC2インスタンス終了時にEBSを削除する | True |
|  |  |  |

ふぁいるしす



##### ECインスタンスのコピー

方法は以下の２つ

・EBSスナップショットからAMIを作成してEC2インスタンスを起動

・EBSスナップショットからEBSボリュームを作成してEC2インスタンスにアタッチ

備考

・EBSボリュームのコピーはできない。EBSスナップショットはコピー可能。

・インスタンスストアは特定のEC2インスタンスに紐づいており、コピーして別のインスタンスに紐づけることはできない

##### マルチアタッチ

一部のボリュームタイプでは複数インスタンスからのアタッチが可能な「マルチアタッチ」機能に対応しています。ただし、利用できるインスタンスタイプやリージョンが限られているなどの制限事項がある。

##### ボリュームタイプ

###### 概要

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ボリューム  タイプ | 主な利用用途 | ボリューム  サイズ | 最大IOPS | 最大  スループット |
| 汎用SSD  (gp2, gp3) | EC2インスタンスの起動ボリュームや開発・テスト環境など | 1 GiB  ～ 16TiB | 16,000 IOPS | 250 MiB/s (gp2)  1,000 MiB/s (gp3) |
| プロビジョンドIOPS SSD  (io1) | 汎用SSDより高い性能が要求される システム用  大規模データベースなど | 4 GiB  ～ 16 TiB | 64,000 IOPS | 1,000 MiB/s |
| プロビジョンドIOPS SSD  (io2) | io1よりも、 高信頼性・高性能が要求されるシステム向け。 io1の100倍の耐久性と、容量あたり10倍のIOPSを発揮できる | 4 GiB  ～ 16 TiB | 64,000 IOPS | 1,000 MiB/s |
| スループット最適化 HDD  (st1) | ビッグデータや、**シーケンシャル(順次)アクセス**を行うデータベース、 ログの処理など | 125GiB  ～ 16 TiB | 500 IOPS | 500 MiB/s |
| コールド HDD (sc1) | 利用頻度の低いデータの保持用。 ログやバックアップなど | 125GiB  ～ 16 TiB | 250 IOPS | 250 MiB/s |

補足

・スループット最適化 HDD、コールド HDDはEC2インスタンスの起動ボリュームには指定できない

・IOPS=Input Output per Second

インスタンスストアとの比較

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ボリューム  タイプ | 主な利用用途 | ボリューム  サイズ | 最大IOPS | 最大  スループット |
| SSD |  | 50GiB  ～ 60TiB | 数百万IOPS |  |
| HDD |  | ～ 48TiB | 数百万IOPS |  |

補足

スループット 単位時間あたりに処理できる量

IOPS 1秒当たりにディスクが処理できるI/Oアクセスの数

###### gp2

###### gp3

#### EFS；Elastic File System

##### 概要

複数のEC2インスタンスからのアクセスを前提としたファイルサーバー

・NFS（Network File System）プロトコルをサポート

・ストレージ容量は無制限

利用している容量に従ってストレージサイズが自動的に拡張・縮小される

補足：EBSでは16TiBが上限

・EC2インスタンスのプライマリストレージ（OSやシステムファイルなどを格納するストレージ）としては利用できない。

・保管データは自動的に暗号化

・マウント時にTLSを使うことも可能（通信の暗号化）

・インスタンスでファイルを更新する際にファイルのロック（排他制御）機能がある

・ボリュームタイプの選択は無い

関連：EBSでは4つのボリュームタイプを選択可能

・複数のAZにデータが自動的にレプリケートされる

設定

|  |  |  |
| --- | --- | --- |
|  | 概要 | def. |
| 自動バックアップ | AWS Backupによるバックアップ | 有効 |
| ライフサイクル管理 | 低頻度アクセスのファイルを移動させるしきい値 | 30日 |
| パフォーマンスモード | 汎用  最大I/O | 汎用 |
| スループットモード | バースト　ファイルシステムサイズに応じてスケール  プロビジョニング済み　固定のスループット | バースト |
| 暗号化 | 保存時のデータの暗号化 | 有効 |

・パフォーマンスモードはファイルシステムの作成後は変更できない

補足：転送中のデータの暗号化（TLSの使用）は、EFSをマウントする際にユーザーが指定

クライアントでのマウント時、マウントオプションに「-o tls」を指定

sudo mount -t efs **–o tls** fs-0d74f175:/ efs

比較

FSx

WindowsファイルサーバーまたはLustreファイルシステムをAWS上で利用できるファイルストレージ

##### スループットモード

バーストスループットモード

|  |  |  |
| --- | --- | --- |
| モード | 概要 | 備考 |
| バースト | ファイルシステムサイズに応じてスケール | ・特定の時間帯に一時的に負荷が高くなるような場合  ・負荷に応じてベースライン性能の2倍の性能を発揮 |
| プロビジョンド  （プロビジョニング済み） | 固定のスループット | ・小さなデータに高頻度でアクセスが発生する場合に向く |

##### パフォーマンスモード

最大I/Oパフォーマンスモード

マウントするEC2インスタンスが数千台規模で、アクセス頻度も多い場合

ビッグデータの解析向き

補足：数百万IOPSを発揮することはできない

##### 他サービスとのプロトコル比較

|  |  |  |  |
| --- | --- | --- | --- |
|  | プロトコル |  |  |
| EFS | NFS |  |  |
| ファイルゲートウェイS3 | NFS/SMB |  |  |
| ファイルゲートウェイFSx | SMB |  |  |
| ボリュームゲートウェイ | iSCSI |  |  |
| FSx for Windows File Server | SMB |  |  |
| FSx for Lustre |  |  |  |

補足

EBS：EC2インスタンスにアタッチするストレージサービス

##### その他の機能など

###### 暗号化

KMSで管理するキーで行う

#### Glacier

##### 概要

データをアーカイブして保存するファイルストレージで、費用は安い（S3の約10分の1）がアクセスするには時間が掛かる、といったサービス

##### 復元リクエスト

アーカイブをダウンロードの為にS3バケットに取り出す事（Glacier Flexible RetrievalとGlacier Deep Archiveはファイルの直接ダウンロードができない）

標準取り出し

S3 Glacier、S3 Glacier Deep Archiveに保存されているデータを取り出す際のデフォルト。

|  |  |  |  |
| --- | --- | --- | --- |
| 対象 | 迅速 | 取り出し時間 | 備考 |
| Glacier Flexible Retrieval  Intelligent-Tiering Archive | 1~5分 | 3〜5 時間以内 |  |
| Glacier Deep Archive  Intelligent-Tiering Deep Archive | 利用不可 | 12 時間以内 |  |

公式：<https://docs.aws.amazon.com/ja_jp/AmazonS3/latest/userguide/restoring-objects-retrieval-options.html>より

但し、データ量に関しての明言が無いが？

迅速取り出し

追加料金を払って標準取り出しよりも迅速にデータを取り出せます。「迅速取り出し」オプションは250MBまでのデータであれば通常1～5分以内に使用可能になります。

大容量（バルク）取り出し

S3 Glacierにある大容量のデータを安価に取り出せるオプション

大容量のデータを取り出す場合に、標準取り出しより時間がかかりますが取り出し料金が低価格。

・

・ペタバイト単位のデータを1日かけて取り出す。

一括取り出し

#### S3；Simple Storage Service

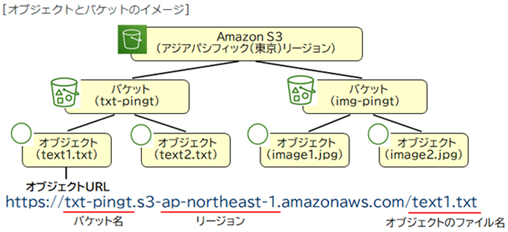
##### 概要

・AWSで利用できる容量無制限のオブジェクトストレージ

・保存したデータを自動的に3つのデータセンターに複製。障害、エラーなどに強い。

・Amazon S3と連携するNAS製品がある（家電量販店で売られているレベルでも）

・NFSプロトコルは非対応



・バケット名はあとから変更できない

・バケット名はグローバルで一意にする必要がある

・１オブジェクトごとにURLが付与される

・保存できるオブジェクトの最大サイズは5TB

・EC2のローカルストレージとしては使用できない

・AWSのストレージサービスの中で1TBあたりの保存料金が最も安い

・160 GB を超えるファイルをアップロードするには、AWS CLI、AWS SDK、または Amazon S3 REST APIを使う

用途

・アプリケーション、Webサーバーなどのバックアップ

・htmlのstaticファイルの配置

次ページへ続く

～S3 - 概要　つづき

料金

・ざっくり、保管：10GB 　25円/月、ダウンロード：10GB 　114円/月

|  |  |  |  |
| --- | --- | --- | --- |
| リージョン | 保管料金 | ダウンロード料金 | アップロード料金 |
| 東京 | 25円 (10GBあたり) | 114円 (10GBあたり) | 0円 (10GBあたり) |
| バージニア | 23円 (10GBあたり) | 90円 (10GBあたり) | 0円 (10GBあたり |

アクション

|  |  |  |
| --- | --- | --- |
| Action | 概要 | 備考 |
| ListBucket | バケット内のオブジェクト一覧 | バケットを指定する。（オブジェクトを指定しない） |
| GetObject | オブジェクトの追加 | オブジェクトの追加 |
| PutObject | オブジェクトの更新／追加 | オブジェクトの更新／追加 |
| DeleteObject | オブジェクトの削除 | オブジェクトの削除 |

ListBucket

対象はbucketで、bucket内のファイルではない

"Action" : "ListBucket",

"Resource" : "arn:aws:s3:::bucket-sample", # arn:aws:s3:::bucket-sample/\* ではない

比較

|  |  |  |
| --- | --- | --- |
|  |  | 備考 |
| EFS ；Elastic File System | 複数のEC2からのアクセスを前提とした**ファイルサーバー** | NFS対応 |
| Elastic Block Store | EC2インスタンスのローカルストレージ（**プライマリストレージ**）として使う |  |

##### チュートリアル

###### バケットを作成

S3 > バケットを作成 >

・バケット名、リージョンを選択

・オブジェクト所有者（ACL）

・パブリックアクセス

新しいアクセスコントロールリスト (ACL) を介して付与されたバケットとオブジェクトへのパブリックアクセスをブロックする

任意のアクセスコントロールリスト (ACL) を介して付与されたバケットとオブジェクトへのパブリックアクセスをブロックする

新しいパブリックバケットポリシーまたはアクセスポイントポリシーを介して付与されたバケットとオブジェクトへのパブリックアクセスをブロックする

任意のパブリックバケットポリシーまたはアクセスポイントポリシーを介したバケットとオブジェクトへのパブリックアクセスとクロスアカウントアクセスをブロックする

・バージョニング

・タグ - オプション

公式：https://docs.aws.amazon.com/ja\_jp/AmazonS3/latest/userguide/CostAllocTagging.html?icmpid=docs\_amazons3\_console

・デフォルトの暗号化

　SSE-S3, SSE-KMS, DSSE-KMS（バケットキー非対応）から選ぶ

　バケットキー？

備考：アップロード処理で保管時の暗号化の詳細を定義していないバケットにアップロードされたオブジェクトに適用される

・詳細設定

　オブジェクトロック

###### IAMの設定

IAMの設定

・EC2 インスタンスからアクセスする場合 IAM ロールを作成

・オンプレからのアクセス IAM ユーザーを作成

・IAMユーザーを作成

Management Console：IAM > ユーザー > ユーザーの作成 > 次へ

　補足：AWS Management Consoleへのユーザーアクセスを提供する のチェックは入れない

> ポリシーを直接アタッチする。以下の２つが基本。（慣れてきたら自分でjsonを編集すると良い）

　AmazonS3FullAccess

　AmazonS3ReadOnlyAccess

参考：

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:\*",

"s3-object-lambda:\*"

],

"Resource": "\*"

}

]

}

・アクセスキーを作成

　IAM > ユーザー > ユーザーを選択 > アクセスキーを作成 > ユースケース：コマンドラインインターフェイス

説明タグ値には半角文字しか使えない？

.csvファイルがダウンロードされる

###### アップロード／ダウンロード

設定

[ terminal ]

aws configure

もしくは環境変数で設定

$ export AWS\_ACCESS\_KEY\_ID=アクセスキー

$ export AWS\_SECRET\_ACCESS\_KEY=シークレットキー

$ export AWS\_DEFAULT\_REGION=ap-northeast-1

補足：~/.aws/credentialsに記録されるらしい

[default]

aws\_access\_key\_id = YOUR\_ACCESS\_KEY

aws\_secret\_access\_key = YOUR\_SECRET\_KEY

アップロード

Terminal を使う場合

[ terminal ]

aws s3 cp ファイル名 s3://{バケット名}

aws s3 cp memo.txt s3://208076554333-wani-test

[調査] pythonでの送受信方法

[調査] キーを動的に変更する方法（環境変数以外の方法）

Pythonを使う場合

[ python ]

import boto3

client = boto3.client(

's3',

region\_name=ap-northeast-1,

aws\_access\_key\_id='AKIAIO5FODNN7EXAMPLE',

aws\_secret\_access\_key='ABCDEF+c2L7yXeGvUyrPgYsDnWRRC1AYEXAMPLE'

)

公式：<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>

###### Management Consoleを使ったアップロード

・アクセス許可

・プロパティ

　- ストレージクラスの選択

・サーバー側の暗号化

・追加のチェックサム

・タグ - オプション

・メタデータ - オプション

###### バケットの削除

まずは空にする

S3 > バケット > 任意のバケットを選択 > 空にする

削除

S3 > バケット > 任意のバケットを選択 > 削除

##### 料金

「保存されたデータ容量＋データへのリクエスト数＋データ転送量」の従量課金

##### 暗号化

###### SSE；Server-Side Encryption

別名：サーバー側の暗号化

以下３種類の方法がある

|  |  |  |
| --- | --- | --- |
| 方法 | 概要 | 備考 |
| SSE-S3 | S3が管理している鍵を使用する | CSE（クライアント側暗号）では使用できない |
| SSE-KMS | KMSに保存されているキーを使用する | 鍵の利用状況を自動的に記録 |
| SSE-C | ユーザーが管理している鍵を使用する |  |

SSE-C CはCustomerの意

###### CSE；Client-Side Encryption

別名：クライアント側の暗号化

以下2種類の方法がある：

・KMSに保存されているKMSキーを使用する

・クライアント側に保存したルートキーを使用する

補足

ルートキーとは？？別名「マスターキー」？

・CSEではS3が管理する鍵は使用できない

###### 具体例

・KMSで暗号化されたオブジェクトをLambda関数でダウンロードして復号する

⇒ Lambda関数にアタッチされているIAMロールをKMSのキーユーザーとして追加

##### アクセス制御

###### 概要

S3リソースへのアクセス権限を設定。

|  |  |  |  |
| --- | --- | --- | --- |
| ポリシー | 対象 | アクセス元IP,ドメイン単位での制御 | 概要 |
| バケットポリシー | バケット | ○ | バケット単位でアクセス権限を設定 |
| IAMポリシー | バケットもしくはオブジェクト | ○ | 自AWSアカウント配下のIAMユーザー単位でアクセス権限を設定 |
| ACL | バケットもしくはオブジェクト | × | AWSアカウント単位でアクセス権限を設定 |

比較

|  |  |  |
| --- | --- | --- |
| ポリシー | 対象 | 備考 |
| 署名付きURL | オブジェクト | S3 Glacierの機能 |

###### ACL；Access Control List

アクセスコントロールリスト。**AWSアカウント**単位でアクセス権限を設定する。

言い換えると、他のAWSアカウントに対して、オブジェクト／バケットへの読み取り/書き込みを許可する。

###### バケットポリシー

S3で保存しているデータにバケット単位でアクセス権限を設定する機能。

出来る事

・アクセス元のIPアドレスやドメイン名によるアクセス制御

・バケットに対してサーバー側の暗号化（SSE）を強制する

操作

S3 > バケット > バケットを選択 > 「アクセス許可」タブ

###### ユーザー（IAM）ポリシー

IAMユーザー単位でアクセス権限を設定する機能。

・自AWSアカウントのIAMユーザーやグループ、ロールに対して、S3リソースへのアクセス権限を設定

・他のAWSアカウントには設定できない。

###### 署名付きURL

非公開設定されたオブジェクトに対して有効期限のついたURLを発行し、AWSアカウントを持っていないユーザーでも一時的にアクセスが可能になる。

##### オブジェクトロック

###### 概要

S3バケットに保存したオブジェクトに対して更新・削除を制限する機能。

補足：追加に関しては対象外。S3へのアクセス制御機能で実現。

・S3バケット作成時にのみ設定可能

・オブジェクトロックを有効にするとバージョニング機能も有効になる。

・

・保持期間が無期限の「リーガルホールド」と、期限付きの「リテンションモード」の2種類がある

・リーガルホールドとリテンションモードは、同時に両方を有効にすることもできる

- リーガルホールド

権限を持たないユーザーに対して、オブジェクトを読み取り専用にする。

・権限を持つユーザーのみオブジェクトの更新・削除、リーガルホールドの解除ができる。

・具体的な権限　s3:PutObjectLegalHold

- リテンションモード

ガバナンスモード

権限（s3:BypassGovernanceRetention）を持たないユーザーに対して、指定した保持期間中オブジェクトを読み取り専用にします。権限を持つユーザーのみオブジェクトの更新・削除と、ガバナンスモードの解除ができます。

コンプライアンスモード

全ユーザー（ルートユーザーを含む）に対して、指定した保持期間中オブジェクトを読み取り専用にする。

・保持期間の指定が必須

・期間中は**ルートユーザーであってもコンプライアンスモードを解除できない**。

###### 操作

Management Console

S3 > バケット > バケットを選択 > プロパティ

##### クロスリージョンレプリケーション

データを**自動的に**異なるリージョンに保存する機能

設定項目

ソースの設定

[ 新しいバケットの作成、]

バケット名、リージョン

送信先の設定

規則のオプションの構成

##### ストレージクラス

###### 概要

利用者の目的に合わせて選択できるサービスの種類の事

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ストレージクラス | 保管料(10G目安) | 取出し料 | 取出し遅延 | 保存AZ | 可用性 | 最短保存期間 | 推奨頻度 | 備考 |
| S3 RRS  低冗長化ストレージ |  |  |  |  |  |  |  | Reduced Redundancy Service |
| S3 Standard | 25円 | 無し | 無し | 3以上 | 99.9% | 無し |  | S3 標準 |
| S3 Standard-IA | 10円 | 有り | 無し | 3以上 | 99.9% | 30日間 | ひと月に１度 |  |
| S3 One Zone-IA |  | 有り | 無し | 1 | 99.5% | 30日間 | ひと月に１度 |  |
| S3 Glacier Instant Retrieval |  | 有り | ミリ秒単位 | 3以上 |  | 90日間 | 四半期に１度 |  |
| S3 Glacier Flexible Retrieval |  | 有り | 数分~数時間 | 3以上 | 99.9% | 90日間 | 年に１度 |  |
| S3 Glacier Deep Archive | 2円 | 有り | 数時間以上 | 3以上 | 99.9% | 180日間 | 年に１回未満 |  |
| S3 Intelligent-Tiering | 無し | 無し | 無し | 3以上 | 99.9% | 無し | 頻度不明な場合に選択 |  |

補足

IA Infrequent Access；低頻度アクセス

Intelligent-Tiering 値段は変動

One Zone-IA

・単一AZでの管理なのでデータの可用性はやや難あり。

・One Zone-IAストレージクラスを除いて自動的に複数のAZへコピーされる？

操作

ファイルアップロードの時：プロパティ

###### One Zone-IA

ユースケース

・保存するデータは再作成可能なので、可用性は低くても良い

・データ消失リスクよりも低コストであることを優先したい

・保存データへのアクセスは低い（月1回程度）が、取り出し時の遅延は許容されない

###### Intelligent-Tiering

概要

アクセス頻度の予測が難しいデータに適している。

・利用状況に応じて「高頻度アクセス用ストレージ」か「低頻度アクセス用ストレージ」へ自動的にオブジェクトを移す。

・

アーカイブ構成

連続 90 日間アクセスされなかったオブジェクトをアーカイブアクセス階層に自動的に移動する、といった事ができる。　Intelligent-Tieringだけの機能？2023-10-24

階層

|  |  |  |
| --- | --- | --- |
| 名称 | 非アクセス期間 | 概要 |
| 非同期アーカイブアクセス階層  （Archive Access階層） | 90日 |  |
| ディープアーカイブアクセス階層  （Deep Archive Access階層） | 180日 |  |

操作

プロパティ > Intelligent-Tiering Archive 設定 > 設定を作成

設定

設定スコープ アーカイブ構成を適合するファイル条件を指定するか、バケット全体に適用するかを決める

フィルター

プレフィックス オブジェクト名に対して設けるフィルター

オブジェクトタグ このタグの値に基づいてアーカイブ層にオブジェクトを入れるかどうかを決める

アーカイブルール

###### Glacier

ボールトロック

「誰に対して」「どの動作を」「どの期間」「許可 or 拒否」という設定に従って、ボールト内のアーカイブに制限をかけます。

###### 低冗長化ストレージ（RRS）

別名：Reduced Redundancy Service

・99.99% の耐久性と、99.99% の可用性を提供

（通常S3は１年に対して99.999999999% の耐久性と、99.99% の可用性を提供）

・2016年以降、標準ストレージクラスの方が安くなったので非推奨

・頻繁にアクセスされる重要度が低いデータを保存

##### ライフサイクルポリシー

別名：ライフサイクルルール

時間経過にともない設定変更やオブジェクト管理を自動化する機能

・ストレージクラスの変更

・指定した期間が経過したデータを自動的に異なるパフォーマンスのストレージへ移動

・保管期限の過ぎたオブジェクトを削除

アクセス頻度が予測できるデータや、保管期間が決められているデータのバケットに設定することで、コスト削減の効果が期待できる。

具体例

・頻繁にアクセスされる時期はデータを「S3 Standard」に保存し、30日後にはアクセス数の低下が見込まれるので「S3 Standard-IA」へ移動させる。

