1B-05

手続き型IaCを利用したフィーチャベースクラウドプロビジョニング

味藤 未冴来 † 大原 貴都 † 清水 遼 † 鹿糠秀行 † 富坂 稔 ‡
(株) 日立製作所 研究開発グループ † (株) 日立製作所 サービス&プラットフォームビジネスユニット ‡

1. はじめに

パブリッククラウドでは，Infrastructure as Code (IaC) を利用して環境を定義し，自動プロビジョニングできる [1]．既存プロジェクトのIaCを新規プロジェクトで利用することで，新規プロジェクトの必要工数削減が期待される．一方で，パブリッククラウドではインフラ層はマネージドサービスとして提供され (負荷分散装置など)，様々な技術バリエーションが存在する [2]．したがって，同じアーキテクチャを採るプロジェクト間であってもプロジェクト毎に採用技術の違いがあり，採用技術の違いに応じたIaCの修正が多数必要となる．

本稿では，IaC再利用時の修正を少数に留めるため，次の3点を特徴とするクラウドプロビジョニング方法を提案する．1) パブリッククラウドの技術バリエーションを，Web3層などのアーキテクチャに基づいたフィーチャモデル (FM) でモデリングする．2) 手続き型IaCを活用し，フィーチャと対応してIaCをモジュール化する．3) パラメータファイルによって各プロジェクトでのフィーチャ (採用技術) とIaCを対応づける．

以降の章で提案法を説明し，ケーススタディを通じて提案法の有効性を示す．

2. クラウドプロビジョニング方法

図1に，パブリッククラウドとしてAWSを対象とした提案法の概要を示す．パブリッククラウドの技術バリエーションをFMで定義し (図1 (a))，プロジェクト毎で選択した採用技術 (図1 (b)) をパラメータファイルを通じてモジュール化されたIaCに対応付け (図1 (c)-(d))，モジュール化されたIaC(図1 (e)) によって生成されたプロビジョニングツール (図1 (f)) を利用してクラウド環境をプロビジョニングする (図1 (f)-(g))．

2.1 アーキテクチャに基づいたFM

ソフトウェアプロダクトライン [3] で提唱されているFMは，プロジェクト間での採用技術が多く共通しているため，本稿ではアーキテクチャ毎に構築する．また，本稿では必須と可変の2種類のフィーチャ定義する．必須フィーチャは図1 (b) の “Web Server” ようなプロジェクト間で共通の採用技術である．可変フィーチャは図1 (b) の “Load Balancer” ようなプロジェクト間で異なる採用技術であり，図1 (b) の “selective” のように “route” フィールドによって定める．ユーザはFM上で可変フィーチャを選択して構築環境を決定でき，必須フィーチャによってプロジェクト間共通の採用技術の構築漏れを防ぐことができる．

2.2 手続き型IaCのモジュール化

採用技術の違いによるIaC修正範囲を少数に留めるため，Web3層アーキテクチャにおける “Web Server” など，プロジェクト間で多く共通する採用技術単位でIaCをモジュール化する．本稿では，AWS Cloud Development Kit (AWS CDK) [4] を利用してIaCをモジュール化する．AWS CDKはPythonのような手続き型汎用プログラミング言語が利用可能であるため，現在広く用いられている宣言型ドメイン固有言語であるAWS CloudFormation (AWS CFn) [5] を直接記述する場合と比べて，容易にモジュール化できる．

2.3 各プロジェクトでの採用技術とIaCの対応付け

パラメータファイル (図1 (c)) はモジュール化されたIaC(図1 (e)) と同じモジュールを持つ．パラメータファイルの各モジュールは，モジュール化されたIaCへ入力するキー名 (図1 (c) の “elb”) を持ち，フィーチャは固有のタグ名 (図1 (c) の “lb”) を持ち，各プロジェクトのフィーチャ (採用技術) とIaCを，キー名とタグ名で対応付ける (図1 (c)“lb”を図1 (d)“True”へ)．

3. ケーススタディ

3.1 シナリオ

本シナリオは，図1のように既存プロジェクトで提案法を構築済みであり，図1(b)“Web Server”の採用技術が異なる新プロジェクトへ図1を適用する場合を想定する．既存プロジェクトでは，クライアントからwebサーバへの通信を2つの地理的に離れたデータセンタ (AWSにおいては，AZ(Availability Zone) と呼ぶ) にAWSのマネージドサービスの1つであるApplication Load Balancer (ALB) を利用して負荷分散している．新プロジェクトではAZをまたぐ通信遅延を低減するため，単一のAZ内での “Load Balancer” 利用を考えているが，ALBは単一AZで動作できない．したがって，

Feature-based Cloud Provisioning with Procedural Infrastructure as Code,
Misaki Mito†, Takatoshi Ohara†, Ryo Shimizu†, Hideyuki Kanuka†, Minoru Tomisaka‡
† Hitachi, Ltd. Research & Development Group
‡ Hitachi, Ltd. Services & Platforms Business Unit

