4. 検証の実施内容

3章までの検討結果を踏まえて、IaCコード生成の実装検証を実施する。 実装検証では作業者のスキル別に3ケースを想定し、それぞれケースでIaCコードを生成する最適な方法を作成する。

4.1. 対象範囲

- **システムを構築するプラットフォーム**：AWS
- **開発言語**：Terraform
- **システム構成の概要**：
    - コンテナ上で実行するWebアプリケーションを作成。
    - ECS上でFagate上で実行する。

4.2. 対象ケース

- ケース①： AWSの設計技術、Terraformコーディングの技術がある場合
    - 作業者がTerraformのリソース、設定を指示。
    - 生成AIが指示に従いTerraformコードを生成。
    - 作業者が生成されたTerraformコードを評価する。
- ケース②： AWSの設計技術のみある場合
    - 作業者がAWSのリソース、設定を指示。
    - 生成AIが指示に従いTerraformコードを生成。
    - 作業者はTerraformコードを実行してAWSに環境を作成し、AWS上の設定が指示どおりに作成されるか評価する。
- ケース③： AWSの基礎知識のみある場合
    - 作業者がアプリケーションの機能、AWSのサービスを指定。
    - 生成AIが指示に従いTerraformコードを生成。
    - 作業者はTerraformコードを実行してAWSに環境を作成、アプリケーションが正常に動作するか確認する。

4.2. コード生成方法の評価観点

生成方法は次の観点を考慮して最適な方法を作成する。

#	観点	説明
1	作業時間	作業者の手動でTerraformのコーディングを行った場合に比べて作業時間が短縮できること。
2	完成度	生成したコードから環境の構築がスムーズに行えること。
3	再現性	コード生成の再現性が高いこと、コード生成を繰り返し実行したときに同じコードが生成されること。
4	適合性	生成したコードはセキュリティや冗長化など、必要な要件を満たすコードになること。
5	可読性	生成したコードは読みやすく、人の手による継続的なメンテナンスが可能なこと。