操作

S3 > バケット > 任意のバケットを選択 > 管理 > ライフサイクルルール

具体的に適用できるルールは

オブジェクトの最新バージョンをストレージクラス間で移動

オブジェクトの非現行バージョンをストレージクラス間で移動

オブジェクトの現行バージョンを有効期限切れにする

オブジェクトの非現行バージョンを完全に削除

##### バージョニング

オブジェクトの世代管理ができる機能。デフォルトでは無効。

更新時

更新前と更新後の両方のオブジェクトが保存され、各オブジェクトに固有のバージョンIDが割り当てられる。

削除時

オブジェクトを完全に削除する代わりに削除フラグを意味する削除マーカーが付与され、それが最新のバージョンとなります。

・変更履歴の保持

・誤ってデータの上書きや削除をしてしまっても元のデータを復元できる。

操作

Management Console

S3 > バケット > バケットを選択 > プロパティ

補足

世代

以前のバックアップデータも保存し続ける事。フル、差分、増分の３つの観点でバックアップファイルを扱う。

差分：前回のフルバックアップからの差分 増分：前回のバックアップから変更のあった分

##### その他の機能

###### 静的サイトホスティング

・HTTPのみ

> HTTPSを利用したい場合：CloudFrontと併用する

###### マルチパートアップロード

一度にアップロードできる最大サイズ（5GB）を超えるファイルをアップロードする際に利用する

・分割されたファイルはそれぞれ並列にアップロードされる

（アップロード時間の短縮にもなる）

・ファイルサイズが100MB以上の場合は、マルチパートアップロードの使用を推奨。

###### S3 API

REST APIを使用してS3へ直接アクセスする

特徴

・レイテンシがそれなりに高い

###### リクエスタ支払い

S3オブジェクトを他のAWSアカウントのユーザーと共有する際にバケット内のオブジェクトの転送料金がアクセス元に請求する設定。

操作

Management Console

S3 > バケット > バケットを選択 > プロパティ > リクエスタ支払い

###### イベント通知

オブジェクトの作成や削除などをトリガに通知を行う機能。

通知先：

|  |  |
| --- | --- |
|  |  |
| Lambda |  |
| SQSキュー |  |
| SNSトピック |  |
| EventBridge |  |

##### 詳細

###### Action(ポリシー)

|  |  |  |
| --- | --- | --- |
| ポリシー | 概要 | 備考／具体例 |
| ListBucket | バケット内のオブジェクト一覧 |  |
| GetObject | オブジェクトの追加 |  |
| PutObject | オブジェクトの更新／追加 |  |
| DeleteObject | オブジェクトの削除 |  |
| ListAllMyBuckets | 権限のあるバケットのリスト表示 |  |
| PutObjectLegalHold | リーガルホールドじに更新ができる |  |
| BypassGovernanceRetention | ガバナンスモード時に更新・削除、解除ができる権限 |  |

Resourceの書き方に注意

"arn:aws:s3:::mybucket" # 「mybucket」バケット　の意

"arn:aws:s3:::mybucket/\*" # バケット内のオブジェクト全ての意

参考：https://www.jpcyber.com/support/minimal-iam-policy-to-access-amazon-s3

###### Transfer Acceleration

遠方のリージョンなどにあるS3へのデータ転送をAWSのエッジロケーションとネットワークプロトコルの最適化で高速化するサービス

###### MFA Delete

S3のバージョニング機能で管理されているオブジェクトを削除する際に、MFA認証を必要とする機能。

MFA Deleteを利用すると、ルートユーザーのみが、世代管理されたデータの削除権限を持つようになる。

補足

IAM 世代管理したデータには対応していない（？）

###### コマンド

こんな感じらしい

[ terminal ]

backup\_dir="/mnt/usbhdd"

target\_s3dir="s3://backetname/backup/usbhdd"

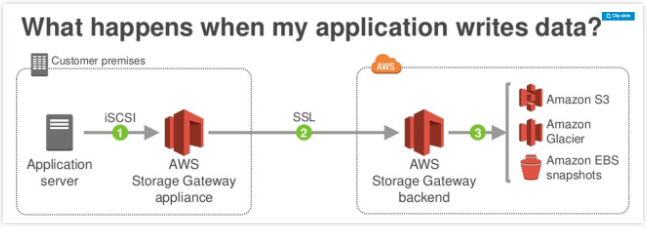
aws --profile=s3backup s3 sync ${backup\_dir} ${target\_s3dir} --delete --storage-class GLACIER\_IR

詳細

#### Storage Gateway

##### 概要

オンプレからNFS/SMBを介してS3バケットへアクセスするゲートウェイ



・LinuxおよびWindowsに対応

・リストア先はAWSまたはオンプレミスを選択できる

比較（データ移行）

Storage Gateway オンプレからNFS/SMBを介してS3へ高速、セキュアにアクセスするゲートウェイ

DataSync オンプレミスと、S3、EFS、FSxなどの間でデータを移行

##### ゲートウェイタイプ

概要

ファイルゲートウェイ

・オンプレミスからNFSまたはSMBを介してS3バケットへアクセス

・Windows環境のAD；Active Directoryに対応

ボリュームゲートウェイ（キャッシュ型；cached volume）

全データをS3に保存、高頻度アクセスのデータのみオンプレミスのキャッシュストレージに複製。

・キャッシュにないデータはS3にアクセスする事になる。

・オンプレのストレージ容量を抑えられる

・リストア先はオンプレミスまたはEBS

・プロトコルはiSCSIにのみ対応（NFSやSMBには非対応）

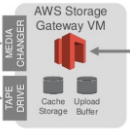
ボリュームゲートウェイ（保管型；stored volume）

全データをオンプレで管理、S3に非同期で複製。

比較

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| サービス名 | | 形式 | プロトコル | 概要 |
| ファイルゲートウェイ | S3 | ファイル単位 | NFS/SMB  →HTTPS | S3をファイル共有ストレージとして利用 |
| FSx | ファイル単位 | SMB  →HTTPS | FSxをWindows用共有ストレージとして利用 |
| ボリュームゲートウェイ | キャッシュ型  cached volume | ボリューム単位 | iSCSI  →HTTPS | 高頻度アクセスのデータのみローカルにコピー |
| 保管型  stored volume | ボリューム単位 | 全データをオンプレで管理、S3に非同期で複製 |
| テープゲートウェイ | | 仮想テープ単位 | VTL→  HTTPS | 物理テープストレージの代替としての利用 |

キャッシュ型 保管型 テープゲートウェイ



補足

VTL；Virtual Tape Library 仮想テープライブラリ。通信プロトコルではない

NFS；Network File System UNIX系で用いられる分散ファイルシステムおよびそのプロトコル

##### ユースケース

・オンプレミスにある複数のWindowsクライアントからAWSでファイル共有

⇒ FSxファイルゲートウェイでSMBを使用して、オンプレミスからFSx for Windows File Serverのボリュームにアクセスする

・S3を共有ストレージとして利用

・災害対策目的でのバックアップやアーカイブを行う

#### Snowファミリー（Snowball）

##### 概要

|  |  |  |  |
| --- | --- | --- | --- |
|  | 概要 | スペック | 物理的な大きさ |
| Snowcone | ・8TBデバイスでデータ転送・保存を行う。  ・エッジコンピューティングに対応したサービス。  ・AWS Datasyncに対応 | 8TB | ストレージ  約2kg |
| Snowball | ・最大１ペタ程度のデータ転送サービス。  ・Edge Storage OptimizedとEdge Compute Optimizedがある  ・Lambda関数の設定が可能 | 40~80TB | ストレージ  約22kg |
| Snowmobile | ・エクサバイト規模のデータ転送サービス。  ・データ転送に数か月必要  (2021年現在、日本国内での提供は無い) | ～100PB？  エクサ対応らしい | トレーラー  （約14mのコンテナに100PBのストレージを搭載したトレーラー ） |

##### Snowball

ペタバイト規模のデータ転送サービス。 ユースケースに応じた 2つのオプションがある:

Snowball Edge Storage Optimized

ストレージとしての利用が主目的。 ローカルストレージや 大規模なデータ転送を行うケースに適している。

・デバイスには内蔵HDD80TBが搭載されている。

・それ以上の容量を転送したい場合は複数のSnowballデバイスを申請する

補足：snowcone（かき氷）と比較して、容量などの面で「雪だるま級の玉」をイメージすると良いかも

Snowball Edge Compute Optimized

CPU、メモリを多く搭載した**高性能**なオプション

エッジコンピューティング向けサービス。 vCPUやメモリが 潤沢に搭載されている。 ネットワークに接続されていない 状態でも高度な情報処理を行うケースに適している

### 仮想化／環境構築系

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| CloudFormation | テキストファイルからAWSリソースをプロビジョニング（構築）する |
| Elastic Beanstalk | アプリのデプロイを行うための定番環境を自動で構築 |
| ECR；Elastic Container Registry | AWS版のDocker Hub  コンテナイメージを登録・管理するサービス |
| ECS；Elastic Container Service | AWS版のKubernetes  コンテナを実行、管理するサービス |
| EKS；Elastic Kubernetes Service | AWSまたはオンプレミスでKubernetesを実行するサービス |
| Fargate | サーバーレスでコンテナを動かすためのサービス |
|  |  |

#### CloudFormation

##### 概要

主に

JSONやYAMLでAWSリソースの設定を定義し、各種AWSサービスの構築や設定を行う

コード例

"Parameters": {

"KeyName": {

"Description" : "EC2KeyPair SAMPLE",

"Type" : "AWS::EC2::KeyPair::KeyName",

},

"DBName": {

}

}

特徴

・負荷予測による最適化を容易にする

比較

|  |  |
| --- | --- |
| サービス | 概要 |
| CloudFormation | AWSリソース基盤（インフラ；開発環境）の定義 |
| Elastic Beanstalk | アプリのデプロイ、プロビジョニング管理  （裏でCloudFormationが動いている？） |
| OpsWorks | アプリのデプロイ、プロビジョニング管理（Chefを利用） |

##### スタックセット

複数のAWSアカウントで、または複数のリージョンでスタックを構築したい場合に利用。

ターゲットアカウントとプロビジョニングを行うリージョンを指定することで、一度の操作で複数の同じ環境を構築できます。

テスト環境を複数構築したい場合や、新たにアカウントを払い出す際に下準備となる操作（IAMやセキュリティグループの設定など）をスタックセットとして作成しておく、といった利用ができます。

#### Elastic Beanstalk

##### 概要

アプリのデプロイ、プロビジョニングを管理するための定番環境（EC2、S3、RDS）を自動で構築

アプリケーションが動作する定番の環境を自動で構成する

補足：プロビジョニング

アプリ配信の為の手続き、アクセス権の設定、必要なリソースの割り当てなどの事

・EC2インスタンスは、OSはもちろん、Webサーバーソフトウェアやアプリケーションの実行環境などがインストール・設定された状態で提供。

（仕様書を書くだけで実行環境が構築されるイメージ）

・単一インスタンス・複数インスタンス（Multi-AZ）の構成を選択でき、関連する各種AWSサービス（ELBやS3、Auto Scalingグループなど）も設定済み

具体的には、以下の項目を設定する：

|  |  |  |
| --- | --- | --- |
| 項目 | 設定値 | 備考 |
| OS | Amazon LinuxまたはWindows Server |  |
| 言語  （プラットフォーム） | Java、.NET、PHP、Ruby、Python、Node.js、Go、Docker |  |
| データベース | RDS |  |
| コンテナ | シングルコンテナ構成とマルチコンテナ構成 | Dockerを使用する事もできる |

プラットフォームのブランチ

プラットフォームのバージョン 純粋にアプリケーションのバージョン

比較[調査]

|  |  |
| --- | --- |
| サービス | 概要 |
| CloudFormation | AWSリソース基盤（インフラ；開発環境）の定義 |
| Elastic Beanstalk | アプリのデプロイ、プロビジョニング管理  （裏でCloudFormationが動いている？） |
| OpsWorks | アプリのデプロイ、プロビジョニング管理（Chefを利用） |

##### プリセット

単一インスタンス

単一インスタンス（スポットインスタンスを使用）

高可用性

高可用性（スポットインスタンスとオンデマンドインスタンスを使用）

カスタム設定

##### コンテナ

特徴（箇条書き）

・シングルコンテナ構成とマルチコンテナ構成がある

・コンテナ内の環境の管理はユーザーが行う

##### より多くのオプションの設定

負荷分散を考慮して複数のインスタンスを構築

データベースサービスのRDSに関する設定

Redshiftは選択できない

#### Fargate

別名：ファーゲート

サーバーレスでコンテナを実行

・タスクが実行されていない時は料金が発生しない

#### ECR；Elastic Container Registry

≒ Docker Hub

ECRに登録されたコンテナイメージをECSが参照し、コンテナ（タスク）を起動できます。

・ECRのようにコンテナイメージを登録・管理する仕組みを「コンテナレジストリ」という

#### ECS；Elastic Container Service

##### 概要

AWS独自のコンテナオーケストレーションサービス（≒Kubernetes）

EC2やFargate上でコンテナを起動

・タスクが実行されていなくても料金がかかる？

比較

|  |  |  |
| --- | --- | --- |
|  | ECS | EKS |
| 概要 | Dockerコンテナの管理とデプロイに特化 | Kubernetesクラスタのマネージメントに焦点 |
| オーケストレーションエンジン | AWSの独自のエンジン | Kubernetes |
| 推奨利用 | モノリシックなアプリケーションの運用  AWSの他のサービスとの連携 | マイクロサービスアーキテクチャ  クラウドネイティブなアプリケーションの運用 |
| 詳細 | ・AWSの他のサービスとの連携に向く  ・AWSのマネージドサービスとして簡単に操作および管理が可能 | ・柔軟性とポータビリティが高い  ・Kubernetesの標準的な機能やエコシステムを利用 |

用途

モノリシックなアーキテクチャを4つのマイクロサービスとして分割

補足：モノリシック

複数の機能やコンポーネントが1つの大きな単位で組み込まれている形態

対義語：マイクロサービス

比較

Fragate サーバーレスでコンテナを実行

##### 主要要素（クラスター，タスク，サービス）

ECSの主要要素は「クラスター」「タスク」「サービス」

タスク

タスク定義に基づいて起動するコンテナの集まり

タスク定義

タスクを作成するテンプレート定義

設定項目

- コンテナ実行環境のCPUやメモリのスペック

- コンテナから他のAWSサービスへアクセスするための権限

公式：<https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/task_definitions.html>

サービス

実行中のタスクを管理する。必要なタスク数を維持。

設定項目

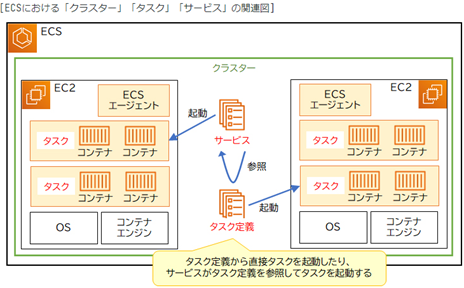
- タスクとELBの連携

- 必要なタスク数　など

クラスター

サービスとタスクを実行する基盤

コンテナが動作するVPCやサブネット



##### 起動タイプ

AWS Fargate（Fargate起動タイプ；サーバーレスのコンテナ）

・タスクで指定したCPU/メモリに応じて課金

実行したタスクのリソース使用量に対して料金が発生するので、不定期にタスクが実行されるシステムや、常にタスクが実行可能な状態でなければならないシステムなどに向いています。

・

EC2起動タイプ

・OSやミドルウェアのアップデート、スケーリングなどのサーバー管理を**ユーザーで実施**

・インスタンスタイプとインスタンスの起動時間に基づいて料金が決まる

#### EKS；Elastic Kubernetes Service

マネージド型のKubernetesサービス

・マネージド型。AWSがEKSのインフラストラクチャやクラスターの管理を行う

・実行中のアプリケーションの負荷状況に応じてリソース（ポッドやノード）を自動的にスケーリングする機能がある

・高可用性：マルチAZ構成をサポートしており、クラスターの各コンポーネントが冗長化されている

・統合： AWSのセキュリティ、監視、ロードバランシング、ストレージなどの機能を活用できます。

・モニタリングとトラブルシューティング：EKSはCloudWatchやX-RayなどのAWSのツールと統合しており、アプリケーションのモニタリングやトラブルシューティングが容易に行えます。

・Kubernetesのオープンソースツールやコンポーネントを使用することができる。

|  |  |
| --- | --- |
|  | 概要 |
| Cluster Autoscaler | ノードの数を増減させるオートスケーラー |
| HPA；Horizontal Pod Autoscaler | ポッドの数を増減させるオートスケーラー |
| Metrics Server | CPU使用率やメモリ使用量といったメトリクスに基づきポッドの数を調整する。HPAを動作させる為に必要 |

用語

|  |  |
| --- | --- |
| ノード | Kubernetesクラスタ内で実際に作業を実行する物理または仮想のサーバーのこと |
| ポッド | Kubernetesクラスタ内でアプリケーションを実行するための最小単位 |

### DB系サービス

#### 概要

比較表

|  |  |  |
| --- | --- | --- |
| サービス名 | 用途など | 概要 |
| Aurora | RDS内で使う | 高性能RDBMS  MySQL/PostgreSQL互換の独自RDB |
| DynamoDB | 蓄積専用 | Key-Value型NoSQL。容量無制限。  スキーマレス  3つのAZでテーブルを保持して可用性を向上 |
| ElastiCache  エラスティキャッシュ | キャッシュ | フルマネージ型のインメモリDB。  Redis、Memcachedを提供する |
| Redshift | BI用 | ペタバイト級を高速に扱う。ビッグデータ向き。  バックアップ（スナップショット）を異なるリージョンに作成することができる |
| RDS | 一般RDBのクラウド版 | RDB型。トランザクション処理。汎用型。  リードレプリカという参照専用のレプリカでパフォーマンスを向上 |

その他

|  |  |
| --- | --- |
| サービス名 | 概要 |
| Database Migration Service | データベースの移行サービス |

参考

|  |  |
| --- | --- |
| Athena | S3内のcsv, json にクエリでアクセス |
| ElastiCache | フルマネージドでインメモリデータベースRedis、Memcachedを提供するサービス |
| Redshift Spectrum | S3上のデータをRedshiftの外部テーブルとして参照できるようにした機能 |

資料

参考：<https://www.distant-view.co.jp/column/6585/>

#### Auroraと他のRDSとの比較

大きな違いは、『クラスタ』と『ストレージ』

比較表

|  |  |  |
| --- | --- | --- |
|  | Aurora | RDS |
| 互換DB | SQLServerやPostgreSQL互換 | OracleやMicrosoft SQL Server互換  （Postgre, SQLServerも対応） |
| 料金 | 割高 | 安め |
| ファイルオーバー | 対応  - 3つのAZに6つのボリュームを置いて冗長化 | マルチAZで対応  （スタンバイDBとして）  シングル構成の場合、 AZ内で冗長化 |
| サイズ | 10GB-128TIB | 20GIB-64TIB |
| スケーリング | 自動 | 手動 (オートスケーリングで自動化) |
| リードレプリカ　数 | 最大15 | 最大5 |
| クロスリージョン | 可能 (MySQL のみ) | 可能 |
| レプリケーション  同期 | 4個は同期、残りは非同期 | マルチ AZ DB インスタンス  　スタンバイDB: 同期 レプリカ : 非同期  マルチ AZ DBクラスター  　1台のスタンバイDBのみ同期 |
|  |  |  |
| その他特徴 | ・クラスターを構成 |  |

#### Aurora

##### 概要

MySQL/PostgreSQL互換のデータベースエンジン。（AWS RDSで用いる）

・高可用性

- RPO（目標復旧時点）を 1秒

- RTO（目標復旧時間）を1分未満

- レプリカインスタンスの自動フェイルオーバー

・自動スケーリング

・OS/データベースソフトウェアへのパッチ

・ストレージは、デフォルトで3つのAZに2つずつ（計6つ）のデータコピーが作成される

・クローンの作成が可能

・データベースインスタンスだけを必要に応じて増減できる

・更新可能なデータベースインスタンスはプライマリインスタンス1台のみ

・レプリカインスタンスは最大15台作成可能

・バックアップ

・最大64TBまでスケール可能な拡張性

・Aurora Auto Scalingによる負荷の自動調整

ユースケース

・MySQL/PostgreSQLからの移行

・ごく小規模なシステムを除き、迷うようならまずAurora（営業文句では？）

参考：<https://qiita.com/leomaro7/items/e48d9941dab5b5f2a718>

##### エンドポイント

|  |  |  |
| --- | --- | --- |
|  | 用途 | 接続先 |
| クラスターエンドポイント | 更新・参照・データベースの定義変更など、すべてのアクセス。 | プライマリインスタンス |
| リーダーエンドポイント | 参照のみを行う。  （別名：読み取りエンドポイント） | レプリカインスタンス |
| インスタンスエンドポイント | インスタンスへの直接アクセス | 各インスタンス |