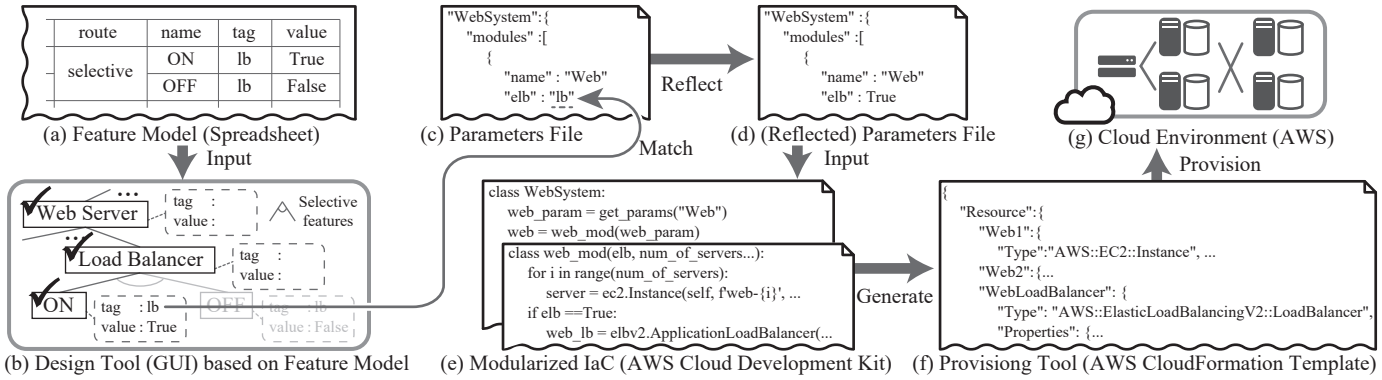

図 1: 提案法の概要.

単一の AZ で動作可能な別の AWS のマネージドサービスである Network Load Balancer (NLB) に，“Load Balancer”を既存プロジェクトから変更して利用することにした.

3.2 提案法におけるツール修正手順

前節のシナリオにおいて図 1 で示す提案法がどのように修正されるのかを図 2 に示す.

まず，パラメータ “azs”を IaC(図 2 (e)’) の “web_mod”に，キー “azs”をパラメータファイル (図 2 (c)’) に加える．“web_mod”内では “if 文”を追加し，“web_lb”を “NetworkLoadBalancer”に切り替える.

次に，“Load Balancer”フィーチャ (図 2 (b)’) と同様に，FM(図 2 (a)’(b)’) の “Web Server”フィーチャ以下に AZ 数を決定する “AZs”フィーチャを追加する．“AZs”フィーチャは, それぞれの AZ 数を “value”フィールドで定めた “1” / “2”フィーチャを持つ．なお，“1” / “2”フィーチャの “route” / “tag”フィールドは同じ値を持つ (図 2 (a)’ ではそれぞれ “selective”と “azs”).

3.3 評価

提案法では，全体で約 40 行のコード修正を必要とした．一方で，従来通りに AWS CFn を直接記述する場合では約 150 行のコード修正を必要とした．これから，3.1 節のシナリオの場合に，提案法は従来法と比較して少数の修正で実現できることがわかった.

4. おわりに

本稿では，プロジェクト間での IaC の再利用時の修正を少数に留めるため，手続き型 IaC を用いるフィーチャベースのクラウドプロビジョニング方法を提案した．ケーススタディを通じて，提案法はツール全体の少数の修正のみでクラウドプロビジョニングできることを示した．今後は，様々な観点でモデル化し生成される複数の FM をバージョン管理し，FM の再利用性向上を計画している.

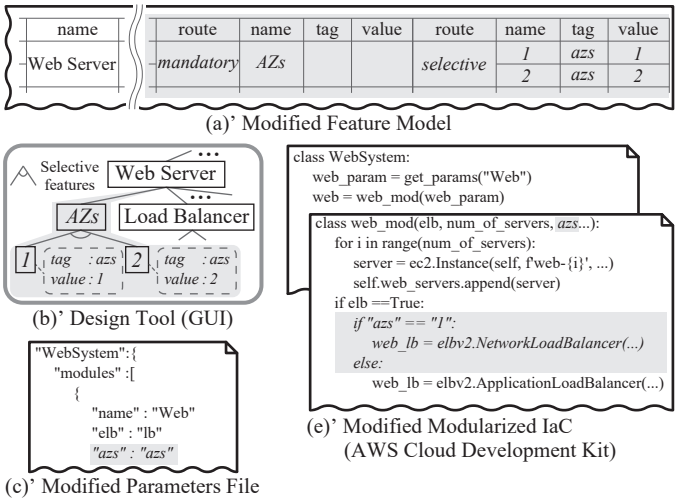

図 2: 任意の AZ 数に応じて，負荷分散装置を Application Load Balancer (ALB) から Network Load Balancer (NLB) に切り替え可能とする修正例．灰色に塗りつぶされた箇所が修正される箇所.

参考文献

[1] Campbell, B.: *The Definitive Guide to AWS Infrastructure Automation*, Apress, Berkeley, CA, 1 edition (2020).

[2] Eisa, M., Younas, M., Basu, K. and Zhu, H.: Trends and Directions in Cloud Service Selection, *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pp. 423–432 (2016).

[3] Pohl, K., Böckle, G. and van der Linden, F.: *Software Product Line Engineering*, Springer-Verlag Berlin Heidelberg, 1 edition (2005).

[4] Amazon Web Services, Inc.: AWS Cloud Development Kit. (accessed 2 Oct., 2021).

[5] Amazon Web Services, Inc.: AWS CloudFormation. (accessed 2 Oct., 2021).