インフラ構築手順書

[1. 業務要件 1](#_Toc2078874937)

[2. 機能・非機能要件 2](#_Toc1912470744)

[機能要件 3](#_Toc180993690)

[非機能要件 3](#_Toc877750880)

[画面イメージ 4](#_Toc1061326577)

[3. インフラ設計 5](#_Toc730837635)

[アーキテクチャ 6](#_Toc983522605)

[テクノロジースタック 6](#_Toc253645165)

[ソーステクノロジースタック 6](#_Toc71035278)

[4. インフラ構築手順 7](#_Toc2088531899)

[作業者情報 8](#_Toc476884891)

[作業実績 8](#_Toc464341246)

[構築手順 8](#_Toc1090637761)

## 業務要件

産業用ドローンの需要拡大に伴い、各分野で異なるニーズに応えるためのカスタマイズ要件が求められています。しかし、従来のパッケージ型生産システムでは対応が困難であると判明したため、新たに独自のドローン生産システムを開発することが決定されました。本プロジェクトでは、その生産システムの一部である在庫管理システムを構築します。

本手順書では AWS App Runnerの構築、ロギングとモニタリングの設定、 インフラコスト管理の設定を行います。

## 機能・非機能要件

### 機能要件

1. 部品情報管理
   * 部品の一覧表示と詳細情報の閲覧
   * 部品の追加、編集、削除機能
   * 部品カテゴリーの設定と管理
2. 在庫管理
   * 部品の入庫および出庫の記録
   * 在庫数量の管理
   * 在庫の閲覧と検索機能（カテゴリー、部品番号、在庫状況などでのフィルタリング）
3. 注文処理
   * 部品の注文と受け取りのトラッキング
   * 注文履歴の表示と管理
   * 注文ステータスの更新
   * 在庫が一定数以下になった場合の自動発注
4. ユーザー管理とアクセス制御
   * 管理者と一般ユーザーの役割の設定と管理
   * ユーザーごとのアクセス権限の設定（閲覧、編集、削除など）

### 非機能要件

1. パフォーマンス要件
   * システムの応答時間：ユーザーの要求に対するシステムの応答時間は2秒以内であること。
   * 同時アクセスのサポート：システムは最大100人の同時アクセスをサポートすること。
   * データ処理速度：在庫データの更新や検索などのデータ処理は高速かつ効率的に行われること。
2. セキュリティ要件
   * アクセス制御：ロールベースのアクセス制御（RBAC）を実装し、ユーザーごとに適切なアクセス権を付与すること。
   * データの暗号化：重要なデータはトランジットおよびアットレストで暗号化すること（AES256など）。
   * ログと監査：システムへのアクセス、変更、操作などのアクティビティをログとして記録し、適切に監査可能な形式で保持すること。
3. 可用性と耐障害性
   * システムの可用性：システムは99.9％の可用性を維持すること。
   * バックアップと復元：定期的なデータバックアップと災害復旧計画を実施し、データの損失を最小限に抑えること。
4. 拡張性と保守性
   * システムの拡張性：将来的なシステムの拡張性を考慮し、新しい機能やユーザーの追加が容易に行えるアーキテクチャを採用すること。
   * コード品質とドキュメント：コードは適切にコメントされ、保守性が高く、新しい開発者が迅速に理解できるようにすること。
5. ユーザビリティ
   * インターフェースの直感性：ユーザーが簡単に操作できる直感的なインターフェースを提供すること。
   * エラーハンドリング：エラーが発生した場合には、ユーザーに分かりやすいエラーメッセージを表示し、適切な対処方法を提供すること。
6. コスト最適化
   * インフラコストの最適化：インフラのコストを最適化することによりサービスの持続可能性を高めること。