補足：書き込みエンドポイント　というものは存在しない

クローンの作成

常に復元もとを参照しつづけ、動作の軽いレプリケーション。

ストレージを圧迫しない、メモリ上のデータベース（らしい）

使用目的

・検証目的で本番環境とまったく同じDBを作成

##### Aurora Global Database

Auroraを複数のリージョンにまたがって運用できるサービス

・データベースアクセスを世界中から高速に行える（レイテンシの向上）

・リージョン単位で発生した大規模な障害の災害対策（DR；Disaster Recovery）になる

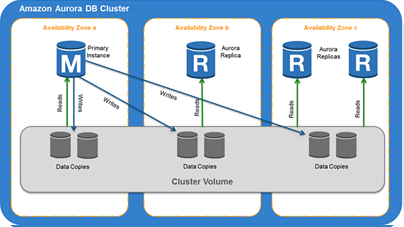
・プライマリリージョンのデータベースが停止した場合、セカンダリリージョンのデータベースを自動で昇格（フェイルオーバー）させ運用を継続

プライマリリージョンで障害が発生した場合

目標復旧時点（RPO）は1秒、目標復旧時間（RTO）は1分未満と定めている

##### クラスター

別名：Cluster

Multi Cluster

Amazon RDSの中でも**Auroraだけが持つ機能。**

複数マシンのリソースを束ねて､1つの仮想的なマシンを構成する

・複数の AZ にまたがる

・プライマリ DB インスタンスはクラスターボリュームに書き込み・読み込みを行う

補足

プライマリ DB インスタンス：書き込みと読み込みができる唯一のインスタンス

##### その他の機能

###### クローン

クローンは「Copy-on-Write」という技術で作成されます。Copy-on-Writeは、データの複製時にコピーしたと見せかけて、実際は複製元のデータを参照

###### Aurora Serverless

DBインスタンスの負荷状況に応じて、自動的にDBインスタンスの起動、停止、スケールアップ/スケールダウンを実施する機能

#### DynamoDB

##### 概要

Key-Value型の簡易データベース（NoSQL）。

料金が分かりにくい；1テーブルが≒70円/月くらいになるとか？

PynamoDBというPythonモジュールがあるらしい。

・単一リージョン内の3つのAZにテーブルを保持する事で高い可用性を持つ。

・ピーク時の秒間のリクエスト数 2000万件

・インデックスを持ったテーブルを構築することでスループットを向上させることができる

・別リージョンへのバックアップ（スナップショット）は非対応（RDSでは対応）

・SQLを用いた柔軟なクエリが書ける

ユースケース

・管理負担の削減。単純にアクセス負荷が読めないケースなど。

・ミリ秒単位のアクセスレイテンシが求められる

・データの拡張性が求められる

⇒ モバイル、ウェブ、ゲーム、広告技術、IoTのようなサービスのバックエンド

補足

ping-tではJSON形式を扱える、とあった。ドキュメント型ではないはずだが？？2023-09-12

##### バックアップ

DynamoDBには2つのバックアップ方法がある。

どちらも基本的にパフォーマンスへの影響は出ない

オンデマンドバックアップ

- ユーザーが任意のタイミングで作成するバックアップ。

- Management ConsoleやAPIを利用してテーブルの完全なバックアップを作成する。

ポイントインタイムリカバリ（PITR）（断続的バックアップ）

- 定期的かつ自動的な**差分バックアップ**を記録する。

- **35日まで**さかのぼる事ができる。

バックトラック

MySQL互換のみで利用可能

クラスターの巻き戻し機能。

最大24時間

##### データ更新のタイミング（一貫性）

3つのAZで運用する仕組みで、書き込みが開始されると2つのテーブルに書き込みが終わった時点で終了。残り１つは後からレプリケートされる。

DynamoDBでは以下の2つの読み込みモードをサポート。

結果整合性のある読み込み（デフォルト）

　・最新ではないデータを読み込む可能性がある。

（書き込みから少し時間をおくことで回避できる）。

強力な整合性のある読み込み

　・強力な整合性のある読み込み（必ず最新のデータを取り出すことができる。）

・読み込み時のコストが2倍になる、

・レイテンシが高くなる可能性がある、などの条件がつく。

##### キャパシティユニット

テーブルに書き込み、読み込みをする際のスループットの単位

言い換えると1秒間にどれだけ読み込み、書き込みを行うかを予約する設定

また、これを基にパーティションの数やサイズが自動的に算出される

|  |  |
| --- | --- |
| WCU  Write Capacity Unit | 最大1KBのデータを1秒間に1回の書き込み |
| RCU  Read Capacity Unit | 最大4KBのデータを |

> テーブルに設定する？

自動的にスケール（増減）する設定が可能

リザーブドキャパシティ

大規模に利用する場合に用い、100ユニット単位で購入する。

##### DAX；DynamoDB Accelerator

インメモリキャッシュクラスタ。DynamoDB専用のElastiCacheみたいな感じ



・RCUを削減し、コストを抑えることができる

・ミリ秒（千分の一秒）だったレスポンスをマイクロ秒（百万分の一秒）レベルのパフォーマンスにまで向上させることができる。

・

ユースケース

・レスポンス時間を最短にしなければ行けない場合

・読み込みが多いアプリケーション

・大規模なデータから何度も読み取りを行うアプリケーション

##### DynamoDB Streams

テーブルに対して行われた直近24時間の変更をログに保存する機能

ログには、アプリケーションがリアルタイムにアクセスできるため、変更内容に応じて処理を組み込むことができます。例えば、ログを監視し問題のある処理があった場合はアラートを上げさせたり、プロフィール画像を更新（追加）したらフレンドに通知させるというような運用が可能です。

##### その他の機能など

###### モード

オンデマンドモード 書き込み・読み込みのリクエスト単位で利用料金がかかかる

プロビジョニングモード 1秒間に行う書き込み・読み込みの量で利用料金が異なる

###### 自動スケール

データ容量が増えた場合でも、DynamoDBは自動的にスケールアウト（拡張）されます。データベース用ノードの追加やディスクを増やすといった作業がないため、ダウンタイム（サービスを停止する時間）がなくサービスを提供し続けることができます。

###### S3へのエクスポート

既存のテーブルデータを任意のS3バケットに直接エクスポートする機能

・実行するためには、対象のテーブルのポイントインタイムリカバリを有効にする必要がある

・対象のテーブルに設定されたRCUのコストに影響を与えずに実行できる

・エクスポート先のS3バケットは事前に作成しておく必要がある

補足

この機能を実行中も、ユーザは対象のテーブルに対して読み取り・書き込みの操作が可能

###### グローバルテーブル

DynamoDBテーブルを複数のリージョンにまたがって運用できるサービス。

複数のリージョンにDynamoDBテーブルが自動的にレプリケートされ、ユーザーは地理的に近いリージョンのDynamoDBテーブルへ高速な読み込みと書き込みが可能になる。データのレプリケーションは通常1秒以内に完了し、リージョン間のデータ冗長化によって高可用性が確保されます。

##### 詳細

###### Action（ポリシー）

"dynamodb:DeleteItem",

"dynamodb:GetItem",

"dynamodb:PutItem",

"dynamodb:Query",

"dynamodb:Scan",

"dynamodb:UpdateItem"

・Lambdaなどで実行する際、テーブル名は環境変数経由で渡し、テーブル名をハードコーディングしない。

#### ElastiCache

##### 概要

インメモリデータベースRedis、Memcachedを提供するサービス

|  |  |
| --- | --- |
| エンジン | 機能 |
| Redis | 高機能  複雑なデータ構造、データ操作が可能  バックアップ、リストア、暗号化などが可能 |
| Memcached | シンプルなKey-Value型  高パフォーマンス  永続保持は無い。シンプルなキャッシュ。  **暗号化は非対応** |

・AWS RDSと連携して、高頻度でアクセスされるデータをElastiCacheで処理する連携が可能。

・インメモリ（すべてのデータをメモリ上に保持）

・フルマネージド

- ソフトウェアへのパッチ適用などが自動で行われる

##### 共通

レプリカノード

参照専用のノード

##### Redis

補足：Redisは一般的に利用されているOSSのNoSQLデータベース。

##### Memcached

###### 自動フェイルオーバー機能

プライマリノードに障害が発生した場合は自動的にレプリカノードがプライマリとして機能する

#### Redshift

##### 概要

BIに向く。列型指向(columnar)のDB。

・S3上のデータファイルのロード（読み込み）や、S3へのエクスポート（保存）に対応

・8時間毎、12時間毎などでスケジュールバックアップ（スナップショット作成）が可能

・Leaderノード一つと複数のComputeノードから成るクラスタ構成

ユースケース

・

##### 構成

・大量のリクエストを分散させ並列で処理するクラスタ構成



・Leaderノード

各クラスタに一台だけ存在する司令塔（Leader）となるノード。

アプリケーションなどからのリクエストを受け付けて各Computeノードへ割り振り、またその結果を取りまとめてアプリケーションへ返す役割を果たします。

・Computeノード

Leaderノードからの命令を処理するノードです。

一つのComputeノードはCPU、メモリ、ストレージを搭載しており、Computeノードを増やすことでパフォーマンスを向上させることができます。

・ノードスライス

Computeノードのリソースを更に細分化した単位をノードスライスと呼びます。

※図中では各ノード2スライスですが、スライスの数はComputeノードのインスタンスタイプによって異なります。

##### クロスリージョンスナップショット

異なるリージョンに作成するスナップショット

異なるリージョンでRedshiftクラスタを再構築したい場合にも利用可能

##### 同時実行スケーリング

メインのクラスタに処理待ちが発生すると、バックグラウンドで独立したクラスタを順次追加して並行処理を行う。

##### Spectrum

RedshiftのS3上のデータを外部テーブルとして参照する

S3上のデータを効率的に参照するための仕組み

補足：旧来のRedshiftでは、データが蓄積され大容量化するとS3にあるデータをRedshiftへロードする際に時間がかかっていた

#### RDS

##### 概要

Relational Database Service

・高頻度でアクセスされるデータをElastiCache上に保持する連携が可能

・自動スケーリング対応

アクセス負荷が高い時間帯にはスケールアップ（拡張）、逆にアクセスが少ないときにはスケールダウン（縮小）を自動的に行う事もできる

・暗号化を行う場合はデータベースの作成時に指定。（作成後は不可）

##### エンジン

Aurora

MySQL

MariaDB

PostgreSQL

Oracle

SQL Server

##### デプロイオプション

1 シングルDB インスタンス

2 マルチ AZ DBインスタンス

3 マルチ AZ DBクラスター (2022/3 新機能)

##### 暗号化機能

暗号化機能はデータベースエンジンに依存

・データベースの作成時に暗号化オプションを有効にする。（途中変更はできない）

⇒ 暗号化されていないデータベースインスタンスを暗号化したい場合は、対象のインスタンスのスナップショットを作成し、スナップショットをコピーする際に暗号化を有効にする

⇒ 暗号化されているデータベースはスナップショットも暗号化されている

##### インスタンスストレージ

|  |  |
| --- | --- |
|  | 概要 |
| 汎用 SSD | 中規模のDBに適した効率の高いストレージで、開発やテスト環境向き。 |
| プロビジョンド IOPS | I/O負荷の高いワークロードに最適な高パフォーマンスストレージ。 |
| マグネティック | 下位互換性のために提供される古いストレージタイプ。 |

##### インスタンスクラス

EC2における"インスタンスタイプ"と同等

|  |  |  |
| --- | --- | --- |
| 名称 | クラス |  |
| 標準クラス | m |  |
| メモリ最適化クラス | r, x |  |
| バースト可能クラス | t |  |

公式（インスタンスクラス）

[https://docs.aws.amazon.com/ja\_jp/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Summary](https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html" \l "Concepts.DBInstanceClass.Summary)

公式（DBごとのインスタンスクラス対応）

[https://docs.aws.amazon.com/ja\_jp/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Summary](https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html" \l "Concepts.DBInstanceClass.Summary)

##### RR；Read Replica

概要

参照専用のデータベースとして動作するレプリカ。参照時のアクセスを分散させる。

・通常は最大5台、Amazon Auroraの場合は最大15台。

・フェイルオーバーには使う事ができない

・異なるAZにも配置できる

- 災害対策

- ユーザーとの距離を近づける

（パフォーマンス向上の目的には向かない）

ユースケース

・

・

作成する際に選択できるオプション

・インスタンスクラス

・ストレージタイプ

・マルチAZ配置

作成後に設定

・いちRRあたりのクエリ発行数の上限（設定できるらしい？）

##### スナップショット

RDSデータベースを所有しているAWSアカウントとは別のAWSアカウントと、データベースの手動スナップショットを共有できる。

ユースケース

・AWSアカウントを別に持つ部署でデータベースのコピーを作成

・新たに作成するAWSアカウントへデータベースを移行

・スナップショットが暗号化されている場合は、共有先のアカウントへKMS暗号化キーの使用を許可する。

・自動バックアップのスナップショットは他アカウントと共有できない

##### パラメータ

DBパラメータグループ

DBインスタンス関連の設定を行う？

公式：<https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/USER_WorkingWithParamGroups.html>

オプショングループ

データベースエンジンの設定を行う

タイムゾーン

最大接続数

監査ログ

公式：<https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/USER_WorkingWithOptionGroups.html>

##### 具体例

まずはRead Replicaによる参照負荷の分散

##### その他の機能など

###### 削除保護

設定されたデータベースを削除できなくする

### 管理／権限系

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| IAM AccessAnalyzer |  |
| Trusted Advisor | 利用者のAWS環境とベストプラクティスと照らし合わしてアドバイスを提供する |
|  |  |

#### IAM AccessAnalyzer

最小権限の許可に向けて

#### Trusted Advisor

##### 概要

利用者のAWS環境をベストプラクティスと照会することにより、推奨されるアクションのアドバイスを行うサービス。

サービスが正常に稼働できているかはもちろん、過剰にサービスを使用しコスト超過に陥っていないかや、セキュリティ設定が適切に設定できているかなどを確認。

・アクセス元の制限がないセキュリティグループが定義されていないかチェックできる

2023-09-26日本語の意味が分からないんだが



##### チェック観点

|  |  |  |
| --- | --- | --- |
|  | 概要 | 具体例 |
| コスト最適化 |  | EC2リザーブドインスタンスの最適化 ... リザーブドインスタンス（予約購入したインスタンス）の利用状況をもとに最適な購入数を算出し、コストの節約が可能か |
| パフォーマンス |  | EC2インスタンスの使用率 ... 直近14日で1日のCPU使用率が90%以上の日が頻出していないか |
| セキュリティ |  | 開かれたポート ... アクセス元の制限がない（0.0.0.0/0）セキュリティグループが定義されていないか（ポート番号に応じて警告を表示） |
| フォールトトレランス |  | ロードバランサ（ELB）の最適化 ... ELB配下のインスタンスが複数AZで稼働しているか |
| サービスの制限 |  | 各サービスに設定された制限値と利用状況を照会し、使用率がひっ迫したり上限に達していないか |

誤解されやすいもの

請求額のアラートを設定 ⇒ Amazon CloudWatch

EC2インスタンスの状況（故障など） ⇒ EC2

補足

フォールトトレランス（fault tolerance）

システムに障害が発生した場合にも、系統を切り替えるなどして正常に稼働させ続けること

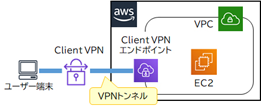
### セキュリティ／監視系

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| Certificate Manager | SSL/TLS証明書を管理する |
| Client VPN |  |
| CloudHSM | 専用のハードウェアデバイスを用いて暗号化鍵を生成・管理 |
| CloudTrail | AWSサービスに対するアクションの記録・保存 |
| CloudWatch | AWSリソース、オンプレの監視とそのイベントを扱う。 |
| DX;Direct Connect | ユーザー環境から専用回線でAWSへセキュアに接続 |
| Firewall Manager | ファイアウォール |
| GuardDuty | AWS環境の脅威を検出する |
| KMS;Key Management Service | 暗号化に使用するキーを作成・管理する |
| Shield | WebアプリへのDDoS攻撃に対する保護 |
| Site-to-Site VPN | オンプレとAWSのゲートウェイをインターネットVPNでつなぐ　比：Direct Connect |
| VPC Flow Logs | VPC内のENI間で行き来するIPトラフィックをキャプチャ |
| WAF;Web Application Firewall | WAF（Webアプリに特化したファイアウォール） |
|  |  |

#### Client VPN

自宅などの環境からVPC(Virtual Private Cloud)へセキュアにアクセスする。



・OpenVPNを使用

比較

Site-to-Site VPN ルーターを用いてネットワーク全体の通信を対象にする

Direct Connect 専用回線を引く

Site-to-Site VPNとの比較

|  |  |  |
| --- | --- | --- |
|  | AWS Client VPN | AWS Site-to-Site VPN |
| 概要 | 利用者のクライアント端末とVPC | 利用者のオンプレミスネットワークとVPC |
| VPC端末 | Clinet VPC エンドポイント | VGW |
| 暗号化 | クライアント端末にインストールしたソフトウェア | ルーター |
| 通信方式 | Open VPNベース | IPSec VPN |
| 備考 | ・クライアント側機器に暗号化ソフトをインストールする | ・オンプレにルータを設置する。 |

#### Site-to-Site VPN

オンプレミスのルーターとVGWを、インターネットVPNで接続するサービス

補足

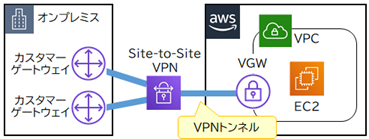
VGW Virtual Private Gateway（仮想プライベートゲートウェイ）

比較

|  |  |
| --- | --- |
| Direct Connect | 専用回線で構成するVPN |

用語

|  |  |
| --- | --- |
| 用語 | 概要 |
| カスタマーゲートウェイ | AWS VPN／Direct Connectで使うオンプレ側のエンドポイント |
| VGW；Virtual Private Gateway | パブリックサブネットに配置され、インターネットゲートウェイとVPC間の通信を制御 |
| VPNトンネル | AWSのVPN接続で使用される仮想的な専用ネットワークトンネル |
|  |  |

イメージ図

補足

|  |  |
| --- | --- |
| 用語 | 概要 |
| Direct Connectロケーション | Direct Connectで使用するAWS側のエンドポイント |

#### CloudHSM

HSM；Hardware Security Module

##### 概要

専用のハードウェアデバイスを用いて暗号化鍵を生成・管理するサービス。

世界的な暗号化ハードウェアの規格（FIPS 140-2 のレベル 3）で認証済み

補足：専用ハードはAWS内にあり、ハードが貸し出される訳ではない

特徴

・ユーザーがキーを作成・管理する

・HSM は VPC 内にあり、他の AWS ネットワークから分離されている

・AWS はHSMアプライアンスを管理　？？

・AWSはキーへのアクセス権は持たない

・アプリケーションのパフォーマンスが向上する (AWS ワークロードと近接しているため)

・複数のAZで利用できる、不正使用防止策が施されたハードウェアでの安全なキーストレージ

比較

|  |  |  |
| --- | --- | --- |
|  | KMS | HSM |
| 鍵の生成 | AWSが生成 | ユーザーが行う（専用VPC内で行う） |
| 鍵の管理 | AWSが管理 | ユーザーが行う |
| その他 |  | ・利用料金が高価 |

ユースケース

・法令や規制等の要件で、認定を受けたハードウェアで暗号化鍵を管理する必要がある場合

#### CloudTrail

##### 概要

AWSアカウントがAWSサービスに対して「いつ」「誰によって」「どのような操作が行われたのか」を記録・保存する**システム監査者向きサービス**。

・利用登録なしに過去90日間のログを参照できる（ただしS3には料金がかかる）

・S3へ保存するログの内容はJSON形式で出力され、gz形式で圧縮される。AWS KMSを利用して暗号化することも可能

・複数のAWSアカウントを管理している場合、それぞれのCloudTrailのログを1つのS3バケットで管理することができる。

証跡情報の作成時に「ストレージの場所」を既存のS3バケットを指定

・CloudWatch Logsとの連携が可能

・操作ログの詳細情報の参照

主な機能

　イベント履歴

　Lake

　証跡 S3に送信される

公式：https://aws.amazon.com/jp/cloudtrail/features/

記録されるもの

|  |  |  |
| --- | --- | --- |
|  | 具体例 | 備考 |
| 管理イベント | ユーザーのログイン  EC2インスタンスの作成  KMSのイベント | デフォルトで有効 |
| データイベント | S3上のオブジェクト（データ）の操作  Lambda関数の実行  （それぞれLogの記録；念のため） | 「証跡情報」の登録が必要 |
| Log Insightsイベント | 通常と異なる操作（書き込みAPIの呼び出し） | 「証跡情報」の登録が必要 |

