Engineer

Bicep

インフラを宣言するためのAzure独自の言語

どのようなものか

Bicepは、Azureインフラをテキストとして記述するための、Microsoftの言語です。どのリソースが存在し、何という名前で、どう設定され、互いにどう接続するかを記述します。Bicepファイルには、手順を並べません。最終的な状態を示すと、Azureの管理要求が通る層、Azure Resource Managerが、そこへ至るために何を作成し、変更し、そのまま残すかを判断します。「Bicepとは」と尋ねる人が知りたいのは、多くの場合、Azureポータルでのクリック操作に代わるかどうかです。代わりになり、そのことに価値の大半があります。

その仕組みは、Azureがすでに理解できるものを、より書きやすくしたものです。Resource Managerは、以前からJSONで書いたデプロイテンプレートを受け付けています。必要なことをすべて表現できますが、読みにくく、レビューはさらに困難です。デプロイ時には、Bicepのコマンドラインツールが `.bicep` ファイルをそのJSONテンプレートへ変換し、Resource Managerが、手書きのJSONとまったく同じようにデプロイします。Microsoftは、リソースプロバイダーが導入した時点から、プレビューを含むすべてのリソースタイプとAPIバージョンにBicepが対応すると述べています。そのため、プラグインの更新を待たずに、新しいサービスを使えます。既存のJSONテンプレートをBicepへ逆コンパイルすることもでき、古い環境の移行は、書き直しではなく機械的な作業になります。

小さなファイルを見ると、その形が分かります。まずパラメーターを置き、次に各リソースを記述します。それぞれに、ファイルの他の部分から参照できるシンボル名を付けます。

```
param env string
param location string = resourceGroup().location

resource store 'Microsoft.Storage/storageAccounts@2023-05-01' = {
  name: 'st${env}${uniqueString(resourceGroup().id)}'
  location: location
```

```
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'
}
```

別のリソースが `store.id` を参照すると、Bicepは、そのリソースがストレージアカウントの作成を待つ必要があると推論します。そのため、人が順序を指定する必要はなく、他に依存しないものはResource Managerが並列にデプロイします。一つのファイルから別のファイルをモジュールとして呼び出せるため、「ネットワークルールを備えたデータベース」を一度記述して、必要な場所すべてで再利用できます。環境ごとに異なる値は、別の `.bicepparam` ファイルに置きます。Microsoftは、Bicepが無償のオープンソースであり、汎用プログラミング言語を目指していないことも明記しています。ループ、条件、関数はありますが、いずれもリソースを記述するためのものです。

他の選択肢とは、どう違うのでしょうか。ARM JSONとは同じエンジンを使い、ソースがはるかに短くなります。Terraformとの主な違いは、対応範囲と状態の管理です。Terraformは多くのクラウドに対応し、独自の状態ファイルを保持します。一方、BicepのAzure Resource Managerを使う方式では、Azureがデプロイ状態を保存するため、独自の状態ファイルを保持しません。Bicepの拡張機能は、KubernetesやMicrosoft Graphなど、Resource Managerの外のリソースにも対応します。TypeScriptやGoなどの汎用言語でインフラを記述するPulumiに対しては、柔軟性と引き換えに、レビューで短時間に読める小さな言語を選ぶことになります。

概念を判断に生かす

「Bicepとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 構成全体をファイルに記述し、特定の人の記憶だけに頼らず、まったく同じ環境を再現できるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

Linuxサーバーを守る1,000件のRBACルールを作成 お問い合わせ →

なぜ重要なのか

手作業で構築したインフラには履歴がありません。ポータルで設定を変えても理由は残らず、変更前のレビューもなく、障害後の環境の再構築は、すべてのクリックを誰かが覚えていることに頼ります。インフラをリポジトリ内のファイルに

すれば、アプリケーションコードと同じように、プルリクエスト、レビュアー、変更履歴、元へ戻す方法を持てます。

二つ目の利点は、環境を同じ形に保てることです。同じファイルから異なるパラメーターで構築した開発・ステージング・本番環境は、パラメーターで指定した部分だけが異なります。Bicepのデプロイは冪等であり、Microsoftの言葉では、同じファイルを何度デプロイしても「同じ種類のリソースが同じ状態」になります。そのため安全に繰り返せます。新しい環境も、同じコマンドの対象を新しいリソースグループへ変えるだけです。

三つ目は、変更前に内容を確認できることです。what-if操作は、ファイルと稼働中の状態を比較し、何かに手を加える前に、作成・変更・削除する予定のリソースを一つずつ列挙します。プルリクエストと組み合わせれば、「ファイアウォールのルールだけを変えると思う」という見込みを、レビュアーが検証できるものに変えられます。

