AWSによるクラウド入門

Tomoyuki Mano

Version 1.0, May 2020

Table of Contents

Chapter 1. アプリケーションアーキテクチャ

1.1. 基本的構成

- Webブラウザをクライアントとして，オンラインで利用するWebアプリケーションとする．
- クライアントはWebサーバーから静的コンテンツを取得し，さらに動的コンテンツをAPIサーバーから取得して，画面表示を行う設計とする．
- Amazon Web Service (AWS)のインフラストラクチャを利用した，サーバーレスアーキテクチャを採用する．

1.2. クライアント（ユーザーインターフェース）

- クライアントコンピューターはWindows/Mac/Linuxの各OSを想定し，最新版のWebブラウザ（ChromeおよびFireFox）に対応することを必須とする
- モバイル端末（iOS/android）への対応は特に考慮しない．が，画面の大きさに関わらずページコンテンツを表示するためのリスポンシブな画面設計は必須とする．

表示言語は基本的に英語とする．英語の文法やスペルは，必要に応じてGrammarlyまたはGoogle翻訳を用いてチェックを行う．ネイティブスピーカーによるチェックは必須ではない．

ユーザーインターフェースに関わる要件として，

- レスポンス高速化のため，クライアントサイドでHTMLを構築する Single Page Application (SPA) 技術を利用する
- Webページの作成には nuxt.js/Vue.js をフレームワークとして用いる．

CSSやUIを構築するフレームワーク/ライブラリー（boostrap, jQuery, Semantic UI, Vuetifyなど）に関しては特に指定を行わない．受託企業との協議により決定する．

1.3. サーバー

一般的な事項として，

- APIはRESTfulな設計とする．
- APIのハンドリング（Lambda）に使用するプログラミング言語は基本的にPython 3.7とする．
- データ交換は原則としてJSONを用いる．
- Webサーバーは原則としてHTTPSによる通信を行うものとする．

AWSが提供する以下のサーバーレスサービスを用いることで，アプリケーションを構築する．

- 静的コンテンツの配信には，WebサーバーモードのS3を用いる．
- APIの構築にはLambdaとAPI Gatewayを用いる．Lambdaで動作するプログラミング言語は基本的にPython 3.7を使用する．
- ユーザーの持つ容量の大きいデータ（画像など）はデータ格納用のS3に格納する．
- アカウント情報，各種メタデータ，パラメータ，セッション情報などはDynamo DBに格納する．
- ユーザーの認証・サインアップ・管理等はCognitoを用いる．
- 事前処理の計算の実行にはECSを用いる．実行は指定された仕様に従ったDocker containerとして提供されるものとする．
- 解析レポートの作成にはECSを用いる．実行は指定された仕様に従ったDocker containerとして提供されるも

のとする．

- Webサービス全体はCloudFront以下に配置し，HTTPS通信を原則的に用いる．
- ドメインの取得にはRoute 53，SSL/TLS証明書の発行にはACMを利用する．

AWSのアーキテクチャの概略図（案）を下に示す．

[architecture] | architecture.png
Figure 1. AWS architecture （案）

1.4. AWSスタック構築

AWS上に展開するアプリケーションスタックは AWS CDK を用いて構築する．AWS CDK によりCloudFormation テンプレートを生成し，自動で（設定ファイルの編集などは除く）デプロイ・アンデプロイが実行できるようにする．AWS CDKで使用する言語はPython 3.7とする．

ドメインの取得や証明書の発行（Route 53 と ACM を利用）など，CDKへの統合が難しいあるいは必ずしも必要でない部分は，CDKの外で設定が行われるものとし，CDKアプリケーションへのパラメーターとして渡されるものとする．

1.5. 課金機能の実装

1.5.1. クレジットカード

StripeとのAPI連携・接続を行う．詳細は，受託開発企業との協議により決定する．

1.5.2. 請求書払い

misocaなどの請求書発行代行サービスとの連携・接続を行う．代行サービスの選考もふくめ，詳細は，受託開発企業との協議により決定する．