出来ない事

・OSのシステムログの記録

・CloudTrailとの連携（代わりにCloudWatch Logsを使う）

##### チュートリアル

CloudTrail > 証跡の作成 > 証跡の表示名を入力 > 証跡の作成

##### 証跡情報

証跡情報を登録することによって実現できるもの

・保存する先のS3バケットの設定

・90日以上の操作ログの保存

・ログを記録するイベントの種類の設定

・データイベント、Log Insightsイベントの記録を有効にする

- データイベント S3上のオブジェクト（データ）の操作、Lambda関数（※）の実行

- Log Insights 通常と異なる操作（書き込みAPIの呼び出し）

記録されるもの

補足

|  |  |
| --- | --- |
| 操作ログの詳細情報の参照 | デフォルトで有効 |
| AWS Lambdaとの連携 | CloudTrailでは達成できない（Lambdaの呼び出しログは記録できる） |

##### 具体例

ログインに3回以上失敗した場合には管理用のメーリングリストへメール通知が行われるようにする

⇒ AWS CloudTrailとAmazon CloudWatch Logsを連携しログインイベントのフィルタを設定する

・キーワードにログインイベント「ConsoleLogin」を指定する

#### CloudWatch

##### 概要

**リソースの状態に応じて**アクションをとることができるサービス。

・AWSやオンプレのシステムを監視

・しきい値（境界値）を超える・下回るなどした場合に管理者に通報

・ログ、メトリクス、イベントの形式で提供

|  |  |
| --- | --- |
| サービス | 概要 |
| CloudWatch | アラームとアクションの定義を行う**総合エージェント**  AWSサービスのリソース変化やスケジュールに応じてアクションをとる  単体でも、通知アクションを取る事もできる（？） |
| CloudWatch Alarm | **メトリクス**（リソース情報）に応じて**アクションを実行** |
| CloudWatch Events  (EventBridge) | イベント（AWS**リソースの変化**）に応じて**アクションを実行**  API呼び出しなど |
| CloudWatch Logs | ログの収集、一元管理  対象：AWSサービス、EC2インスタンスのOS・アプリケーションのログ  ログ内容に応じたメール通知も可能 |
| CloudWatch Metrics | AWSリソースのメトリクスを収集・管理 |
| CloudWatch Logs Insights | **クエリの実行**による柔軟なログ解析を行う |
|  |  |

・請求額によるアラートの設定が可能

・CPU使用率や空きストレージ容量など監視する項目を設定する

比較

|  |  |
| --- | --- |
| CloudTrail | 管理コンソールへのログインや各種AWSサービスで行ったアクションを記録 |

補足　例えばEC2のログをCloudWachで使う場合は以下の２つの方法がある

・アプリにaws cliかCloudWath API組込むなどして能動的にCloudWatchにログをアップロード

・EC2にCloudWatch Logsエージェントをインストールする

##### アラーム

・1つのアラームには1つのメトリクスしか指定できない

・複数のアラームを指定して複合アラームを作成する事は可能

操作：複合アラームの作成 例：NetworkIN-alarm＋CPU-alarm

##### アクション比較

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| サービス | トリガ | メール通知 | Lambda実行 |  | aws SQS | その他／備考 |
| CloudWatch | メトリクス |  |  |  |  | ・アラームアクション  ・状態にはOK、ALARM、INSUFFICIENT\_DATAがある |
| CloudWatch Alarm | メトリクスがしきい値を超えた時 | ○ |  |  |  | ・AWS SNS  ・一般的なSNS  ・AutoScaling連携 |
| CloudWatch Events  (EventBridge) | AWSリソース（インスタンス）の状態変化 | ○ | ○ |  | ○ | ・Kinesisストリーム |
| CloudWatch Logs | ログ内容 | L | ○ |  | ○ |  |

補足

・Lambdaに対応していると以下の事に対応できる

- インスタンスの動作 [復旧、再起動、停止、終了]

- WebAPIのキック

- メール通知（上の表では「L」と表記）

・

##### メトリクス

###### 概要

CPU使用率などシステムのパフォーマンスに関するデータを加工し、評価や分析に適した形式に加工したもの

標準メトリクス 標準で用意されているメトリクス

カスタムメトリクス 独自で定義することのできるリソース情報の

|  |  |
| --- | --- |
|  |  |
| DynamoDB |  |
| EBS |  |
| EC2 |  |
| ElastiCache |  |
| RDS |  |
| 使用 | ？ |

###### EC2

標準メトリクス

CPU使用率 (CPUUtilization)

ディスクの読み取り ・ 書き込み量　(DiskReadBytes /DiskWriteBytes)

ネットワーク使用率　(NetworkIn/NetworkOut)

カスタムメトリクス

メモリ使用量

ディスクの空き容量・使用状況

プロセス情報

カスタムメトリクスを設定するには、EC2インスタンスへ「CloudWatchエージェント（CWエージェント）」をインストールし、AWS CLIコマンドやAPIを用いて監視項目をCloudWatchへ送信する、といった能動的な設定が必要。

###### RDS

標準メトリクス

|  |  |
| --- | --- |
| FreeStorageSpace | ストレージの空き容量 |

##### ログのカテゴリ

\vended Logs

カスタムログ

##### CloudWatch Alarm

収集したAWSリソース情報に応じてメール通報などのアクションを行う機能

##### CloudWatch Logs

システムのログデータを収集、監視、管理する為のシステム。

Lambda関数、VPCフローログなど、さまざまなAWSリソースからログデータを収集する。

・ログのフィルタリング結果を条件に管理者にメール通報できる

・ログの内容をフィルタリングしてKinesis，Lambda，OpenSearch Serviceへ転送できる

・EC2インスタンス上のアプリケーションやOSのログを収集するには、対象のEC2インスタンスへ「CloudWatchエージェント」をインストールする必要がある。

比較

|  |  |
| --- | --- |
| Athena |  |
| CloudWatch Log | アプリケーションのログを収集 |
| CloudTrail | 管理コンソールへのログインや各種AWSサービスで行ったアクションを記録 |

##### CloudWatch Logs Insights

クエリの実行による柔軟なログ解析を行う機能

特定のアプリケーションが一定期間に出力した"Error"を含むメッセージをカウントする」のようなクエリを実行することができる

##### CouldWatch Events(EventBridge)

トリガ AWSリソース（インスタンス）の状態変化

アクション

・Lambda関数の実行が可能

補足

Amazon SNSを用いても同様の事ができる（らしい）

##### CloudWatch Metrics

AWSリソースのパフォーマンスに関するデータを確認できる

Metrics Insight

CloudWatch Metricsの「クエリ」タブで設定できる

##### CloudWatch Subscription Filter

リアルタイムにKinesis Data Firehoseなどに連携

##### Logs Insights

ログの解析を行う。

##### CloudWatchエージェント

概要

EC2インスタンスやオンプレミスのサーバーからCloudWatchにデータを送信する為のエージェント

補足　例えばEC2のログをCloudWachで使う場合は以下の２つの方法がある

・アプリにaws cliかCloudWath API組込むなどして能動的にCloudWatchにログをアップロード

・EC2にCloudWatch Logsエージェントをインストールする

##### 具体例

EC2インスタンスを毎朝9:00に起動・22:00に停止する

⇒ Amazon EventBridgeを使用し、決まった時間に起動と停止を行うLambda関数を呼び出す

#### Direct Connect

##### 概要

専用回線を用いたVPNで、オンプレミスなどのユーザー環境からAWSへ接続する。

・パブリック接続とプライベート接続がある。（後述）

・専用回線を敷設して使用する為、使用開始までに時間がかかる。

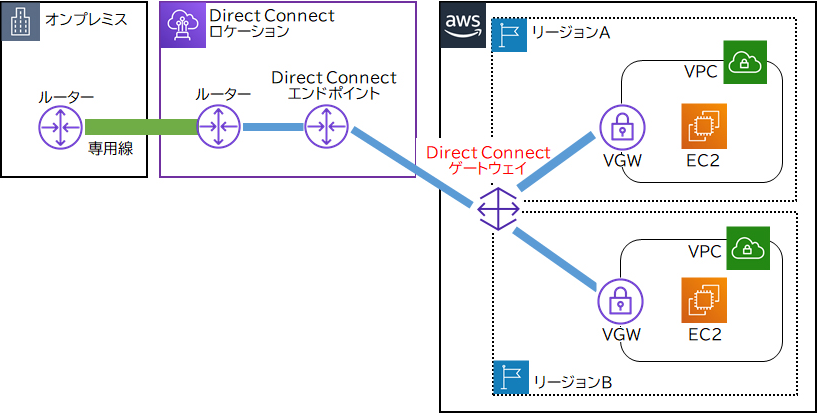
・Direct Connectロケーションは各リージョンに複数用意されており、ユーザーは接続するロケーションを選択できる。

比較（セキュアな回線）

AWS VPN インターネット回線を用いたVPN

##### Direct Connectゲートウェイ

Direct ConnectエンドポイントとVGWの間に配置され、複数のVPCに接続する為に用いるゲートウェイ。（VGWは１つのVPCにしか接続できない）

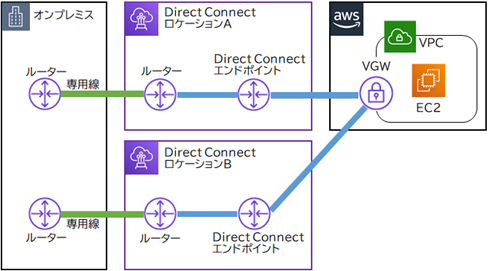


・世界中の各リージョンにある複数のVPCへ接続できる

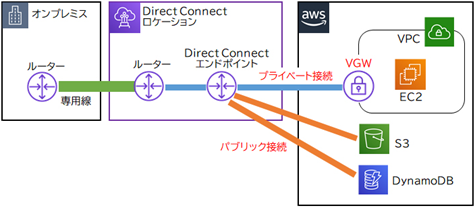
##### 具体例

高可用性の構成

複数のDirect Connectエンドポイントを用意する



プライベート／パブリック接続

- プライベート接続

VGWを通じてVPCのみに接続

- パブリック接続

VPC外にあるAWSリソースへ（S3やDynamoDBなどの）へ接続

#### KMS；Key Management Service

##### 概要

暗号化に使用するキーを作成・管理する

キーステータス

Enabled

Disabled

PendingDeletion

暗号化ステータス

アクション

|  |  |  |
| --- | --- | --- |
| Action | 概要 | 備考 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

・Lambda関数のIAMロールへ鍵の「kms:\*」アクション

鍵の作成者のみに設定するアクション 2023-09-26時点では理解できていない

比較

|  |  |
| --- | --- |
| CloudHSM | FIPS 140-2 のレベル 3で認証されたハードウェア上で暗号化鍵を管理 |

##### KMSキーの種類

AWSマネージド型

AWSによって作成・管理されるもの

ユーザーは削除できない

カスタマーマネージド型

ユーザーが作成・削除、および管理を行う

鍵の自動ローテーション（定期的な更新）をサポート

##### キーポリシー

カスタマーマネージド型のKMSキーへ適用するリソースベースのポリシー。

キーユーザー

鍵を使用可能なIAMユーザーまたはIAMロール

クロスアカウント

別のAWSアカウントで鍵を使用する場合に使用する

具体例

##### エンベロープ暗号化

「KMSキー」「データキー」と呼ばれる2種類の鍵を使用してデータの暗号化および復号を行う方式の事

KMSキー データキーを暗号化する際に使用。１つだけ作成。

データキー データを暗号化する際に使用。暗号化する対象の数だけ作成する。

#### Shield

##### 概要

DDoS攻撃に特化した保護サービス。

##### AWS Shield Standard

ネットワーク層およびトランスポート層への一般的なDDoS攻撃からAWSリソースを保護。

デフォルトで有効

##### AWS Shield Advanced

EC2インスタンス、ELB、CloudFrontなどを標的とした高度な保護サービス

・高度化された大規模な攻撃からの保護

・DDoS攻撃発生時のモニタリングやレポート

・AWSのDDoS対応チームによるサポート

・攻撃によって増加したAWS利用料金の補填がある

#### WAF;Web Application Firewall

##### 概要

Webアプリケーションに対する攻撃から保護するサービス

特徴

・国別にアクセスを制限できる

・「Web ACL」というアクセスコントロールリストで、IPアドレス、HTTPヘッダー、HTTP本文、URI文字列などに対してフィルタリングの条件を設定できる

割当先

・CloudFront

・ALB（NLB、CLBは非対応）

・API Gateway

対応

クロスサイトスクリプティング

SQLインジェクション

など

比較

AWS Shield DDos攻撃からの保護に特化

##### Web ACL

IPアドレス、HTTPヘッダー、HTTP本文、URI文字列などに対してフィルタリングの条件を設定をしたアクセスコントロールリスト

### データ関連

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| EMR；旧Elastic MapReduce | 大量データの**バッチ処理**を迅速に効率よく処理。Hadoopに対応。 |
| Data PipeLine | データの移動、変換を自動化 |
| Glue | データETL（抽出／変換／ロード）サービス |
| Kinesis | **ストリームデータ**を扱う。DataStreams、DataFireHouse、DataAnalyticsがある |
| IoT Core |  |

|  |  |  |
| --- | --- | --- |
|  | Glue | Data Pipeline |
| 動作 | サーバーレス | EC2インスタンスが起動 |
|  |  |  |

#### Data PipeLine

データの移動や変換を自動化する。

・処理の実行時にEC2インスタンスが起動する

具体例

S3からDynamoDB

RDSからRedshift

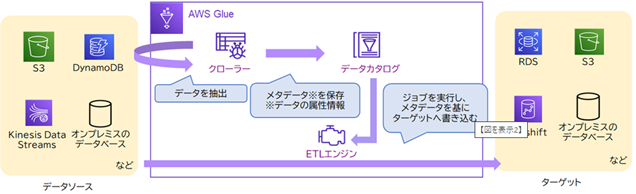
#### Glue

##### 概要

フルマネージドのサーバーレスETLサービス

複数のデータソースからデータを抽出し、変換・統合したデータをターゲットへ格納する

データソースからクローラーがデータを抽出し、データカタログを作成。



|  |  |  |
| --- | --- | --- |
| 項目 | 内容 | 備考 |
| データソース | S3、Dynamo、Kinesis Data Streams、オンプレのDB |  |
| データカタログ | Apache Spark、Scala  Athena、EMR |  |
| 出力 | S3、RDS、Redshift |  |

・データ変換処理のベースにApache SparkやScalaを用いる

##### Data Catalog

別名：データカタログ

抽出したデータのメタデータ（データの属性情報）が保存されたもの。データそのものではなく、データソースのどのデータをどのような方法で抽出するかなどの情報が記載されている。

参照元

Athena、EMR、S3、DynamoDBなど

保存先

S3、RDS、Redshift

ジョブ

データの変換およびターゲットへ書き出す処理の事

##### Crawlers

別名：クローラー

データソースからデータを抽出し、「データカタログ」を作成する。

1回の実行で複数のデータストア(AWS S3 などの対象データを格納している場所など)から収集したデータをData Catalogに作成または更新します。

Crawler がもつ主な３つの機能

１. フォーマットやスキーマ、データの性質を定義するためにデータを分類する機能

２. データをテーブルまたはパーティションにグループ化する機能

３. メタデータをデータカタログに書き込む機能

##### Data Integration and ETL

まさしくETL（Extract，Transform，Load）を行う

##### Jobs

抽出、変換、ロード (ETL)の処理のコードをスクリプトに記述して実行する。

#### Kinesis

##### 概要

ストリームデータを扱う

補足１：ストリームデータとは何？を補足する資料　一般的なデータの処理方式

|  |  |  |
| --- | --- | --- |
| 処理方式 | 概要 | 処理頻度 |
| ストリーム処理 | イベント発生の度にリアルタイムで処理を行う | 随時 |
| バッチ処理 | 例えば昨日のデータを夜中の内にまとめて処理する、といった具合 | 日次、週次など |

補足２

送信者、受信者を表す単語について：

|  |  |  |
| --- | --- | --- |
| 送信側 | 受信側 | 備考 |
| Producer | Consumer | ビッグデータやストリーミングデータ処理の分野で使われている印象 |
| Publisher | Subscriber | 軽量のメッセージを扱うアプリで好まれる |

特徴

・巨大なデータを処理するため、アプリケーションの実行をハードウェアのクラスター上で行う

・オープンソースのソフトウェアフレームワーク。

・EMR；Elastic MapReduceのコンピューティングとストレージは分離されているため、スケーリングができます

機能

|  |  |
| --- | --- |
| サービス |  |
| DataStreams | ストリーミングデータを収集する |
| DataFireHouse | ストリーミングデータをS3などのサービスへ配信 |
| DataAnalytics | ストリーミングデータをリアルタイムをSQLなどで処理、分析して出力 |
| Video Streams | ビデオストリームを取り込み、解析や加工を行う |

比較

|  |  |  |
| --- | --- | --- |
|  | 得意な処理 | 概要 |
| Kinesis | ストリーム処理 | データを受け取り、ストリームとして処理する |
| EMR | バッチ処理 | 異種のデータソースを収集し、エクスポートや分析などの処理を行う |

補足

OSSでのストリーム処理ソフトはApache Flink、Apache Sparkなどがある

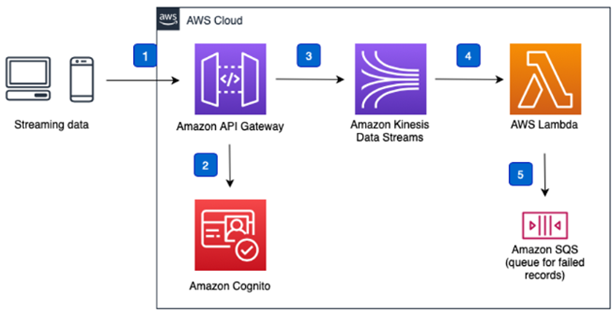
##### DataAnalytics

比較

Quick Sight 視覚的なデータ分析（BI）

##### 具体例

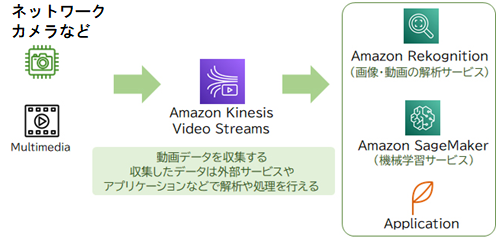
###### 基本構成



情報元：<https://aws.amazon.com/jp/solutions/implementations/streaming-data-solution-for-amazon-kinesis/>

###### 動画解析や加工

カメラやビデオなどの動画（ビデオストリーム）を取り込み、再生や解析、加工などを行う



#### EMR；Elastic MapReduce

##### 概要

大量のデータを迅速に効率よく処理するためのサービス。

ビッグデータフレームワーク(Hadoop、Spark)の構築や運用、また分散アプリケーションの実行を行う

補足

MapReduce Googleが開発した、ビッグデータを分散処理するフレームワーク

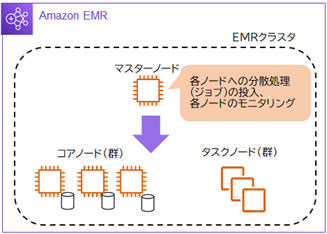
比較

|  |  |  |
| --- | --- | --- |
|  | 得意な処理 | 概要 |
| Kinesis | ストリーム処理 | データを受け取り、ストリームとして処理する |
| EMR | バッチ処理 | 異種のデータソースを収集し、エクスポートや分析などの処理を行う |

用語

|  |  |
| --- | --- |
| EMRクラスター | ビッグデータフレームワークやツールが事前にセットアップされたEC2インスタンス |

##### ノード

・マスターノード

　クラスター全体を管理するノード。コアノードおよびタスクノードへ処理を投入したり分散処理の調整を行う。

・コアノード

　演算処理を実行するノード。データを保存するファイルシステムHDFS（Hadoop Distributed File System）を持つ。

・タスクノード（オプション）

　コアノードと同様に演算処理を実行する。ファイルシステムを持たない、演算処理専用のノード

特徴

・スケーリングは、EMRクラスター内でコアノードまたはタスクノードを増減することで調整する。

##### ファイルシステム

HDFSとEMRFSを利用できる。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ノード | 概要 | データ保持 | 備考 |
| HDFS | マスター  コア | 分散処理ソフトウェアHadoopのファイルシステム | EMRクラスター終了で失われる | デフォルトの分散ファイルシステム |
| EMRFS | すべてのノード | EMRクラスターからS3をファイルシステムとして利用 | 永続的に保持できる | S3の機能が使える |

HDFS；Hadoop Distributed File System

・

EMRFS；EMR File System

・S3が持つ機能（データの暗号化やデータ読み込み時の強い一貫性のサポートなど）も有効。

### AWSサービス用ツール

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| Pricing Calculator |  |
| Cost Explorer |  |
| SSM；Systems Manager | 複数のAWSリソースをグループ化し運用タスクの自動化などを行う |

#### Pricing Calculator

#### Cost Explorer

AWSのコスト管理を行いグラフで可視化

#### SSM；Systems Manager

##### 概要

複数のAWSリソースをグループ化し、グループ内リソースの運用データの一元化や運用タスクの自動化などができる運用管理サービス

自由にリソースをグループ化し、データを見ることができる

|  |  |
| --- | --- |
|  |  |
| Patch Manager | OSパッチの適用を自動化する |
| Parameter Store | データベースのログイン時の認証情報などのパラメータを管理する |
| Run Command | EC2、オンプレのサーバーに対してコマンドやスクリプトを安全に実行 |
| Session Manager | Management ConsoleからEC2インスタンスへセキュアにログイン |
|  |  |