制約も明示すべきです。対応するリソースはデプロイの方式と拡張機能によるため、他社のDNSや第二のクラウドのリソースには、引き続き別のツールが必要な場合があります。Resource Managerを使う方式では、BicepはAzureリソースを記述します。仮想マシン内のパッケージやファイルは、Ansibleなどの構成管理が担います。また、望ましい状態を宣言するツールなら、ファイルから消したものも削除すると期待していると、Resource Managerの削除・再デプロイに関する既定の動作に驚くことになります。この点は第四節で説明します。

実際の活用例

このサイトで紹介する複数のAzureプロジェクトはBicepを基盤としています。そのうち三つが、時間の経過とともにチームへ何をもたらすかを示しています。

あるプラットフォームでは、Azure環境全体をBicepで記述しています。全環境を同じテンプレートから生成します。Azure Container Appsのapiとwww、PostgreSQL Flexible Server、キャッシュ用Redis、前段のWAFポリシー付きFront Door、その下のVNet・NSG・プライベートDNS、診断のLog Analyticsを含み、イメージはプロジェクトのAzure Container Registryから取得します。すべてパラメーター化し、新しい環境の構築も、既存環境の変更も、プルリクエストで行います。そのページは、再現・レビューが可能になり、コードが正本のため、ずれが謎ではなくなったと報告します。

同じテンプレートが、リリース経路も支えます。リリースは、開発からステージング、本番へと進みます。Bicepが各環境の対象インフラを同じ形に保つため、ビルドの昇格先が微妙に異なる環境になることはありません。環境固有の設定をシークレットと分け、一つのビルド成果物を、再ビルドせず全段階へ通せます。

復旧計画の一部も、Bicepに記述しています。守るデータベースの隣のBicepで、バックアップ保持を設定し、特定時点への復元を、誰かが一度クリックした設定でなく、テンプレートの性質にします。周囲の環境もコードです。記憶から再構築する環境へデータを戻すだけでは、復旧と言いにくいためです。未解決の点も示します。地理冗長性は無効で、復旧時間目標は実測でなく提案です。テンプレートは設定を示し、復元の時間を示すのは訓練だけです。

問題が起きるところ

Bicepの問題の多くは、リソースグループ内のすべてを管理するツールだと期待することから生じます。既定では、ファイルに記述したものだけを、その記述に従う形で管理します。

ポータルの変更が、次のデプロイで消えます。 リソースの再デプロイは全プロパティを再適用し、省略したものを既定値へ戻します。インシデント中の手動修正が、次の通常のリリースで取り消されます。ファイルを各リソースの最終状態全体として扱い、既定値と異なる値をすべて書き、緊急変更もプルリクエストで行います。デプロイスタックの拒否設定で、外部からの編集自体も防げます。

テンプレートから消したものが、稼働・課金し続けます。 標準は増分モードで、ファイルにないリソースは残します。データベースをコードから消しても、Azureには残ります。Microsoftの答えはデプロイスタックです。対象を記録し、外れたものを切り離すか削除するかを、`actionOnUnmanage` で決めます。完全モードも、ファイルに記述のないリソースを削除しますが、Microsoftは勧めず、段階的に廃止しています。

what-ifに、誰もしていない変更が並びます。 ファイルでは省略してもAzureが既定値を補うプロパティを、削除と報告する場合があり、Microsoftは雑音と呼びます。`reference` 関数から値を設定するプロパティは毎回変更扱いで、`utcNow()` やセキュアなパラメーターはデプロイ中だけ存在するため、what-ifでは評価できないまま表示します。毎回のプルリクエストで実行し、暫定的な結果として読み、ノイズを生むプロパティの値を明示して、本当の変更が目立つようにします。

```
az deployment group what-if \
  --resource-group rg-staging \
  --template-file main.bicep \
  --parameters staging.bicepparam
```

パスワードがデプロイ履歴に現れます。 普通の文字列パラメーターは記録され、履歴を読める人には見えます。シークレットのパラメーターには `@secure()` デコレーターを付けて値をデプロイ履歴やログから除き、シークレット自体は、リポジトリ内のパラメーターファイルではなくKey Vaultに置きます。

参考資料

- Bicepとは：言語、利点、目指していないものの公式説明です。
- Bicepモジュール：互いを呼ぶファイルへの分割です。
- Bicepのパラメーター：型、デコレーター、既定値、`@secure()` が履歴から除くものです。
- デプロイモード：増分と完全、それぞれの省略リソースの扱いです。
- Bicepのwhat-if操作：事前確認、変更の種類、既知の雑音です。
- デプロイスタック：削除と拒否を含む、まとめて行うリソース管理です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「Bicepとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/bicep/