2. 背景と目的

2.1. 背景

検証を計画した時点の背景として、生成AIの性能向上により特にシステム開発の分野においての活用範囲が急速に拡大している。 アプリケーション開発の活用事例が多数紹介されるなか、IaCコードのベンチマーク「DPIaC-Eval」では生成AIを活用したコード生成ではIaC特有の課題が指摘されている。

2.2. 目的

- IaC の開発に生成AIを活用する際の「効果的な使用方法」を検証する。
- 生成AIを使用することの「有効性（どの程度有益か、どの条件で機能するか）」を検証する。
- 上記目的の達成のため、前提の異なるケース①〜③で比較検証を行う。

検証の実施内容

- IaCのコード生成に使用する生成AIの基盤モデル、開発ツールを選定する。
  基盤モデルの選定にあたって、生成AIのプログラミング性能を評価したベンチマークを対象に、基盤モデル選定の基準とするベンチマークを選択する。
- 検証を通じて、有効なIaCコードを生成するための方法を作成する。主に生成AIに与える指示をフィードバックのタイミングを調整する。

2.3. 範囲と対象外

- 範囲: 生成AIを活用したIaCコードの生成。Terraformのコードを生成し、AWS上に環境を構築する。
- 対象外: 生成AIの利用料と作業量の削減から算定される費用対効果。

2.4. 想定ケースの位置づけ

- 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、AWS上の設定が指示通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。

2.5. 成果物

ケース①、②、③それぞれにおいて環境構築が成功するまでの手順を作成する。
生成AI活用の有効性の観点からケース①、②、③の比較を行う。