##### Parameter Store

##### Patch Manager

OSパッチの適用を自動化する

具体例

・複数のEC2インスタンスのOSパッチ適用を自動化する

##### Run Command

##### Session Manager

ブラウザからEC2インスタンスへログインできる

補足：Management ConsoleではEC2から設定するらしい？2023-10-23

・Session Manager経由のログインでは、キーペア（SSHキー）は必要ない

### ネットワーク最適化系

|  |  |
| --- | --- |
| サービス名 | 概要 |
| CloudFront | コンテンツ配信サービス。オリジンサーバーへのリクエストを受けるリバースプロキシ。 |
| Global Accelerator | 複数のリージョン間のリソースへの負荷分散 |
| SQS；Simple Queue Service | キューイングサービス（第三者経由でのアプリ間通信） |
| SNS；Simple Notification Service | プッシュ型のメッセージングサービス |

### 機械学習系

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| Comprehend | 自然言語処理サービス |
| Comprehend Medical | テキストデータから医療情報を抽出 |
| Textract | OCR（光学的文字認識）サービス |
| Transcribe | 音声をテキストに変換 |
| Translate | 自然言語処理サービス |

#### Comprehend

エンティティ認識、感情分析、キーフレーズ抽出

### 分類グループなしのサービス群

#### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
|  |  |
| API Gateway | EC2などAWSサービスと連携するAPIを作成。REST APIなど。 |
| APP |  |
| Athena | S3のcsvやjsonに対するデータクエリを提供。実行環境の構築が不要。 |
| Backup | AWSリソース全体、オンプレミスで管理しているデータのバックアップを作成 |
| CloudTrail | 管理コンソールへのログインや各種AWSサービスで行ったアクションを記録する |
|  |  |
|  |  |
| DLM;Data Lifecycle Manager | バックアップスケジュールを設定する |
|  |  |
| KMS;Key Management Service | 暗号化に使用するキーを作成・管理する |
|  |  |
| OpenSearch Service | オープンソースのデータ分析、検索とKibana(BI)の利用　旧：Elasticsearch |
| Quick Sight | 視覚的なデータ分析（BI） |
| Service Catalog | CloudFormationテンプレートを製品として登録し、他のAWSユーザーへ共有できる |
| Step Functions |  |
|  |  |
|  |  |
| TGW；Transit Gateway | 複数VPCを相互に接続するハブ |
| Trusted Advisor | サービス、コスト、セキュリティなどのベストプラクティスを提案 |

#### API Gateway

##### 概要

補足：あくまでAPIのゲートウェイのみで、実行環境ではない点に注意

|  |  |
| --- | --- |
| API | 概要 |
| REST API |  |
| HTTP API | RESTと同じ形式だが、APIキャッシュを利用できない |
| WebSocket API |  |

##### チュートリアル

###### Lambdaを併用したREST API

全体の流れ

・DynamoDB テーブルで操作を実行するための Lambda 関数をPython または Node.js で作成し、設定

・API Gateway で、Lambda 関数に接続するための REST API を作成する。

・DynamoDB テーブルを作成し、コンソールで Lambda 関数を使用してテーブルをテストする。

・API をデプロイし、ターミナルで curl を使用してセットアップ全体をテストする。

ポリシーの作成

IAM > ポリシー > ポリシーの作成 > ポリシーエディタでJSONを選択

[ lambda-apigateway-policy ]

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "Stmt1428341300017",

"Action": [

"dynamodb:DeleteItem",

"dynamodb:GetItem",

"dynamodb:PutItem",

"dynamodb:Query",

"dynamodb:Scan",

"dynamodb:UpdateItem"

],

"Effect": "Allow",

"Resource": "\*"

},

{

"Sid": "",

"Resource": "\*",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Effect": "Allow"

}

]

}

ロールの作成

IAM > ロール > ロールを作成 >

信頼されたエンティティタイプに「AWSのサービス」、ユースケースに「Lambda」で次へ

> 許可を追加　の画面で先程のlambda-apigateway-policyを選択して次へ

ロール名は「lambda-apigateway-role」とする > 作成

arnを控えておく　IAM > ロール > 検索 > ロールを選択　でarnを確認できる

次ページに続く

〜チュートリアル　Lambdaを併用したREST API　続き

[公式チュートリアルで不足している情報]

CreateFunctionを持ったポリシーを作成

[ LambdaCreateFunctions ]

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"lambda:CreateFunction"

],

"Resource": "\*"

}

]

}

IAMグループを作成し、任意のユーザーをグループに入れる

操作：

IAM > ユーザーグループ > グループを作成

・グループ名（例）

Lambda-CreateAndExcute

・ポリシー

LambdaCreateFunctions（上で作成したカスタムポリシー）

IAM > ユーザーグループ > グループを選択 > ユーザーを追加

cliを利用して

zip function.zip LambdaFunctionOverHttps.py

aws lambda create-function --function-name LambdaFunctionOverHttps \

--zip-file fileb://function.zip --handler LambdaFunctionOverHttps.handler --runtime python3.9 \

--role arn:aws:iam::123456789012:role/service-role/lambda-apigateway-role

##### API キャッシュ

あるリクエストとそれに対するレスポンスをキャッシュし、次に同一リクエストを受け取るとキャッシュからデータを返す事で（同一リクエストのみ）応答速度が向上する。

#### App Runner

コンテナ化されたウェブアプリケーションをデプロイする為のサービス。

#### Athena

##### 概要

S3内のcsv, jsonをテーブルとみなし、標準SQLでデータを抽出する。

・クエリはANSI SQLに準拠

・デフォルトではJSONには非対応SerDeというデータ処理が必要

・複雑な結合クエリなどには不向き。

・EC2インスタンスなどのセットアップ不要で。

補足

S3が構造化データ、非構造化データなどを格納できるオブジェクトストレージである為にこのような事が可能

##### SerDe；Serialize/Deserialize

DDL Data Definition

#### Backup

・EBSボリュームやAmazon RDSなどのAWSサービスのバックアップが対象

・インスタンスストアは非対応

#### CloudFront

##### 概要

高速なコンテンツ配信ネットワーク。オリジンサーバーへのリクエストを受けるリバースプロキシ。

出来る事

・国別にアクセスを制限できる

対象外

・DDoS攻撃から保護（代案：AWS Shieldを使う）

参考

CDN；Content Delivery Network（コンテンツ配信ネットワーク）

大容量のデジタルコンテンツをWeb上で大量配信する機能の事。

Webサーバーの負荷軽減の為に静的ファイル専用にリクエストにこたえるサーバー。

##### ユースケース

・S3に配置した静的コンテンツ(static)のキャッシュサーバー

・S3の静的WebサイトホスティングのHTTPSゲートウェイとして

補足：S3の静的Webサイトホスティングで提供されるのはHTTPのみ

##### ディストリビューション

コンテンツを配信するために必要な各種設定

キャッシュTTL キャッシュを保持する時間（単位）

補足：キャッシュTTLを含むキャッシュの設定のことを「キャッシュBehavior（ビヘイビア）」という

|  |  |
| --- | --- |
| コンテンツオリジンの指定 | 最大25個 |
| ユーザーアクセスの制御 |  |
| HTTPSの必須化 |  |
| キャッシュキーの設定 |  |
| オリジンリクエスト設定 |  |
| 地域制限 |  |
| ログの作成 |  |

##### その他機能など

###### エッジサーバー

世界各地に配置された配信用サーバー

オリジンサーバーのコンテンツを保管（キャッシュ）しておく

キャッシュが削除されるとオリジンサーバーに再取得に行く

###### Invalidation

エッジサーバーに保存されているキャッシュを即座に削除できる機能

###### OAI；Origin Access Identity

S3バケット内のコンテンツへのアクセスをCloudFront経由でのアクセスに限定する事で、ユーザーにS3に直セスアクセスできないようにする機能

###### フィールドレベル暗号化

ユーザーから送信されたHTMLフォームのPOSTリクエストの一部を、特定のアプリケーションでのみアクセス可能とする

###### 地理的制限

クライアントからのアクセスを国別に制限する機能

###### キャッシュBehavior

キャッシュの設定

#### DLM;Data Lifecycle Manager

EBS スナップショットや AMI の作成といったバックアップスケジュールを設定できるAWSサービス

バックアップスケジュールを設定する

#### Global Accelerator

##### 概要

複数のリージョンで展開しているWebアプリケーションなどへのリージョン間の負荷分散が可能

2つの固定パブリックIPアドレスを保有しており、エンドポイントが増減してもこの固定IPアドレスを用いてサービスを継続できる。

グローバルネットワークのエッジで動作し、ユーザーとアプリケーションの間のネットワークラウンドトリップを削減します。

特徴

・AWSが保有するグローバルネットワークを通じて効率的にルーティングする

（補足：通常AWSのリージョン間通信はAWS内のプライベート接続）

・オンラインゲームなどアプリケーションを用いたサービスのパフォーマンスを向上させる

パケットレベルのロードバランシングを提供し、高い可用性とパフォーマンスを実現します。

単一のIPアドレスに複数のAWSリージョンのエンドポイントを関連付けることができます。

バックエンドのAWSサービスだけでなく、オンプレミスのインフラストラクチャにも適用することができます。

比較

|  |  |
| --- | --- |
| Global Accelerator | ネットワークをレイテンシの観点で最適化。アプリやゲーム向き。 |
| CloudFront | ユーザーに近いところでキャッシュを提供するCDN。メディア配信に向く。 |
| ELB | EC2専用のロードバランサー |

|  |  |  |
| --- | --- | --- |
|  | Global Accelerator | CloudFront |
| 概要 | レイテンシの低いネットワークを構築する | CDN  (コンテンツ配信ネットワーク) |
| 通信プロトコル | HTTP/HTTPS、TCP/UDP | HTTP/HTTPSのみ |
| キャッシュ | 使わない | 使う |
| 得意分野 | 動的サイト | 静的コンテンツ |
| クライアント証明書 | 利用する | 利用しない |
| クライアントIP | 保存する | 保存されない |
|  |  |  |

##### NLB;Network Load Balancer

リージョン間の負荷分散はできない。

##### 具体例

複数のリージョンに配置されたEC2インスタンスのロードバランスをとる

⇒EC2インスタンスのフロントエンドにAWS Global Acceleratorを配置し、リージョン間で負荷分散する

#### Memcached

高パフォーマンスを提供

データの永続保持やレプリケーション機能は無い

シンプルなキャッシュとしての利用

#### OpenSearch Service

フルマネージド型の検索・分析エンジンサービス

テキストの全文検索やログ分析などの用途に利用

#### Route 53

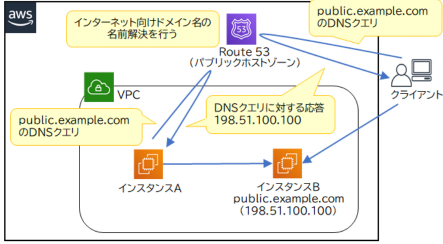
##### 概要

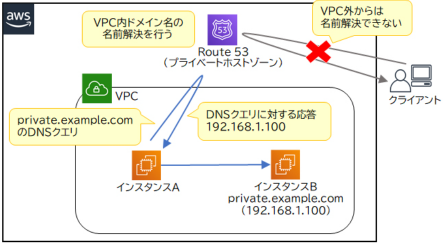
AWS版のDNS; Domain Name Serviceサーバー

##### ゾーン

補足：ゾーン　１つのDNSサーバーが管理しているドメインの範囲

パブリックホストゾーン

インターネットなどVPC外に公開されたドメインDNSレコードのこと

プライベートホストゾーン

VPC内に公開されたドメインのDNSレコードを管理する

##### ルーティングポリシー

概要

権威DNSサーバーに登録されているIPアドレスの中からDNSクエリのドメイン名に対応するIPアドレスを決める方針のこと。

|  |  |  |
| --- | --- | --- |
| 名称 | 概要 | 用途 |
| シンプルルーティングポリシー | 特定の単一のリソースに対してつかう。 | example.comに１つのウェブサーバーを提供 |
| フェイルオーバールーティングポリシー | ヘルスチェックの結果を基にアクティブ/パッシブフェイルオーバーのルーティングを構成 | 障害発生時に代替ページを自動的に表示 |
| 位置情報ルーティングポリシー | ユーザーの位置情報を基に、**任意のルールでルーティング** | クライアントの地域固有の言語で表示 |
| 地理的近接性ルーティングポリシー | クライアントと**物理的距離**が近いリソースのIPアドレスを自動的に回答 |  |
| レイテンシールーティングポリシー | レイテンシを最小にする為のルーティングを構成  ヘルスチェックも任意で利用可能 | クライアントから最も遅延が少なくリクエストを処理する |
| IPベースのルーティングポリシー | 送信元のIPが判明している時に、その位置に基づいてトラフィックをルーティングする |  |
| 複数値回答ルーティングポリシー | ヘルスチェックを複数のリソースに行い、ランダムに選ばれた最大 8 つの正常リソースを使用 | DNS クエリに応答する場合に使用。 |
| 加重ルーティングポリシー | 指定した被率でトラフィックを割り振る | オンプレミスとEC2インスタンスへのアクセスの割合を3:7にさせる |
|  |  |  |

公式：<https://docs.aws.amazon.com/ja_jp/Route53/latest/DeveloperGuide/routing-policy.html>

##### ヘルスチェック

ドメイン名に対応するリソースが正常に動作しているかを定期的に確認する。

正常であったリソースのIPアドレスのみDNSクエリの回答となる。

猶予期間

リソースが起動してから初回のヘルスチェックまでの待機時間

##### フェイルオーバー

ヘルスチェックを利用して、サービスを提供するリソースがプライマリからセカンダリへ自動的に切り替わる仕組み。

##### DNSレコードの種類

###### エイリアスレコード

Route 53用の特殊なDNSレコード。

・動的に変化するAWSサービスのIPアドレス（※）に対応する固定のドメイン名を登録する場合に使用する。

※ELBのパブリックDNSホスト名や、S3の静的Webサイトホスティングのエンドポイント（URL）など

・CNAMEレコードでも良いが、エイリアスレコードの方がDNSクエリへの応答が早くなる。

・Zone Apexの別名を登録できる。（通常Zone ApexにCNAMEは設定できない）

補足：Zone Apex　サブドメイン(wwwなど)を含まないドメイン名

###### NSレコード

ドメインのゾーン情報を管理するDNSサーバー

###### CNAMEレコード

###### MXレコード

ドメイン宛メールの配送先メールサーバー

###### AAAAレコード

ドメイン名に対応するIPv6アドレス

#### SQS；Simple Queue Service

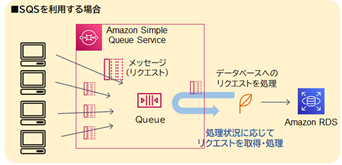
##### 概要

フルマネージドのメッセージキューイングサービス（第三者経由でのアプリ間通信）

・メッセージは最大256KB

・１つのキュー内のそれぞれのメッセージの優先度を変更する事はできない

⇒ 複数のキューにそれぞれ優先度を設定する事で達成できる



ユースケース

・DBシステムの負荷分散

・トラフィックの量が想定できないようなケース

（右図参照）

・Auto Scalingによって運用負荷を軽減するケース

・処理に優先順位を設定したいケース

|  |  |  |
| --- | --- | --- |
|  | SQS | SNS |
| 概要 | キューイングサービス | A2AとA2Pのメッセージングサービス |
| 詳細 | サービス対サービス | Publish – Subscribe型 |
| ユースケース | マイクロサービス  分散システム | 不特定多数へのメッセージ  アラーム通知 |
| 備考 | メッセージは明示的に削除する |  |

用語

|  |  |  |  |
| --- | --- | --- | --- |
| 用語 | 概要 | 備考 | MQTTで言うと |
| プロデューサー | 送信側 |  | Publisher |
| コンシューマー | 受信側 |  | Subscriber |
|  |  |  |  |

補足：AWS SNSではPublisher/Subscriberという言葉を使う

公式：<https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html>

##### 可視性タイムアウト

Visibility Timeout

メッセージがあるクライアントに受信された際、指定時間（デフォルト30秒、最大12時間）が経過するまで他のクライアントからそのメッセージを見えなくする機能。

受信クライアントが明示的な削除を完了するまでの間に別クライアントに受信されないようにする為の物。

個人的な意見：こんな機能設けるくらいなら、受信後に自動的に削除される機能を付ければ？

比較

|  |  |  |
| --- | --- | --- |
|  | 可視性タイムアウト | メッセージ保持期間 |
| 概要 | 受信クライアント以外から該当メッセージを隠す | 単純に永遠には保存されない、という話 |
| デフォルト長さ | 30秒 | 4日 |
| 期間の長さ | 0秒～12時間 | 1分～14日 |

公式：<https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html>

##### メッセージ保持期間

##### キュー

###### FIFOキュー

FIFO；First In First Out

送信された順番にキューに残っている最も古いものからメッセージを処理

（標準キューのメッセージの処理順序はベストエフォート（最善の努力）で、必ずしも送信された順とならない）

補足

・標準キューはFIFOキューよりもスループットが優れている

・FIFOキューの方が利用料金は高い

###### 一時キュー

###### Dead Letter Queue

別名：デッドレターキュー

一定の回数（1～1000）エラーになったメッセージを隔離するためのキュー

##### ポーリング

コンシューマーがキューに問い合わせた時に該当トピックのメッセージが無い場合の挙動が変わる

|  |  |  |
| --- | --- | --- |
|  | 該当トピックのメッセージ無い場合 | 備考 |
| ショートポーリング | 即座に「メッセージ無し」と送信 |  |
| ロングポーリング | 設定された時間（最大20秒）を待つ |  |

※pingtの説明ではメッセージが「無い」場合か、「中身が空」である場合なのか分からない（前者らしい）

※ポーリング（polling） 通信機器に一定間隔で順番に問合せを行うこと

補足：AWS SNSでは非対応

##### 遅延

###### 遅延キュー

メッセージの送信者が送信したメッセージを指定時間経過後に受信させたい場合に使用する機能

キューへの送信自体は即座に行われ、一定時間見えない状態でキューに配置された後、受信者へ送られる。

###### メッセージタイマー

###### 共通項目

・共に送信側が設定する

・キューへの送信自体は即座に行われる

・両方設定されている場合はメッセージタイマーが優先される

###### 比較

|  |  |  |
| --- | --- | --- |
|  | 遅延キュー | メッセージタイマー |
| 対象 | キュー全体に対して作用  （すべてのメッセージ） | 特定のメッセージに対して作用 |
| 概要 |  | 非表示状態でキューで待機する |
| 設定長さ | 0秒～15分 | 0秒～15分 |
| 備考 | 「即座に発動する可視性タイムアウト」とも言える | FIFOキューでは未サポート |

公式（メッセージタイマー）<https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-message-timers.html>

公式（遅延キュー）<https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-delay-queues.html>

#### SNS；Simple Notification Service

##### 概要

プッシュ型のメッセージングサービス

メールやモバイルへのプッシュ通知だけでなく、MQTT的な送信、Lambdaの起動などにも対応

・発行者はメッセージのプロトコルを意識する必要がない

・ほぼ無制限にスケーリングできるため、突発的にメッセージが増加した場合にも対応可能

MQTTとの比較

|  |  |  |
| --- | --- | --- |
|  | AWS SNS | MQTT |
| メッセージ形式 | テキストメッセージやJSONなどの複数の形式をサポート | バイナリデータやカスタムメッセージ形式をサポート |
| Publisher/Subscriberの制御／認証 | アプリケーションの制御や認証などを管理するAWSのサービスと連携 | アプリケーションが直接制御と認証を行う |
| スケーリングと耐久性 | 自動的にスケーリング | 自身で制御する必要がある |
| プロトコル | HTTP、HTTPS、AWS SQS、AWS Lambdaなどをサポート | IoTデバイスとの通信に特化 |
| 配信ポリシー | SNSではパフォーマンス、コスト、信頼性などの要件に応じてカスタム配信ポリシーを設定できる | ユーザーが配信の仕組みを設定する |

ユースケース

・複数のサブスクライバーに対して並列で処理（MQTTのような感じ）

・イベントの発生を契機にリアルタイムで処理を行いたいケース

CloudWatchで異常を検知したら即座に通知

・システム内のコンポーネントを低い結合度（疎結合）で連携するケース

・複数のコンポーネントで並列に処理を行うシステム

ユーザーの画像投稿をトリガに：

データベースへ登録する、サムネイルを作成、画像を解析する処理　を並列に行う事ができる

##### SNSトピック

###### 概要

SNSにおける情報の単位

###### イベントソース

|  |  |
| --- | --- |
| サービス名 | イベントタイプ |
| EC2 | 状態の変化、 |
| DB |  |

###### サブスクライバー

プロトコル

Amazon SQS

Lambda

Eメール

HTTP/HTTPS

JSON

SMS

モバイルへのプッシュ通知

Redriveポリシー

配信不能メッセージをデッドレターキューに送信

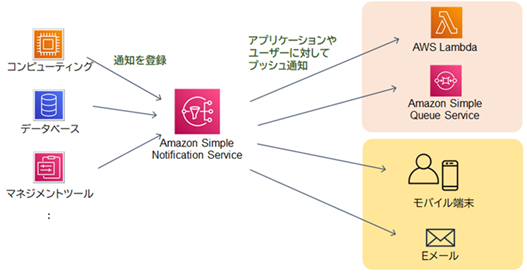
###### トピックの種類

|  |  |
| --- | --- |
| 種類 | 概要 |
| Standard(標準) | 最低限のサポートを提供する。一度に扱うメッセージ数に上限がある |
| FIFOトピック | メッセージが順番に配信されることを保証 |
| ダイレクトポイントモデル  Direct Publish Mode | 直接エンドポイントにメッセージを配信する |

