SRA の技術情報誌
GS LetterNeo
Vol.198　2025 年 4 月

Microsoft Azure を用いた コードレス運用自動化システム

大島 聡一郎

アドバンストクラウドエンジニアリング事業部

はじめに

近年、運用業務の自動化において「ノーコード開発」「ローコード開発」といったキーワードがよく聞かれるようになり、コードレスで開発が行えるような製品も数多く販売されています。

一見すると簡単に導入できそうに見えますが、実際の現場では予算や機能制限、セキュリティといった現実的な問題で導入ができない現場も少なくありません。

本記事では、さまざまな制約でコードレス開発ツールを導入できない環境でもコードレス開発を行えるよう、Microsoft Azure 上に構築したコードレスな運用自動化システムのアーキテクチャをご紹介します。

なお、本記事は関係者の許諾を得て執筆しております。

当初の目標

このシステムを構築した現場は、開発担当者と運用担当者に異なる人員が配置されていました。この現場では、クラウドのコンソールにログインしなくても設定変更が行えるような、一部の運用を自動化できる機能が実装されていましたが、開発者が作ったツールを運用者が利用するという一方向の開発となっていたため、運用現場のニーズに合わない機能もたくさんありました。

そのため、開発経験やスキルを持たない運用者でも簡単に開発を行えるよう、コードレスで開発できるシステムを用意して、運用者が主体となって現場が必要な機能を素早く導入できる環境を目指しました。

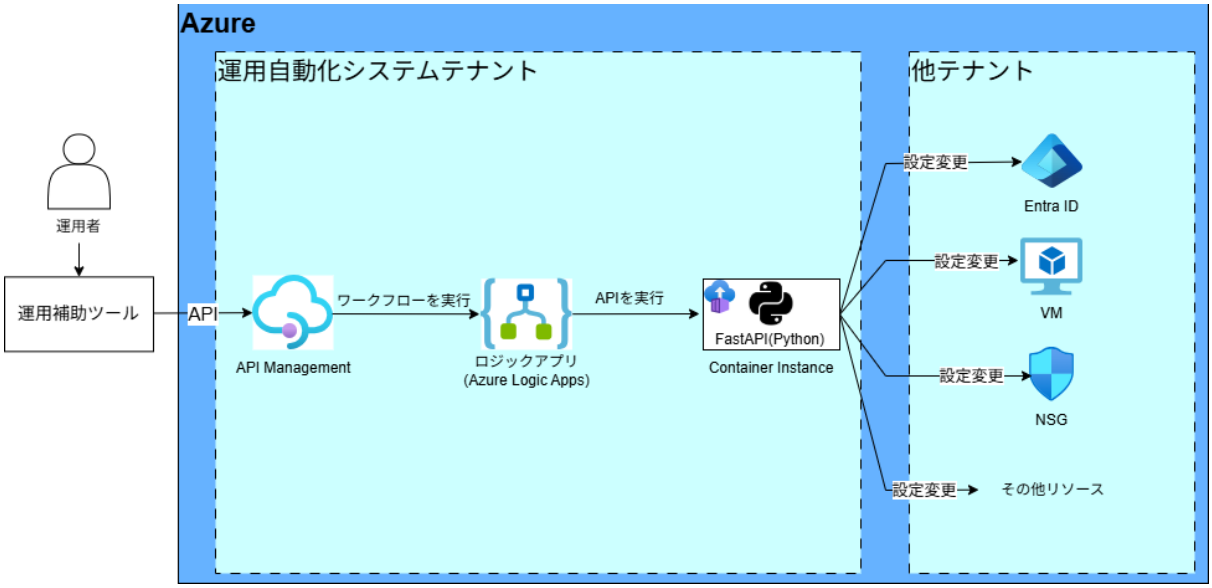
要件

本システムの開発の要件は以下です。

1. Azure 上に構築してほしい
2. Azure で提供されている機能を利用してほしい
3. 開発経験のない運用者でも運用を自動化する機能を開発できるよう、ビジュアルエディタが欲しい
4. スクリプトを実行する場合はコンテナを利用してほしい
5. 現場で利用している運用補助ツールから、自動化機能を実行できるようにしてほしい
6. できるだけランニングコストを抑えてほしい

アーキテクチャ

以下構成図です。



運用補助ツールから運用自動化システムの API を実行し、他サブスクリプションや他テナントのリソースの設定変更を行う処理の流れとなっています。

要件 1～3 を満たすため、Azure が提供しているビジュアルエディタを備えたサービスとして、ロジックアプリ（Azure Logic Apps）を採用しました。

ロジックアプリとは、Azure が提供しているビジュアルエディタを備えたワークフローの作成や実行が行えるサービスです。

ループや条件分岐などの基本的な機能のほかにも、VM の起動/停止や設定変更など、Azure が提供している一部のサービスに対する操作を行うジョブも標準で用意されていますが、本構成では権限の関係で利用できなかったため、要件 4 に従いコンテナ上で Python を実行してリソースの操作を行っています。なお、要件 6 を満たすため、コンテナは元々 Python 実行のタイミングだけ起動する予定でしたが、最も安いプランを利用していた影響で起動時間がまちまちだったり、ときには起動に失敗することもあったので、常駐するタイプを利用しています。

また、要件 5 を満たすため、API Management を利用し、外部から API でロジックアプリのワークフローを実行できるようにしています。

この構成で、円安の影響を受けてなお月 1 万円台の利用料金（ほぼコンテナサービスの利用料金）に収まっています。

構築時に判明した課題

いざ構築してみると様々な課題にあたりました。その中でも特に大きな問題だったのが、開発経験がないとロジックアプリの開発をするのが難しいという点です。もちろんロジックアプリ自体は素晴らしい機能で、スクリプトを書くよりもはるかに楽に実装できますが、ワークフローを実行する際のトリガーのスキーマ設定や、ジョブがエラーとなった際の分岐など、開発のスキルやロジックアプリに特化した知識が必要となり、当初の目標である、開発経験のない運用者が中心となって開発を進めるのが難しいことがわかりました。

しかし、スクリプト開発などよりも開発難易度は低く、学習を行えば開発が行えるようになること、また、ロジックアプリにはワークフローから別のワークフローを呼び出す機能があり、開発に知識が必要な処理をワークフローにまとめることで開発難易度を下げられることから、構築を続行しました。

コードレス開発システムの使用感

運用者に引き渡せていないため、開発経験者から見た使用感となってしまいますが、一度作成したワークフローや Python を使いまわせるため、開発速度が大幅に上がりました。

また、どのタイミングでどの処理を行うかを視覚的に表現できるため、運用者との認識合わせが行いやすく、要件の食い違いによる手戻りも防ぐことができています。

課題としては、開発が進むにつれてワークフローや Python スクリプトが増えていき、二重開発のリスクがあるため、開発した機能の管理の徹底が必要となります。

今後の展望

今後は開発難易度が高い処理のワークフロー化を進め、当初の目標である開発経験のない運用者が中心となって自動化機能を開発できる環境の達成を目指しています。

おわりに

本システムは表面的には”コードレス”な見た目をしていますが、実態としては多くの技術的工夫とスクリプトで支えられた自動化基盤です。

目標である運用者中心の開発までは達成できていませんが、開発時間の短縮など目に見える効果は出ているので、今後は運用者が自ら改善を進められる環境が整うよう、段階的な改善を進めていきたいと考えています。

GSLetterNeo Vol.198
2025 年 4 月 20 日発行
発行者　株式会社 SRA 技術本部 先端技術研究室

編集者　熊澤努　方 学芬
バックナンバー https://www.sra.co.jp/public/sra/gsletter/
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8