### 画面イメージ

Fig.1 画面イメージ

## インフラ設計

### アーキテクチャ

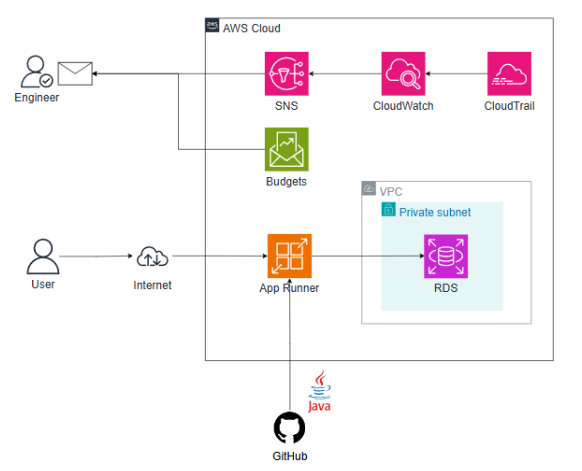


Fig.2 構成図イメージ

### テクノロジースタック

* サーバ
  + App Runner
* データベース
  + RDS

### ソーステクノロジースタック

* Java 11
* Spring 2.7.15
* Docker

## インフラ構築手順

### 作業者情報

氏名：佐藤

連絡先：satoushouta1205@gmail.com

### 作業実績

工数：7h

結果：正常完了

### 構築手順

* AWSリソースの命名規則

下記の命名規則に従って構築する。

{sysname}-{env}-{user}-{service}-{予備}

|  |  |
| --- | --- |
| 要素 | 詳細 |
| sysname | 固定値として **drone** を使用 |
| env | 固定値として **dev** を使用 |
| user | （個別）**IAM user名** |
| service | （個別）**対象サービス** |
| 予備 | 一意にできない場合に使用 |

* タグの命名規則

作成したリソースに以下の命名規則でタグ付けする。

個別のリソースに係るコストや削除漏れなどを防ぐ他、

作成した人物やプロジェクト名の可視化などの効果がある。

|  |  |
| --- | --- |
| キー | 値 |
| Cost | drone\_**IAM user名** |
| Project | （固定）**infra-course-drone** |
| Name | **リソース名** |
| createdBy | （個別）**IAM user名** |

1. AWS App Runner の構築
   1. App Runnerで「サービスの作成」を行う
   2. ソースおよびデプロイ
      * 「ソースコードリポジトリ」を選択してGitHubと連携する
      * GitHub連携の「ソースディレクトリ」は「/dev」を指定する
   3. デプロイ設定
      * 「自動」を選択する
   4. 構築を設定
      * ランタイムは「Corretto 11」を選択する
      * AppRunner 上ではビルドコマンドとスタートコマンドを設定する

|  |
| --- |
| $ mvn clean package  $ java -Xms256m -jar target/dev-0.0.1.jar |

1. CloudWatch ダッシュボードの作成
   1. CloudWatch >ダッシュボードから「ダッシュボードの作成」を選択し、ダッシュボード名を入力して作成する
   2. 「ウィジェットの追加」から、下記の各メトリクスグラフを追加する
      * 線またはスタックされたエリアを選択し、メトリクスを設定する
      * 「ウィジェットの追加」を繰り返し、複数のウィジェットを追加する

AppRunner >インスタンスメトリクス＞ CPUUtilization

AppRunner >インスタンスメトリクス＞ MemoryUtilization

ログ > ロググループメトリクス> IncomingLogEvents

AppRunner >サービスのメトリクス＞ Requests

AppRunner >サービスのメトリクス＞ 2xxStatusResponses

AppRunner >サービスのメトリクス＞ 4xxStatusResponses

AppRunner >サービスのメトリクス＞ 5xxStatusResponses

AppRunner >サービスのメトリクス＞ RequestLatency

* 1. ログに関する情報をダッシュボードに掲載する
     + 「ウィジェットの追加」からデータ型「ログ」を選択、

クエリを下記に変更しダッシュボードに追加する

|  |
| --- |
| fields *@timestamp*, *@message*, *@logStream*, *@log*  | filter *@message* like /PAUSED/ or *@message* like /pause/  | sort *@timestamp* desc  | limit 100 |

1. CloudWatch Alarmの作成
   1. CloudWatch > すべてのアラームから「アラームの作成」を行う
   2. 監視するCPUメトリクスを選択しアラートを作成する
      * CPUの閾値設定は以下とする

閾値：80%より大きい

データ取得間隔：5分

アラームを発生させるデータポイント数： 3

評価期間数：３

* + - SNS(Simple Notification Service)を利用し、自身のメールへ通知が来るように設定を行う
    - 作成後、「AWS Notification - Subscription Confirmation」という件名でAWSより認証メールが届くため承認する

1. CloudTrail での監査
   1. CloudTrail で特定ユーザーの利用ログを確認する
      * CloudTrail > イベント履歴 から、[ルックアップ属性] ： ユーザー名 として、利用ログを確認する
2. インフラコスト管理の設定
   1. AWSの利用状況を監視するために、AWS Billing and Cost Management を利用する
   2. 「請求とコスト管理」 > 「コスト分析とレポート」 > 「Cost Explorer」でレポートを作成、保存する
   3. AWSの予算を設定して、予算超過のタイミングで管理者にメールでアラートを通知する
      * 「請求とコスト管理」 > 「予算」 でアラートを月次コスト予算を設定する

以上