比較

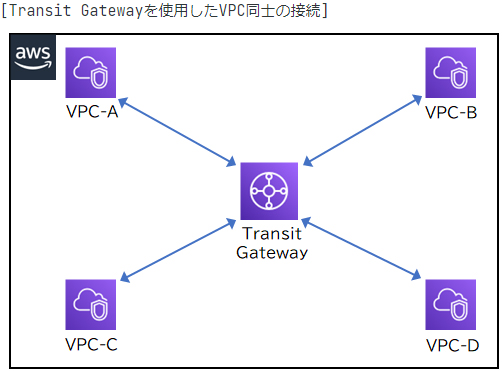
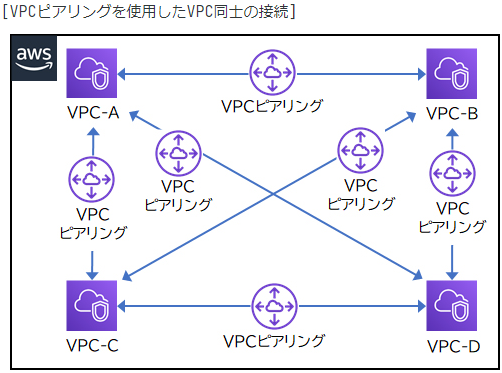
|  |  |
| --- | --- |
| SQSのFIFOキュー |  |
| Amazon SNSのFIFOトピック | 大量の受信者に対する通知向き |

##### 具体例



#### TGW；Transit Gateway

複数のVPCとオンプレミス環境をつなぐ仮想ルーターのようなサービス



### プログラム系情報

#### AWS CLI

別名：AWS Command Line Interface

##### 概要

コマンドラインから実行するCLIツール

[Linux](https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/getting-started-install.html)

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

Windows

Windows 用の AWS CLI MSI インストーラ (64 ビット) をダウンロードして実行します。

<https://awscli.amazonaws.com/AWSCLIV2.msi>

公式：<https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/getting-started-install.html>

##### コマンド

aws configure

環境変数で設定してもよい（らしい）

AWS\_ACCESS\_KEY\_ID、AWS\_SECRET\_ACCESS\_KEY、AWS\_DEFAULT\_REGION

#### python(awscli)

pip3 install boto3 awscli

比較

|  |  |  |
| --- | --- | --- |
|  | 概要 | 用途 |
| boto3 | 公式のAWS向きpython用SDK | アプリでAWSリソースを使う場合 |
| awscli | AWS CLIのPythonバージョン | AWSリソースの操作 |

#### boto3

##### 共通

##### S3

###### 概要

[ python ]

import boto3

s3\_client = boto3.client(

's3',

region\_name='us-east-1',

aws\_access\_key\_id='\*\*\*\*\*',

aws\_secret\_access\_key='\*\*\*\*\*'

)

response = s3\_client.list\_buckets()

print(response)

公式：<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>

###### get\_object

response = client.get\_object(

Bucket='string',

公式：https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3/client/get\_object.html

##### EC2

###### terminate\_instances()

###### run\_instances()

import boto3

ec2 = boto3.client('ec2')

response = ec2.run\_instances(

ImageId='AMI\_ID',

InstanceType='INSTANCE\_TYPE',

MinCount=1,

MaxCount=1

)

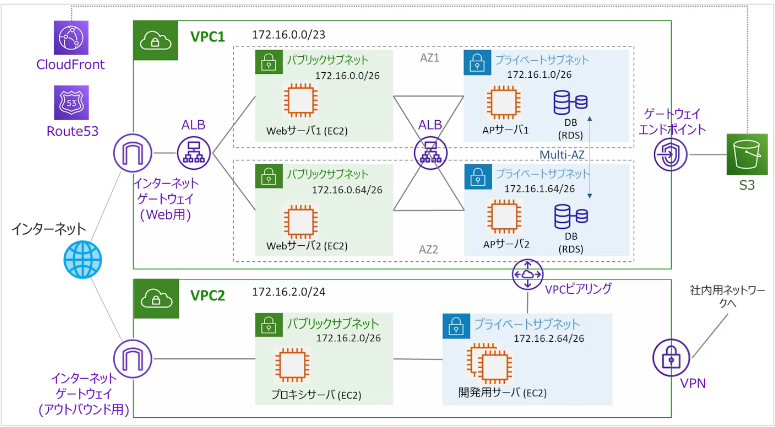
instance\_id = response['Instances'][0]['InstanceId']

print(f'EC2インスタンスが起動しました。インスタンスID: {instance\_id}')

### 具体例（構成例）

#### Webアプリ

##### 3層アーキテクチャ

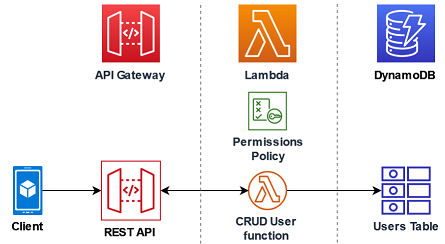


情報元：<https://qiita.com/c60evaporator/items/2f24d4796202e8b06a77>

##### LambdaとAPI Gatewayによる最小構成

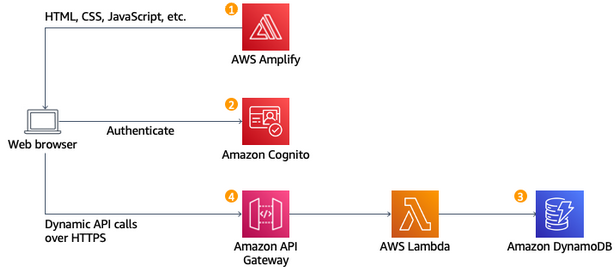
概要

Webブラウザから送信されたHTTPリクエストを受け取り、DynamoDBテーブルを検索して結果を返すWebアプリケーション



情報元：<https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/services-apigateway-tutorial.html>

##### Cognitoによる認証でREST APIを実行



情報元：https://aws.amazon.com/jp/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/

１　静的Webページのホスティング

２　ユーザー管理

３　サーバーレスのバックエンド

４　REST API

### その他

#### クライアント機からの利用

##### Linuxからの利用

#awscliのインストール

$ sudo apt install awscli

#awscliの最新化

$ sudo pip3 install awscli --upgrade

#awscliのバージョン確認

$ aws --version

##### ざっくりAWS料金

https://aws-rough.cc/

### 資料

#### 学習用資料

##### ロードマップ

・市販の本などを読み、全体像をつかむ

・EC2インスタンスを構築してみる

・IAMを触り倒す

・

注意点など

・料金がかかることもあり、「構築しては壊す」を繰り返した方が良い

・

参考

すごろく風ロードマップ：<https://qiita.com/KurokawaKouhei/items/4e9aa3b526f3f233bf85>

補足：少し古い情報ではあるが、分かりやすく感じた

##### チュートリアル系

###### 公式のskill builder

<https://aws.amazon.com/jp/training/digital/>

##### 具体的な構成例

CAPSULE CLOUD

<https://capsulecloud.io/aws-casestudy>

##### 資格

・クラウドプラクティショナー

#### 参考資料

AWSサービスを１行で説明：<https://gigazine.net/news/20200528-aws-one-line-explanation/>

### 用語

#### 責任共有モデル

AWSが提供するサービスの範囲はAWSがセキュリティ対策の責任を持ち、サービスを利用する上でユーザーが設定・変更できる範囲はユーザーが責任を持つ、という考え方。

#### Well-Architected Framework

AWSが提唱するクラウド設計・運用のベストプラクティス集

一般的な設計原則

・本稼働スケールでシステムをテストする

- 本番環境と同等の環境でテストを行う

・必要キャパシティーの推測が不要に

- 需要の増減に応じてリソースを変更

・自動化によってアーキテクチャでの実験を容易に

・ゲームデーを利用して改善する？？

・発展するアーキテクチャが可能に

・データに基づいてアーキテクチャを進化させる

- 各リソースのメトリクスを自動的に収集する事を推奨

<https://wa.aws.amazon.com/wat.design_principles.wa-dp.ja.html>

#### Wild Rydes フリート

AWSのチュートリアル資料で出てくる、ユニコーンを使ったタクシーを提供する架空の会社。

フリートとはfleet（船団）のことだが、ビジネス領域で企業が所有・運用する車両も表し、レンタカー会社やバス会社でも用いられる。

## Azure

### 特徴

Microsoftのサービスなので，当然Windowsサーバーと親和性が高い。

案外Excel用のAPIとかは存在しないのかも。

### API

#### Microsoft Face API

### Cosmos DB

Cosmos DB NoSQL DB。1桁ミリ秒での高速データアクセスも可能。JOINを使わないようなシンプルな構成が得意

Azure Cosmos DB Emulator

Docker for Windowsで動く

### Azure Lab Services

教育／ハンズオン環境に向いたGoogle Colaboratoryと言った感じ（らしい）

## GCP；Google Cloud Platform

### 概要

|  |  |
| --- | --- |
| サービス名 | 概要 |
| BigQuery | ビッグデータを解析。 |
| Google App Engine | Googleのインフラ上でウェブアプリケーションの作成・実行・管理。 |
| Cloud Machine Learning Engine | 機械学習モデルを簡単に構築できます。 |
| Google Cloud Functions | FaaS（Function as a Service） |
| Cloud SQL | データベースサービス |
| Kubernetes | コンテナを提供 |

（参考）<https://cloud.google.com/iot/docs/quickstart>

### チュートリアル

Google Chromeから<https://console.cloud.google.com/> にアクセス。

Googleアカウントでログイン

### Google text-to-speech

・上の検索BoxにText-to-Speechと入力

もしくはAPIとサービス → APIとサービスの有効化

※念の為、「割り当て」でリクエスト上限を指定して置くこと

（2020.03.03現在）

|  |  |  |
| --- | --- | --- |
| 機能 | 1 か月あたりの無料枠 | 有料使用 |
| 標準（非 WaveNet）音声 | 0〜400 万文字 | 100 万文字あたり $4.00（USD） |
| WaveNet 音声 | 0〜100 万文字 | 100 万文字あたり $16.00（USD） |

・認証情報→概要→認証情報を作成

（認証情報がない場合：もう一度Text-to-Speechを検索→管理）

・I!Mと管理→サービスアカウント→

キー→キーを追加→新しいキーを作成→JSONを選んで作成

インストール（仮想環境を推奨）

pip install --upgrade google-cloud-texttospeech

oauth関連：

pip install google-api-python-client google-auth-httplib2 google-auth-oauthlib

oauth公式資料：https://cloud.google.com/docs/authentication/end-user?hl=ja

有効化まではここがわかりやすい：https://blog.apar.jp/web/9893/

公式チュートリアル：<https://cloud.google.com/text-to-speech/docs/quickstart-client-libraries?hl=ja>

### Colaboratory

通称：Colab

#### 概要

・Jupyter Notebookのようなもの

・GPUの仮想環境が利用可能。

・複数人で開発環境を共有できる。

環境

python3.10

.whはlinux\_2\_17\_x86\_64用のを選べば良いっぽい

比較：Azure Lab Services

#### チュートリアル

##### 基本設定

GoogleドライブでColab向きの設定

→ 新規 → その他 → アプリを追加 → Colaboratoryを選択 → インストール

Google ドライブに.ipybファイルを作成

新規→その他→Google Colaboratoryを選択

試しにセルに以下のように記述する

!pip list

→セルを右クリックして「現在のセルを実行」すると、pipパッケージのリストが表示される。

##### GPUの使用

2022年8月現在、連続12時間まで無料でGPU(Tesla K80 or Tesla T4)を使用する事ができる。

90分GPUへの？アクセスがないと接続が切れる

メニュー：ランタイム > ランタイムのタイプを変更 > ハードウェアアクセレレータを「GPU」に。

GPUの情報は以下のようにすると確認できる。

[ notebook(colab) ]

!nvidia-smi

コンボボックスが表示される

use\_custom\_dataset = False #@param ["False", "True"] {type:"raw"}

#### 詳細

##### colabでpythonのバージョンを変更する方法

!sudo update-alternatives –config python3

補足

colabではpythonが常に起動しているような状態

!を付ける事でbashコマンドを実行する

##### Google Driveのマウント

from google.colab import drive

drive.mount('/content/drive')

・/content/というのが作業ディレクトリらしい

##### zipアーカイブの作成とダウンロード

補足

各ディレクトリのパスは左側のファイルエキスプローラーでフォルダを右クリックして「パスをコピー」を選択すると取得できる。

##### python内の変数をbashに渡す

普通に$つければbashで使える

t = "test"

!echo $t

# 対象ディレクトリ内のディレクトリのみ抽出してlistを作成する

TARGET\_DIR = './'

dir\_names\_list = [ f for f in os.listdir(TARGET\_DIR)

if os.path.isdir(os.path.join(TARGET\_DIR, f))]

# １つずつ名前を表示

for dir\_name in dir\_names\_list:

!echo %dir\_name

##### リファレンス

###### files

#### トラブルシューティング

##### NotImplementedError: A UTF-8 locale is required. Got ANSI\_X3.4-1968

import locale

locale.getpreferredencoding = lambda: "UTF-8"

### Kubernetes

クバネティス/クバネテス

### Google Drive

共有ドライブ

有料版？エンタープライズ版？だと「共有ドライブ」

・共有ドライブのオブジェクトのオーナーは共有ドライブとなる。

### Google スプレッドシート

#### チュートリアル

##### pythonで使う

###### 事前準備

Google Cloudへログイン：<https://console.cloud.google.com/>

初回は利用規約への同意が必要

プロジェクト作成

 →  → 適当な名前を付ける → 

ライブラリ追加

（ナビゲーションメニュー）→ 

→ ライブラリ：Google Drive API，Google Sheets APIを有効に

認証情報

（APIとサービスの画面で）→ サービスアカウント

→ アカウント名を付ける。（説明は任意でつける）→ 

ロールは「オーナー」を選択。→ 「続行」

キーを作成

今作ったサービスアカウントの （サービスアカウントを編集）をクリック。

→ キー → 鍵を追加 → 新しい鍵を作成 → キーのタイプ：JSON → 作成

→ キーがWebブラウザを通じてダウンロードされる。

###### スプレッドシートを共有

または，既に作成してあるシートを開く。

先ほど作った.jsonのキーのファイルを開き，”client\_emailの値をコピーしておく。”

スプレッドシートに戻り  をクリック → コピーしたメールアドレスを張り付け → 完了

###### gspread

python -m pip install gspread # インストール

python -m pip install oauth2client # インストール

test.py

import gspread as gc

# 下のいずれか一つで開く

spread = gc.open('ファイル名')

spread = gc.open\_by\_key('スプレッドシートのキー') # /dと/editの間の英数字

spread = gc.open\_by\_url('URL')

# 各オブジェクトの取得

wsheet = spread.get\_worksheet(0) # シートを取得：番号は0から始まる

values\_list = wsheet.row\_values(2) # 行の値を取得：番号は1から始まる

補足：1行目はヘッダになっている事が多いので，実際のデータは2行目から

（参考資料）<https://zak-papa.com/python_gspread_setting>

（公式資料）<https://docs.gspread.org/en/latest/>

### Google Calendar API

主に以下の用途で使われている模様

・予定管理

・休日判定

### Google Map API

特にhtmlの中で使うらしい。言語はJava？

### Cloud Function

#### 概要

GCP内のサービスの一種

curlで実行するらしい

#### チュートリアル

表示されたURLにcurlでアクセスする事で実行が可能。

参考：<https://qiita.com/kai_kou/items/dca21cdfd8375a247c2f>

### Cloud Run

コンテナを動かすサービス

比較：AWSのFargate（ファーゲート）；App Runner

動作言語

Go

Node.js

Python

Java

.NET Core

### GAS ； App Script

#### 概要

スプレッドシート版のVBAといったところ。

・書き方はJavaScriptに近い。（らしい）

・JavaScriptやPythonほど機能が多くない

<https://script.google.com/home>

高度な処理

・BigQueryと連携

種類

Standalone Scripts

１つのスクリプトファイルをGoogle Drive上に作成し、実行するシンプルなもの。

Container-Bound

G Suiteアプリ（スプレッドシート、ドキュメント、スライド、フォーム等）にくっついて保存されるタイプのスクリプト。別名：バンドル型

Web Apps

htmlを利用したWebアプリに対応するスクリプト

#### トリガ

対応表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | スタンドアロン | バンドル | Web  アプリ |  |
| 時間主導型 | ○ | ○ |  |  |
| カレンダーから | ○ | ○ |  |  |
| スプレッドシートから |  | ○ |  | 起動時、編集時、変更時、フォーム送信時 |

#### チュートリアル

公式：https://developers.google.com/apps-script/samples?hl=ja

##### スタンドアロン

Google Drive → 新規 → その他 → Google Apps Script

公式：<https://developers.google.com/apps-script/guides/standalone?hl=ja>

トリガ

左メニュー： →

##### バンドル型

作成方法

スプレッドシートを開く → メニュー：拡張機能 → Apps Script

トリガ

変更時 セルの値を変更した時

編集時 「変更時」に加え、列や行の追加・削除の操作も含まれる

##### Webアプリ

#### レファレンス

### Cloud Run

Cloud Run はサーバーレスで提供される GCP のサービスで、フルマネージドなコンテナ実行環境。

Cloud Run はリクエストがないときにインスタンス数が 0 になるまでスケールインできる点が大きな特徴

### 費用・ライセンス

Google Workspace

オフィス製品，メール，Google Driveなど

|  |  |  |  |
| --- | --- | --- | --- |
|  | Business Starter | Business Standard | Business Plus |
| 料金 | 680/月 | 1360/月 | 2040/月 |
|  |  |  |  |

Google Could Platform

Always Free

いつまででも無料で使える機能の事。（当然限定される）

G Suiteに関するAPI どこにも明記無いが，無料らしい

(docsは明記あり) <https://developers.google.com/docs/api/pricing>

Storage 5GB／月

Cloud Functions 200万回／月

Speech-to-Text 60分まで／月

（参考資料）<https://cloud.google.com/free?hl=ja>

##### 上限設定

下の２点がポイント

・アラートを作成する

ナビゲーションメニュー→お支払い→予算とアラート

・リクエスト上限を設定する

上限金額が設定できた気がするが？？

そのほか

ストレージ（Google Drive）15GBまでは無料

## Google Charts

### 概要

htmlに埋め込む形で可視化するツール軍。

Java Script で記述する？

（公式チュートリアル）<https://developers.google.com/chart/interactive/docs/quick_start>

## Heroku

PostgreSQLを使用

対応言語

Node.js Ruby Java PHP Python Go Scala Clojure

## SAP Cloud Platform

## IBM Cloud

## Acquia

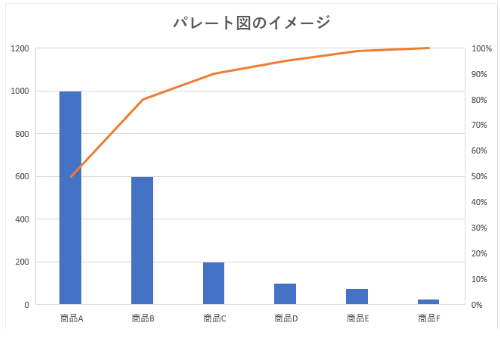
## そのほか

OpenWeatherMap API

# データ可視化

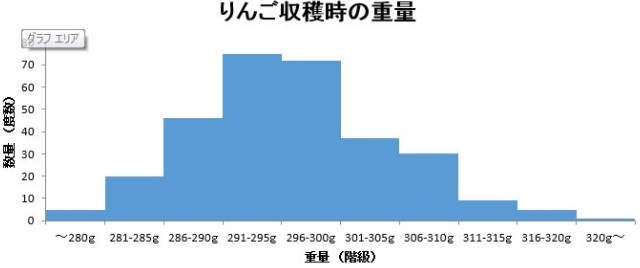
## グラフ

### パレート図

値の大きい順に分析項目を並べた棒グラフと、累積構成比を表す折れ線グラフを併記した複合グラフ。

重要度の高い要素の分析に用いられる。

「上位２割の要素が全体の８割を占める」をいう「パレートの法則」を可視化するためのツールともいわれる。

ヒストグラム

量的データの分布の様子を見るのに用いられます。 データをいくつかの階級に分け、度数分布表を作成してからグラフを作成します。 横軸にデータの階級を、縦軸にその階級に含まれるデータの数（人数、個数など）をとります。

# 色々なWebAPI

## e-stat

日本の統計情報

<https://dashboard.e-stat.go.jp/api/1.0/Json/getIndicatorInfo>?

