IaCにおける生成AI活用の可能性

1.1. 背景・目的

クラウド推進部40期目標としてIaCの実装検証を行った。 2024年後半からの時代背景として生成AIを活用したシステム開発のためのツールが急速に発展していたため、 IaCの実装検証でもTerraformのコーディングに生成AIを活用することを検討した。

1.2. 検証内容

生成AIを使用してTerraformのコード生成を実施する。

- 使用する基盤モデル、開発ツールは公開されている活用事例や基盤モデルのベンチマークを参考にして選定する。
- コード生成のために生成AIに与える指示などを含む生成方法は、複数の方法を検証する。検証した方法をメリット、デメリットを評価する。
  生成方法は、次の3ケースを想定する。
  - ① AWSの設計技術、Terraformコーディングの技術がある場合
  - ② AWSの設計技術のみある場合
  - ③ AWSの基礎知識のみある場合

1.3. 検証結果

1.3.1. 基盤モデル、ツールの選定結果

開発ツール

Cursor

基盤モデル

Claude 3.7 Sonnet

生成方法の指針

選定した開発ツール、基盤モデルを使用する場合は以下の指針に従って生成を行う。

- コード生成指示プロンプトと同時に、設計書とコーディングルールを与える。
- コード生成は開始から完了までを一回の指示で実施する。
  - コード生成指示から作業タスクへの分割、生成完了までの反復実行は開発ツールにより自動で実行する。
- コード生成が失敗する場合はプロンプト、設計書、コーディングルールのいずれかを修正して再実行する。

1.3.2. 検証より得られた考察

1.3.2.1. 開発プロセス

検証した3ケース全てにおいて、コードの生成からAWS上の環境生成完了までの開発プロセスは同一のプロセスになる。
以下に実施したプロセスを記載する。

- （1） 設計
  想定ケース毎に担当者の技術に応じた設計書を作成する。
  設計書の記載ルール、コーディングルールなどを作成する。
- （2） コード生成
  「(1) 設計」で作成した設計書とルールを与えて生成AIにTerraformのコードを生成させる。
- （3） IaCコード解析、レビュー
  生成されたTerraformのコードをチェックして、改善箇所を指摘する。指摘がある場合は、設計書とルールを修正して「(2)コード生成」を再実行する。
  Terraformのコードのチェックには以下の機能を使用する。 - ① コード解析ツール - ② 生成AIによるレビュー - ③ 担当者によるレビュー
- （4） 構築
  Terraformのコマンドを実行してAWS上に環境を構築する。
  生成AIはTerraformのコマンドを監視する。エラーが発生した場合はTerraformコードを修正してコマンドを再実行する。
- （5） 構築結果確認、動作検証
  担当者により、AWS上に構築された環境を目視確認と、システムの動作確認を行う。
  不具合がある場合は設計書とルールを修正して「(2)コード生成」から再実行する。

1.3.2.2. 開発方法毎のメリット、デメリット

（考え中）

1.3.2.3. 作業時間の短縮効果

- (1) Terraformのコーディングを行う作業時間は大幅に短縮される。
- (2) Terraformのコーディング以外の作業時間はほぼ増減なし。

1.3.2.4. 品質に与える影響

- (1) 以下の効果により、メンテナンスしやすい設計が保たれるようになる。
  - ① 設計書を修正してTerraformコードを生成する手順が守られる。
  - ② 設計書修正から環境の再構築のサイクルが早い。

1.3.2.4. 技術者の育成に与える影響

生成AIを活用してコード生成を行うと、Terraformの基礎学習を行わずともIaCを用いた学習プロセスが体験できる。

このことは、IaCに関する理解を促進するためには有効であるが、一方でTerraformに関する理解が不足したまま構築を行う危険がある。

対策として、開発リーダーによるメンバーの理解度のチェックを行う必要がある。

1.4. まとめ

IaCに生成AIを活用することで作業時間の短縮効果がある。また、技術者の育成にも一定の効果はあるが生成AIに依存するため必要な技術が習得できない危険がある。

生成AIの特徴をよく理解して使用することが重要となる。