## Gmail API

## OpenWeatherMap API

https://api.rakuten.net/community/api/Open%20Weather%20Map

## Wikipedia

Wikipedia Extractor

<https://github.com/attardi/wikiextractor>

JSON

curlでJSONデータをダウンロード

curl -sLH 'Accept: application/json' http://www.wikidata.org/entity/Q233495 | jq . > shinagawa.json

wgetで.bz2ファイルをダウンロードして解凍。

DBPedia

少し古い情報らしい・

## 気象庁（json)

### 天気

東京　<https://www.jma.go.jp/bosai/forecast/data/forecast/130000.json>

名古屋　<https://www.jma.go.jp/bosai/forecast/data/forecast/230000.json>

天気予報（明後日まで・週間）://www.jma.go.jp/bosai/forecast/data/overview\_week/(pathCode).json

天気概況（明後日まで）://www.jma.go.jp/bosai/forecast/data/overview\_forecast/(pathCode).json

天気概況（週間）://www.jma.go.jp/bosai/forecast/data/overview\_week/(pathCode).json

アメダス://www.jma.go.jp/bosai/amedas/const/amedastable.json

[://www.jma.go.jp/bosai/amedas/data/map/(YYYYMMDDHH0000).json](https://www.jma.go.jp/bosai/amedas/data/map/(YYYYMMDDHH0000).json)

注意点

・都市コードはhtmlサーバーにアクセスするとわかる

<https://www.jma.go.jp/bosai/forecast/>

・weathersやwindsが２つ（以上）あるのは、今日、明日、明後日の予報（らしい）

・このjson、なにやら作りがおかしい。

いきなりリストになっていて、その後dict型になっているような構造。

[

{

# こんな感じ。ここがdict

}

]

補足：さらに言うと、改行されていない。<https://jsonlint.com/>などで整列する必要がある。

その為、以下のような取得方法になる

[ weather\_test.py ]

import json

import requests

# 愛知県西部 [230000]の天気データ

url = 'https://www.jma.go.jp/bosai/forecast/data/forecast/230000.json'

header = {"content-type": "application/json"}

response = requests.get(url, headers=header)

data=response.json()

weather\_data=data[0]['timeSeries'][0]['areas'][0]['weathers'][0]

なぜ、こんな仕様にしたのだろうか？？2022-02-13

### 暑さ指数（WBGT）

<https://www.wbgt.env.go.jp/data_service.php>

# クラウドアプリ（SaaS）

## Salesforce

顧客関係管理ソリューションを中心としたクラウドコンピューティング・サービス

見込み顧客との接点を創り出して強化し、成約までの一連のプロセスを効率化・最大化させる

# コマンド実行（FaaS）

## 概要

### 情報

### 主流サービス一覧

|  |  |  |
| --- | --- | --- |
| サービス名 | 対応言語 | 備考 |
| AWS Lambda | C#, Go, Java, Python, Ruby | 最もポピュラー |
| Azure Functions | C#, F#, Node.js, Java |  |
| Google Cloud Functions | Node.js, Python |  |
| IBM Cloud Functions | Node.js, Python, Swift, Ruby, PHP |  |
| OpenWhisk |  | IBM |

参考資料：<https://blog.apar.jp/web/11017/>

## AWS Lambda

## KEDA

### 概要

k8s上で動作するイベント駆動型のオートスケーラー

### チュートリアル

### 資料

<https://zenn.dev/nkte8/articles/2021-11-28-r01>

# 開発環境(BaaS)

## Firebase

### 概要

モバイルアプリやWebアプリを開発するためのプラットフォーム。

簡易的なデータベース？csv？のようなものを提供するらしい。

そのデータベースにはJavaScriptを使って接続するらしい。

BaaS（Backend as a service）の1つ。

より厳密にいうとmBaaS（mobile Backend as a Service）らしい

mBaaS

モバイルアプリを開発する際に必要となるサーバーやデータベースなどのバックエンド機能をワンパッケージにした，モバイルアプリ開発プラットフォーム。

料金

|  |  |  |
| --- | --- | --- |
| Spark |  |  |
| Flame |  |  |
| Blaze |  |  |

アプリのプロジェクト ローカル用アプリを作成。Android Studio、Flutterなどを使う

Firebaseプロジェクト WebのFirebaseコンソールから作成

https://firebase.google.com/pricing

大きく以下の機能がある

|  |  |
| --- | --- |
| Authentication | ユーザー認証機能。ログイン。  ソーシャルログイン（Twiiterなど）の利用も可能 |
| Database | Realtime Database、Cloud Firestore の２種類。いずれもNO SQL |
| Storage | オンラインストレージ |
| Hosting | Webアプリを提供 |
| Functions | RESTのようなFaaSを提供 |

### チュートリアル

https://firebase.google.com/

→ コンソールへ移動 → プロジェクトを追加

利用方法

CDNでhtmlに埋め込む

<script src="https://www.gstatic.com/firebasejs/5.8.4/firebase-app.js"></script>

Node.jsで利用する

const firebae = require("firebase")

## Bubble

## Airtabe

## Google App Sheet

## Zero Creater

## PowerApps

Microsoftが提供するアプリケーションプラットフォーム。Office365と親和性が高い。

## Kintone

Cybouzuが提供するクラウドベースのアプリケーションプラットフォーム。

Salesfors、Zapierなどの他のクラウドアプリとの親和性が高い。

# 仮想サーバー(VPN)

Lightsail

月額$3.5~

EC2

リクエスト数で料金が決まる

Compute Engine

Google

リクエスト数で料金が決まる

E2 低価格

N1 VM の第 1 世代

N2 高パフォーマンス

[https://cloud.google.com/compute/docs/machine-types?hl=ja#recommendations\_for\_machine\_types](https://cloud.google.com/compute/docs/machine-types?hl=ja" \l "recommendations_for_machine_types)

# BIツール

## BIまとめ

### 概要

viewデータのGUI的まとめアプリを作成するツール群，といった感じ。

主な機能

レポーティング機能

BIツールで収集・分析したデータを、ダッシュボードやグラフなどにして可視化し共有する機能。

OLAP分析機能

OLAPとは、Online Analytical Processing（オンライン分析処理）の略で、問題がある指標に対し、要因を掘り下げ特定する機能。ドリルダウン、スライシング、ダイシングという3つのデータ分析操作によって、データを多次元的に分析する。

データマイニング機能

蓄積したデータを、統計学・パターン認識、人工知能などを用いて有効な傾向や予測を発見する機能。

プランニング機能

売上や収益予測の際に、蓄積された過去のデータを分析し、その分析を元にシミュレーションを実行し、その計画の根拠を確認する機能。

（参考）[https://kigyolog.com/service.php?id=138#2-1](https://kigyolog.com/service.php?id=138" \l "2-1)

### ピックアップ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | PowerBI | Data Portal | Tableau | Metabase | Gafana |
| 形式 | クラウド | クラウド | クラウド/オンプレ | クラウド | オンプレ |
|  |  |  |  |  |  |
| ノーコード | 〇 | 〇 |  | 〇 |  |
| 無料 | 〇 | ◎ | △（2週間） | △（◎；OSS） | 〇 |
| 運営 | Microsoft | Google |  | Metabaseﾌﾟﾛｼﾞｪｸﾄ | Gafana Labs |
| 備考 | Power BI Desktopもある。 | ・旧Data Studio  Google Chartとの違いは？ | ・Desktop版（可視化のみ）有り | ・OSS Edition有  （Docer又は.jar）  ・比較的簡単 | ・Enterprise版とOSS版がある  ・Clound版もある  ・日本語表示なし |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Kibana | Redash | Superset | Qlik Sense | Pentaho |
| 形式 |  |  |  |  | ソフトウェア |
|  |  |  |  |  |  |
| ノーコード |  |  |  |  | 〇 |
| 無料 |  |  |  |  | 〇 |
| 運営 | Elasticsearch | Re:dashﾌﾟﾛｼﾞｪｸﾄ | Apache |  | Pentaho |
| 備考 |  | ・Dockerのみ  ・少し古い？使いにくいとの情報もあり |  |  |  |

無料：△無料期間あり；○機能限定で無料；◎完全無料

### 箇条書き

Tableau 分析に向く。

Qlik Sense 分析に向く。無料でも使える。

Power BI 無料でも使える。

SAP

Google Data Studio 無料版は機能が限定。レポート作成に向く？

Quick Sight AWS。

Domo

MotionBoard

MicroStrategy

Actionista!

LaKeel BI

Metabase 日本語表示なし？オープンソース。バグが多い？

Grafana 幅広いデータソースと連携

Re:dash 幅広いデータソースと連携

Kibana ビッグデータの解析

Graylog 高速なデータ処理やリアルタイム分析

Pentaho （ペンタホ）無料でも使える。現在は日立ヴァンタラ社が提供。

Zoho

Apache Superset ビジュアル化の種類が豊富。オンプレミス可。

Salesforce Analytics

Tibco

### ダイレクトクエリ

通常BIツールはDBからのデータのキャッシュを保持しているが、キャッシュ0で常にクエリが発行されるシステムをダイレクトクエリと呼ぶ

## 2022-11-10 pickup

・ダッシュボードが使える

・オンプレ

・操作が簡単

⇒要するに可視化ツール、または超シンプルなBIツールが欲しい

無料

Re:dash

Grafana

資料が英語。

Pentaho

Metabase

直感的な操作はあまりできないらしい。

バグも結構あるとか？

有料

FineReport

90日間無料で使える

Looker

料金は問い合わせるしかない

ライブラリ

JavaScriptのデータ可視化ツール

◎Echarts（Apache）

〇Google Charts

vega

Kibana（Elastic社） Elastic Search専用ツール

Superset（Apache）

## POWER BI

マイクロソフトが出しているBI(ビジネス・インテリジェンス）

主にデータベースのデータの視覚化，分析補助を行う。

対象はクラウド上のデータでも，社内サーバー（オンプレミス）でも良い。

ファイル：Excel，CSV，

DB：Azure SQL Database，PostgreSQL，Oracle，Accessなどなど

情報元：<https://docs.microsoft.com/ja-jp/power-bi/connect-data/power-bi-data-sources>

Analyzing Data with Microsoft Power BI という資格も存在する。

|  |  |  |
| --- | --- | --- |
| プラン | | 機能 |
| 無料版 | | ほかの人と共有できず、個人使用。 |
| 有料  （サブスク） | ユーザー | 他人が発行したWebレポートを閲覧できる。 |
| ノード | 大規模な公開に向く |

### クラウド版

業務用ドメインのメールアドレスが必要？（hotmailやgmailではダメ）

### 対応データソース

（公式）<https://docs.microsoft.com/ja-jp/power-bi/connect-data/power-bi-data-sources>

MongoDBはODBC経由でならOKらしい。

### チュートリアル

レポート メイン画面

モデル ER図（Entity Relationship Diagram）の表示と編集

#### 概要

##### サイドバー（左）

レポート メイン画面



データビュー

モデル ER図（Entity Relationship Diagram）の表示と編集

各列の画を他のテーブルの列にドラッグ&ドロップするだけでリレーションが作れる

##### 自作メジャー

レポートorデータタブの「新しいメジャー」

メジャー名=DAX関数

で作成

詳細は「DAX（メジャーへ）」

#### ナビゲーター

読み取り

データの変換

#### メモ

### レポート画面

#### 「挿入」フィールド

#### 「視覚化」フィールド

レポート画面の右側サイドバー

地図

ファイル>オプションと設定>オプション>グローバル>セキュリティ>地図と塗り分け地図の画像を使用する

スライサー

#### 「フィールド」フィールド

レポート画面の右側サイドバー

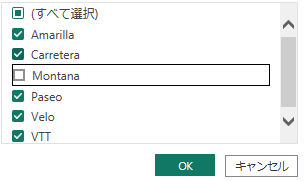
### データ画面

新しい列

### 具体例

#### データのフィルタリング

元データで絞る

Power Queryエディター

= Table.SelectRows(#"{列名} ", each ([Product] <> "Montana"))

列のドロップダウンを利用してチェックボックスでも対応可能（右図参照）

#### 前の行の値を参照する

##### DAXでやる方法

・まずは「インデックス」列を作成

・「まえの行のインデックス」を表す列を追加

[ DAX ]

last index = [インデックス] - 1

・「まえの行の値」を表す列を追加

last value = LOOKUPVALUE(my\_table[取得したい列], my\_table[インデックス], my\_table[last index])

補足

時間の場合の計算

[ DAX ]

interval = (MINUTE([時刻]) \* 60 + SECOND([時刻])) - (MINUTE([last time]) \* 60 + SECOND([last time]))

##### Power Queryでやる方法

ある？ 2023-02-10

インデックス列を作成

リストを作成

対象列を選択 →「変換」タブ → 「リストに変換」

#### データソースの修正

「レポート」ビューの「フィールド」：クエリの編集 > Power Queryエディターへ

詳細エディター

### レファレンス

#### レポート

##### メジャー

データ分析に対してよく使用される計算方法を保存したもの。

例えば数値であれば「合計」「平均」といったメジャーが自動的に割り当てられる。

##### パーセント表示

値の「」を クリック→値の表示方法→統計のパーセント

#### データ

#### モデル

リレーション　ドラッグ&ドロップでできる。

#### 表の種類

##### スモールマルチプル

１つの区域に複数のグラフを表示する。

その他のビジュアルを取得

例として「BciCalendar」を取得

ファイルをダウンロードして取得する事も可能

<https://github.com/mannymerino/bci-calendar/blob/master/dist/bciCalendar.pbiviz>

BciCalendar.1.0.0.0.pbivizをダウンロード

#### グラフ関連

２軸以上のX軸

総計のパーセント

### ビジュアル（グラフ）

・ファイルからインポート

・

### DAX（メジャー）

DAX（Data Analysis Expression）

#### 概要

メイン画面の「ホーム」タブ → 「計算」セクション → 新しいメジャー

作成すると、レポート画面において新しい列として使用可能

#### 統計

SUM

SUM( {テーブル名} [ {列名} ])

#### 条件式

#### ループ

##### each

構文：each {条件}

each Date.IsInCurrentYear([日付列])

each if [インデックス] = 0 then 0 else [日時])

前のインデックス =

VAR xID = [ID]

#### CALCULATE

#### DATESINPERIOD

#### DIVIDE

#### SUM

#### FORMATE

datetime型の値の文字列書式を指定する

#### LOOKUPVALUE

<https://learn.microsoft.com/ja-jp/dax/lookupvalue-function-dax>

#### Table.RecorderColumns

#### Table.AddColumn()

書式：ADDCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]…)

table データのテーブルを返す任意の DAX 式。

name 列に指定された名前。二重引用符で囲みます。

expression スカラー式を返す任意の DAX 式。table の各行に対して評価されます。

<https://learn.microsoft.com/ja-jp/dax/addcolumns-function-dax>

#### 資料（DAX）

https://learn.microsoft.com/ja-jp/dax/dax-function-reference

### POWER QUERY

開き方

・Power BI Desktopのデータの変換をクリック

・

概要

データの抽出条件はGUI上で大体行う事が出来る（右図参照）

データ型の変更

列ヘッダを右クリック→型の変更

#### ツールバー

##### ホーム　タブ

###### 変換　セクション

データ型

###### その他のセクション

閉じるセクション – 閉じて適用

##### 変換　タブ

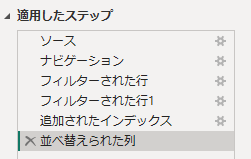
###### 任意の列　セクション

値の置換

###### テキストの列　セクション

データを文字列に変換する際の書式設定などができる。

#### Csvの表題行の削除

概要

編集には独自言語（DAX）を用い、その履歴は「適用したステップ」に保存される。



データの変換→データの変換

行の削除



行の削除→上位の行の削除（上から行を削除するの意？）→行数：１

表題の変更

　Column1を選択→名前の変更



#### 列の追加

メニュー：列の追加　>

カスタム列

条件列 GUIを用いて条件に当てはまる

インデックス列 データを扱う為に使うインデックス値を扱う列

DAX（Data Analysis Expression） エクセルの関数のようなもの

計算式の中に列を入れる場合には [列名] とする。

DAXにおいては、テーブルは'table\_name’、列は[column\_name]で選択する。

#### 列ツール

表示書式を変更したりできる。

#### カレンダーテーブル

→新しいテーブル

=calendar(開始日,終了日)

### スクリプト

#### python

##### 概要

大きく、データ読み込みと可視化の用途でpythonが利用可能

データ読み込み

効果

・pandas(numpy)のデータ読み込みパフォーマンスを享受できる

→ pandasは結構早い

・PowerQuery(M言語)はロードに徹し､Python側でデータ加工という運用ができる｡

→ WebAPIを直接叩いてデータを取得するような場合に役に立つ

メモ

ただし、PowerBI側のpythonを書く画面が貧弱なので使う気が起きない。

別ツールで加工スクリプトを書いておいて、コピペ、という運用になる。⇒いまいちな印象

可視化

・PowerBI既定やカスタムビジュアルで実装していないときに便利。

・機械学習して予測値を出すような場合に使える

・PowerBIではProライセンスが必要だが、PowerBI Desktopでは使える

##### チュートリアル

PowerBIの設定

ファイル > オプションと設定 > オプション

> グローバル：Pythonスクリプト

仮想環境

Pythonホームディレクトリを設定します

注意点：venv/Scriptsまで選択する必要がある

仮想環境の準備

python -m venv venv

venv\scripts\activate

pip install pandas matplotlib

データを取得

メニュー：ファイル > データを取得 > その他 > Pythonスクリプト

import pandas as pd

sample\_csv = pd.read\_csv("D:\\path\\to\\sample.csv")

### Power BI Desktop

#### 概要

Microsoft Storeで入手

法人向けのemailアドレスが必要

#### チュートリアル

X軸にドラッグ＆ドロップ

#### データベースの取り込み

##### PostgreSQL

Windows

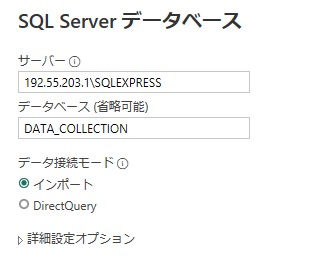
データベース→PostgreSQLだと何故か接続できない。

ODBCの設定を先に行い、ODBC→ODBC設定名→ユーザー名とパスワードを入力

PowerBI

データを取得→その他→ODBC→データソース名で上で設定したものを選択。

##### SQL Server

SQL Serverからデータをインポートする >

又は　リボン >

>「このデータベースへのアクセスには…」と表示 >

左側の「データベース」というタブを選択

→ ユーザー名、パスワードを指定

> ナビゲーター　画面でテーブルを選択

ちなみにPower Queryエディターが開く

#### 基本機能

##### クエリ変換

##### スライサー

Ctrl押しながらクリックで複数選択。

#### 詳細

##### 自動更新

Power BI Dektopメニュー「ファイル」>「オプションと設定」>「オプション」

「プレビュー機能」から「ページの自動更新」にチェック　無いが？？？

補足：対象はDirect Queryデータソースのみ

SQLServer、PostgreSQL、MariaDBは対象。MySQLは非対称。

公式のDB毎の接続情報：

[https://docs.microsoft.com/ja-jp/power-bi/connect-data/power-bi-data-sources#data-sources-s](https://docs.microsoft.com/ja-jp/power-bi/connect-data/power-bi-data-sources" \l "data-sources-s)

### 学習資料

公式

https://learn.microsoft.com/ja-jp/power-bi/

https://learn.microsoft.com/ja-jp/training/powerplatform/power-bi

https://learn.microsoft.com/ja-jp/power-bi/create-reports/desktop-excel-stunning-report

パソナ

https://lab.pasona.co.jp/data-operation/skill/812/

メモ

書式設定で画面の比率を16;９とかにできる

動的なデータ移動はできる？

## Google Data Portal

### 概要

旧：Google Data Studio

企業では有料

### 対応データ

ファイルのアップロード（csv）

Google アナリティクス，Google広告，Google スプレッドシート

Amazon Redshift，BigQuery，Cloud Spanner，Cloud SQL，

Google Cloud Storage，Googleアドマネージャー，Googleサーベイ

SQLServer(2017)，MySQL，PostgreSQL，

Search Console，YouTube アナリティクス，Tables by Area 120

など

補足

プライベートネット内のデータベースにはアクセスできないらしい。

（要確認；情報が無い！！！）

その場合、データベースからCSV作成すればできない事もないが…。

使ってみて

データの更新頻度の最短が15分なので、リアルタイム性はいまいち。

あくまでデータ分析用と割り切る。

### チュートリアル

#### データソースの追加

 → → データソースの選択

#### 可視化（グラフ）

 → 

データを追加するように促されるので、データを追加。

または、を選択してデータを追加。

削除

レポートが表示されているところの右側にメニューが表示される（下図参照）



→ 削除を選択。

### 疑問

コントロールで選んだ値をデータ抽出の際のキーワードとして使える？ 2022-01-19

メモ

更新頻度

右側の「データ」→ データ選択 → データを管理 → 編集

（頻度が一番高くて15分）

## BigQuery

オンライン上でデータを超高速で分析するツール。

料金

|  |  |  |  |
| --- | --- | --- | --- |
| カテゴリ | 課金対象 | 料金 | 詳細 |
| ストレージ | アクティブストレージ | ＄0.023 / GB | 毎月10GBまで無料。 |
|  | 長期保存 | ＄0.016 / GB | 毎月10 GBまで無料 |
| データ取り込み | BigQuery Storage API | $0.03 / GB | 毎月最初の 2 TB まで無料 |
|  | ストリーミング挿入 | ＄0.012 / 200MB | 挿入に成功した行が対象課金となります。  最小サイズ1KBで各行が計算されます。 |
| データ分析 | クエリ（オンデマンド） | ＄6.00 / TB | 毎月1TBまで無料。 |
|  | クエリ（月定額） | ＄2,400 / 100 slots | 追加スロットは100スロット単位で購入できます。 |
|  | クエリ（年定額） | ＄2,040 / 100 slots | 追加スロットは100スロット単位で購入できます |

## Google Analytics

## Tableau

対応環境

ただし料金が高め

|  |  |  |
| --- | --- | --- |
| 製品 | 料金 | 内容 |
| Tableau Creator | $70/月 | Tableau Desktop、Tableau Prep Builder、および Tableau Cloud の Creator 1 ライセンス。 |
| Tableau Explorer | $42/月 | Tableau Cloud の Explorer 1 ライセンス。 |
| Tableau Viewer | $15/月 | Tableau Cloud の Viewer 1 ライセンス。 |

Tableau Desktop　無料で14日間試すことができる

<https://www.tableau.com/ja-jp/pricing/teams-orgs>

Tableau Server for Linux

Ubuntu 16.04 LTS および 18.04 LTS

Amazon Linux 2、

Red Hat Enterprise Linux (RHEL) 7.3 以降 (8.x でない)、

CentOS 7.3 以降、Debian 9.0 以降、Oracle Linux 7.3 以降 (8.xでない)、

その他

Microsoft Windows Server 2016、2019 (x64)

CPU が SSE4.2 および POPCNT 命令セットをサポートしていることが必要

## Grafana

### 概要

時系列のデータを扱いやすい

クエリビルダーが無い

日本語化されていない

主要な強みは、視覚化と分析であり、監視と警告の副次的な強み

特にPrometheusという時系列データに特化したDBとペアで用いられる。

（この関係はelasticsearchとKibanaの関係に似ている）

対象OS

Debian / Ubuntu

RPM-based Linux（CentOS / Fedora / OpenSuse / RedHat）

macOS

Windows

公式Dockerイメージ(Alpineベース/Ubuntuベース)

対応データベース

SQLite、MySQL（MariaDB）、PostgreSQL、SQL Server

InfluxDBなど

参考

<https://openstandia.jp/oss_info/grafana/>

### チュートリアル

#### インストール

# リポジトリの鍵設定

wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

# 補足：ok とでる事はでるが、2022年6月現在apt-keyは非推奨となっている

# ↓

# trusted.gpg.d にファイルを作る方法

wget https://packages.grafana.com/gpg.key # 適当な場所にgpg.keyをダウンロード

gpg --no-default-keyring --keyring "gnupg-ring:$dest" --import "$src"

# リポジトリ追加：OSS版

echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

# インストール：OSS版（リポジトリ追加につき、updateが必要？）

sudo apt update; sudo apt install grafana

# リポジトリ追加：enterprise版

echo "deb https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

# インストール：enterprise版

sudo apt install grafana-enterprise

Grafana Enterprise https://packages.grafana.com/enterprise/deb stable main

Grafana OSS https://packages.grafana.com/oss/deb stable main

（公式；debian）<https://grafana.com/docs/grafana/latest/installation/debian/>

（公式；windows）<https://grafana.com/docs/grafana/latest/installation/windows/>

# こんな情報もあるが？

sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"

## Metabase

### 特徴

概要

クエリビルダーなど、SQLの知識があまりない人でも使いやすい。

クラウド版は有料だが、オンプレで動かすOpen Source Edition有り。Docker版と.jar版がある。

<https://www.metabase.com/start/oss/>

Windowsサーバーでも動く

対応データベース

MySQL（MariaDB）、PostgreSQL、SQLServer、Amazon Redshift、BigQuery、

MongoDB、SQLiteなど

補足

H2での使用は非推奨らしい。

### チュートリアル

#### インストール

metabase.jarをダウンロード。

wgetで手に入れる（コンソール版の場合）

wget https://downloads.metabase.com/v0.35.3/metabase.jar

webブラウザで手に入れる（デスクトップ版の場合）

またはクライアント機でWebブラウザでダウンロード → scpやftp、sambaなどで転送

（公式）https://www.metabase.com/start/oss/

インストール

sudo apt install default-jre # javaのインストールが必要な場合（Debian）

java -jar metabase.jar

実行

Webブラウザにてlocalhost:3000。

（初期設定でのポート番号は3000になっている）

初期設定

ユーザー名、メールアドレス、パスワードなどの設定が必要

補足

H2 databaseでの使用は非推奨との事

#### ダッシュボード

サンプルデータベースが用意されているので、まずはそちらを使うと良い。

流れとしては「質問」を作成して

## Superset

## Ambient

さすがに高い！普通にVM借りれそう！

|  |  |  |  |
| --- | --- | --- | --- |
| チャネル数 | 追加チャネル数 | 月額 | 年額 |
| 8ch |  |  |  |
|  |  |  |  |
| − | 無料 | 無料 |  |
| 16ch | 8ch | 20,000円+税 | 200,000円+税 |
| 40ch | 32ch | 40,000円+税 | 400,000円+税 |

# html系

## ngrok

概要

ローカル環境のWebアプリを公開する中継サーバーを提供する。

発行されるURLを固定にする場合は最低でも$20/月かかってしまう（別の仮想サーバーを借りられる金額）。

→ 一時的に公開URLを発行する必要がある場合に向く。

<https://ngrok.com/>

# ホスティング

## netlify

### 概要

静的サイトをホスティングすることができるWebサービス

JAMstackのサービス？

公式ドキュメント（英語）：<https://docs.netlify.com/>

・サイトの公開／更新はGitコマンドを用いる。（SCPやFTPを使わない）

・

### チュートリアル(GitHubから)

#### netlifyに登録

下記urlで登録できる。emailでもアカウント作成は可能だが，Gitで使う場合が多いと思う為，このチュートリアルではGitHubアカウントを用いる。

<https://app.netlify.com/signup>

 → 

→ Get startedと表示されたページで簡単なアンケートに答える。

一番上のI’m using Netlify for だけ答えれば次の画面に進める

#### 最初のデプロイ

Start from a templateのBrowse templates → ここでは「Next.js Starter」を選択する

→ Choose a Git providerでGitHubを選択 → Authorize netlify → Deploy site

補足：通常GitHubとの連携には特定の操作が必要らしい。この方法で新規サイトを作成する場合は，GitHubとの連携は自動で行われる。

#### カスタムドメイン

Netlifyのサイトの左上にデプロイされたサイトのURLが表示される。（右図参考）

「サイト名」を変える事でサイトのURLはある程度変える事ができる。

→ Domain management → Options → Edit site name

netlifyは ttp://your-site-name.netlify.app になるので，サイト名を変える事で実質URLを変えられる

補足１：Domainmanagementにはデプロイ後に表示される2 Set up a custom domain からでも行ける  
補足２： → Domainsでも良い

自身でURLのドメイン（通常有料）を持っている場合は以下の場所でカスタムドメインを使用できる。

→ Domainmanagement → Add custom domain

#### SSL

カスタムドメインの場合のみ？

→ Domainmanagement → HTTPS →

#### 編集

GitHubのpages/index.js を編集する。

ファイル構成

重要なファイルのみ説明

/

├ pages

│ ├ index.js コンパイルされたWebページ。ここを直接編集しても良いが，

│ │ 通常，静的サイトジェネレーターを用いる（らしい；2021-11-29）

├ jsconfig.json importやエイリアスの設定ファイル？

├ netlify.toml 開発用？

#### 削除

netlify.comでサイトを選択 → Site settings → General：Danger zone → Delete this site

### チュートリアル(cliから)

Linux

sudo apt install npm # npm（Node.jsのモジュールを管理）が入っていなければ

# 補足：2021-11-29 aptからのインストールではnpmが上手く動かなかった

# 公式サイト <https://nodejs.org/ja/> からソースコードをダウンロードして，make install する必要がある。

npm -v # npmのバージョンを確認

node -v # nodeのバージョンを確認

sudo npm install netlify-cli -g # netlify をグローバルインストール。※要root権限

mkdir ~/netlify-test

参考資料

<https://qiita.com/ub0t0/items/7a31369692e03a9428f0> dockerの使用が必要？

### 詳細

#### jsconfig.json

### 情報

#### 値段

|  |  |  |  |
| --- | --- | --- | --- |
|  | Starter | Pro | Business |
| 月額 | 無料 | $19 | $99 |
|  |  |  |  |
|  |  |  |  |

### その他

.tom1ファイル？

Nuxt.js

## GitHub Pages

GitHubが運営する静的Webサイトのホスティングサービス。

## Cloudflare Pages

静的ページ構築サービス。

Gatsby + Contentfulで構築？

## Heroku

ヘロク

アプリ開発・運営用のPaas（Platform as a Service）。

## Jamstack

JavaScript ・ APIs ・ Markup の組み合わせ。

WordPressなどの LAMP stack（ Linux + Apache + MySQL + PHP）から来ているとか

# ストレージ系

## 概要

|  |  |  |  |
| --- | --- | --- | --- |
|  | 運営 | 保存形式 | 備考 |
| dropbox | Dropbox社 | オブジェクト |  |
| one drive | Microsoft | オブジェクト |  |
| S3 | Amazon | オブジェクト |  |
| Google Drive |  | ？ | 無料プランでも15GB の容量  画像の文字検索、画像の文字化などが可能 |
| box | Box, Inc | オブジェクト |  |
| Amazon Drive | Amazon | オブジェクト？ | AWS上で動作する |
| Azure File Storage | Microsoft | ファイル |  |

特徴比較

|  |  |  |  |
| --- | --- | --- | --- |
| 保存形式 | 長所 | 短所 | 備考 |
| ブロック | ・I/O操作が高速  ・スナップショット（バックアップ）の作成が容易  ・データ更新の際、ブロックのみ修正すれば良い | ・スケーラビリティーが低い  ・高コストである事が多い |  |
| オブジェクト | ・APIが提供しやすい  ・メタデータを豊富に記録できる  ・スケーラビリティー面で優れる | ・高速アクセスに向かない  ・一部のシステムとの互換性が悪いことが多い  ・ファイル操作が煩雑 |  |
| ファイル | ・様々なシステムと互換性が高い  ・ファイル操作が簡単 | ・バックアップ、復旧が煩雑  ・大量のデータを効果的に管理する事が難しい |  |

## AWS S3

## Delta Lake

スケーラブルなストレージレイヤー

あまり語られていない？？

## Minio

S3互換のオブジェクトストレージ

## OpenStack Swift

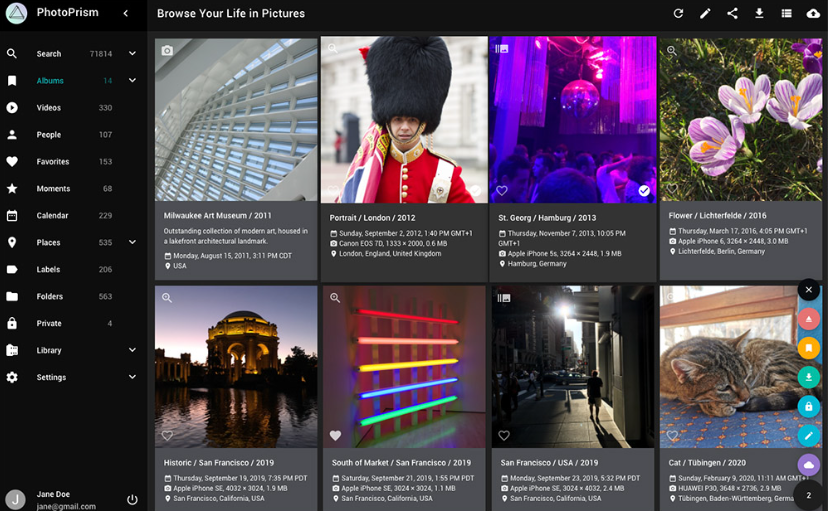
「OpenStack」の一部として開発が進められている

OpenStack：オープンソースソフトウェアのIaaS環境

## Microsoft Azure Blob Storage

## PhotoPrism

ライセンス：GPL v3

開発言語：Go

特徴

・内部でTensorflowを使っており、高度な検索が可能

　例：色で検索

　自動で分類分け

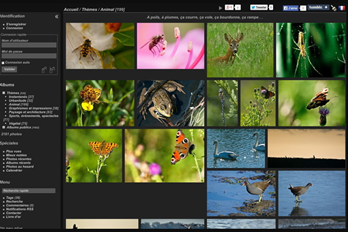
・Docker対応

・Nextcloud（ファイルサーバーアプリ）との連携可能

公式：<https://photoprism.app/features>

デモサイト：<https://demo.photoprism.app/library/browse>

## Piwigo

ライセンス：GPLv2

特徴

・検索結果をカレンダー形式で表示

・ユーザー権限管理

・各種プラグインあり

・REST API

デモ動画：<https://piwigo.org/demo>

公式（英語；特徴）：<https://piwigo.org/features>

# 静的サイトジェネレーター

## 概要

Static Site Generator（SSG）

静的Webサイトを「ビルド」によって生成できるソフトウェアの事。

|  |  |  |
| --- | --- | --- |
| 名前 | 種類 |  |
| Gatsby.js | Reactベース |  |
| Next.js | Reactベース |  |
| Nuxt | Vue.jsベース |  |
| Hugo |  | Linux機にはインストールできるらしい |

## Gatsby

JavaScriptがある程度書ける人でないと扱いが難しい。

## Hugo

静的サイトジェネレーター。

GitHub,Gitlabにデータを置いて、Netlifyでデプロイしてブログを完成させる

Linux機にはインストールできるらしい

sudo apt install hugo

## Netlify CMS

GatsbyやHugoのような静的サイトジェネレーターと組み合わせて，CMS上で記事を書きながら静的サイトの運用ができる。

## その他

### NEXT

react.jsベースのJavaScriptフレームワーク(SSR; ServerSideRender)。

### NUXT

Vue.jsベースのJavaScriptフレームワーク(SSR; ServerSideRender)。

# Linuxアプリ

## motion

Webカメラのリモート監視プラットフォーム

sudo apt install motion

# セキュリティ

## IPAのガイドライン

<https://www.ipa.go.jp/security/guide/sme/about.html>

## ホワイトペーパー

自社の

また、自社のサービスを使うユーザーに気を付けてもらいたい事などを記す。

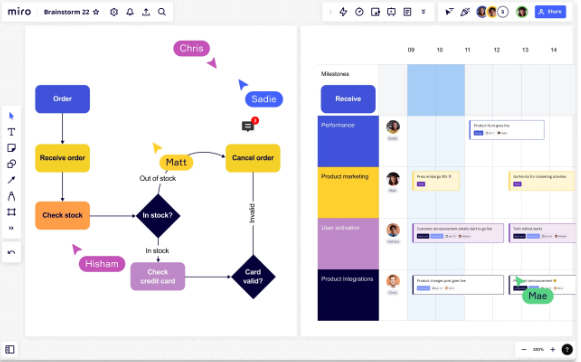
# 便利Webツール

## 業務系

### miro

https://miro.com/ja/

電子ホワイトボード



## 電気系

# その他

## CMS

最有力

Redmine OSS。プラグインが充実。多機能。

backlog プロジェクト管理に特化。UIが見やすい。

　参考；<https://note.com/pyonnpyon/n/nfd1dd29fddce>

Notion wiki機能が強い。カスタマイズ性高く、多機能。アクセス権限機能。

Stock シンプルさが売り。フォルダ形式の情報管理機能「ノート」を主軸とする。

Wrike 直感的なデザイン。Gitと連携可能。

その他有名どことその特徴

Confluence オンプレミス版あり。あまり人気がない？

DocBase

era 情報を育てる、がコンセプト。シンプルなUIが特徴。タスク管理は無い？

flouu チャット機能有り

GanttProject OSS。ガントチャートを基本とした進捗管理が中心。

Guru AI機能搭載。Gmail、Slack連携。

MediaWiki OSS。Wikipediaで仕様されている。Wiki記法と呼ばれる独特の記法。

Knowledge OSS。オンプレ対応。

Kibela ドキュメント作成機能が充実。Blog形式にも対応。アクセス権限機能。

NotePM 情報蓄積に特化。検索機能が強力。

slack チャットツール。

ScrapBook 大容量に対応。情報の自動分類→悪さする事もあるのでは？

Qiita:team 柔軟な料金プラン。エンジニア向き。

参考

https://www.stock-app.info/media/?p=13852

## VMware Horizon

複数のデスクトップ環境をサーバー上で稼働

## Go言語

## CI / CDパイプライン

Continuous Integration，Continuous Delivery

ビルド→テスト→デプロイといったソフトウェア配信プロセスの自動化。

## Pushbullet

スマホへ何かしらのイベントの発生を通知するシステム。

⇒ そもそもPushbulletというアプリが存在する？

2020年5月18日　存続の危機？

## IFTTT

イフト

IF, This, Then, Thatの頭文字から来ている。直訳すると「もし～なら…する」

Webサービス同士，またはプログラムとWebサービスを連携させて使えるWebのプラットフォーム。

IFTTT上で作成するプログラムは「アプレット」と呼ばれる。

例：

Gmailに来たメールをLINEに転送

Google Homeからエアコンを操作する

ラズパイのセンサをトリガにしてGoogleスプレッドシートの列を追加する。

（参考）<https://www.pc-koubou.jp/magazine/31960>

## Ambient

IoTデータの可視化

## Travis CI

継続的インテグレーション（CI）のサービス。

ソフトウェアの開発段階におけるテストの自動化。

GitHubと連携した自動テスト実行サービス。

## InfluxDB

## exercism

プログラム言語学習用のSNS

<https://exercism.org/>

## Notes/Domino

IBMのグループソフトウェア。電子メール，スケジュール管理，文書共有　などができる。

但し全盛期はWin XPの頃で今はほとんど使われていない。

## Redmine

オープンソースのWebツールで、業務、プロジェクトなどの管理ができる。

ガントチャート、RSSフィードなど様々なツールが使用可能。

# 用語

Ajax；Asynchronous JavaScript + XML

Webページを表示した状態のまま，別のページや再読込などを伴わずに，動的にWebページの表示内容を変更する手法。

BFF；Backend For Frontend

クライアントとバックエンドの中間に位置し双方の複雑性を吸収するような作られたサーバー

CLI; Command Line Interface

CMS；Content Management System

Dataware（データウェア）

さまざまなデータ源からデータを収集・統合・蓄積し、分析のため保管しておくシステム

読み書きは超低頻度で，分析のために一括で大量読み書きを行う。

html/2

2015年頃から使われ始めたhtmlの新規格。

ハンドシェイクを一度だけ行い，データをストリーム形式で送信する。

https://qiita.com/mogamin3/items/7698ee3336c70a482843

JAMstack

JavaScript，APIs，Markupの頭文字

LSUDs；large set of unknown developers

大多数の未知の開発者が利用する

Lift ＆ Shift

オンプレで動作する既存システムの実行環境をできるだけクラウドでもそのまま使えるように移行する事

npm；Node Package Manager

Node.jsのモジュールを管理する。

Mirakurun

デジタルTVチューナーサーバー

MFA；Multi-Factor Authentication

関連：OTP；One Time Password

MVVM

Model，View，ViewModel　Webサービスの設計思想

MQTT

軽量なWebSocketプロトコル。１つのセンサーのデータのような、多少セキュアな軽量データが得意。

よくRESTfulと対比されるが、RESTfulは幅広い人が利用するデータを公開する時に向く。

SSKDs；small set of known developers

小数の知っている開発者が利用する

REST 呼び出し

**RE**presentational **S**tate **T**ransfer；Webサービスの設計モデル

サービスのURIにHTTPメソッドでアクセスしてデータの送受信を行う。

RDF

Resource Description Framework

メタ情報を記載するデータ形式の一つ。

RSSの原型になった仕様。

SSG；Static Site Generator

静的サイトジェネレーター。有名なものにNext.js　Hugo　Gatsbyなどがある。

SSO；Single Sign On

一度ログインするだけで、複数のクラウドサービスを利用できる仕組み。

yarn

Node.jsのモジュールを管理する。

sudo npm install -g yarn

インスタンス

クラウドで提供される仮想環境の，仮想マシンの事。

エンドポイント

ネットワークの末端に接続された端末，またはそのIPアドレスやURL/URIの事。

オブジェクトストレージ

データをオブジェクトという単位で扱うストレージの仕組み。

AWSのAmazon S3（Simple Storage Service）のように，ユーザはインフラや，用途によっては開発環境も意識することなくサービスを利用できる。

スケールアップ

サーバーのメモリなどの処理能力を性能の高いものに交換／増設する事で１サーバーあたりの処理能力を向上させる。

スケールアウト

サーバーの台数を増やすことでサーバー群全体としての処理能力などを向上させる。

同等の性能アップを図る場合はスケールアップよりも低コストな事が多い。

マイクロサービス

複数の規模の小さなサービスを組み合わせてひとつの大きなアプリケーションを構成する，ソフトウェア開発の技法

クラウドネイティブ

最初からクラウドでの運用を想定して作られたサービスの事

PWA；Progressive Web Apps

モバイル向けWebサイトをスマートフォンアプリのように使えるようにするための仕組み

モバイル端末にアプリアイコンやプッシュ通知などを表示できる。

Laravel Octane

サーバーアプリのパフォーマンスを向上させる？

HoloLens

TypeScript