

Build 2021 (たぶん)最速Recap Day

bicep 💪

Build 2021 (たぶん)最速Recap Day

by Takekazu Omi(@Baleen.Studio)

2021/05/29 v1.0.0



Takekazu Omi @Baleen.Studio



自己紹介

近江 武一 [@takekazuomi](#)

- 所属 [jazug](#)、[baleen.studio](#)（仲間を募集中）
- [GitHub](#)、Azure関連のPull request、気がついたら直す方向で
 - Private AKS Cluster の `.bicep` を [PR](#)
 - SQL Elastic Pool のTemplate が動かなくなってたので [PR](#)
- Blog [kyrt.in](#)、[zenn.dev](#)を使い始めた
 - [ARM template DSL、Bicep を使おう\(1\)](#)
 - [ARM template DSL、Bicep を使おう\(2\)](#)

今日の話 💪

- //Build 2021で、Bicep 0.4 が発表
 - 発表は、[Ask the Experts: Infra as Code - Bicep](#)
- Bicep の紹介
 - そもそも、Bicep is 何？
- Bicep 0.4 詳細
 - Demonなど

//Build 2021

- //Build 2021 で、Bicep 0.4  が ~~リリース~~
- リリースは、6/2 予定
- リリースイベント [Bicep v0.4 Launch Call](#)、参加しよう！
- 未リリース  だけど
- [Nightlyのインストール](#) で試せるので是非

//Buildに間に合うかと思ったが、惜しかった

Bicep 💪 の 👍 👎

Bicepの紹介



Bicep 💪 の短い紹介

GitHubのレポジトリ [bicep](#)

- Azure ARM Template のDSL、現在プレビューの[v0.3.539](#)
 - json 直は辛い。DSL と Language Server で記述しやすく
 - プレビューだけど、MS 公式Azureサポートの対象
- ARM Template <-> bicep 間は薄いラッパーで相互変換も可
 - ARM Template をILとした、トランスペイラ
 - 類似: JavaScript(ARM Tempate) <-> TypeScript(Bicep)

Bicep 💪 の 👍

- ARM Template直より100倍楽に書けるシンプルな構文
- Azure RP の Day 0 サポート
- モジュール化、再利用性の向上
- Language Server 同時開発
- 構文のデプロイ前検証

Bicepの取組む課題

ARM Template は中々難しい。根本的な難しさは3つに分類できる

1. JSON構文の煩雑さ
2. モジュール化、再利用性の困難性
3. Azure Template Runtime/Azure RPの複雑さと不透明性

1. JSON構文の煩雑さ(1)

DSLの導入でARM TemplateのJSON構文よりシンプル。[例](#)

1. 文字列内に `[...]` で埋め込んでいるが式が直接書けるようになる
2. プロパティ名をいちいち、`" "` で囲む必要が無い
3. 文字列結合を、`concat()` ではなく `' ${name}-vm'` のように書ける
4. `reference(parameters('aksName')).properties.fqdn` の代わりに、
`aks.properties.fqdn` のようにプロパティアクセスで書ける

2. モジュール化、再利用困難性(1)

- 複数のbicepファイルに分割、 modules 構文で再利用できる
- moduleのパラメータ部分の補完が効く

```
vnet.bicep > ⚙ vnetName
  param vnetName string [
    default: 'vNet'
    metadata: {
      description: 'The name of the vi
    }
  }

  param addressPrefix string {
    default: '10.1.0.0/16'
    metadata: {
```

```
main.bicep > ↴ method
  module vnetMod './vnet.bicep' = {
    name: 'vnetMod'
    params: {
      v
      🔑 vnetName
    }
  }

  vnetName

  Type: string
  Write-only property
```

2. モジュール化、再利用困難性(2)

- **bicep module**はコンパイル時にパラメータチェックされ
- 変換後は、`Microsoft.Resources/deployments` のインラインテンプレートになる

参考：

- [bicep moduleを使う](#) <- 0.2からサポート

1. JSON構文の煩雑さ(2)

module と loop を使った例

- loop 例
- vm module

3. Azure Template Runtime/Azure RPの複雑さと不透明性

これは、元々のAPIの問題で `bicep` は、そのレイヤーを解決するツールでは無い

Bicep Team □ ARM Tempate Team 「問題があるなら、SR切るか、Issue上げて欲しいとのこと」

参考

Decompile

v0.2.59から、ARM Template -> bicep 変換の最初のリリースが入っている。

```
bicep decompile azueredeploy.json
```

生成後の `.bicep` を名前を直して使っているが、結構イケてる。

Bicep 0.4 の新機能

- The new Bicep linter
- Resource Snippets
- Resource and module scaffolding
- The `getSecret()` function for retrieving secrets for module params
- Bicep visualizer

Demo

Bicep linter

Linter, ルール

- Environment URL がハードコードされています
- パラメータが使われていません
- 変数が使われていません
- 文字列補間を使いましょう
- secure parameter にデフォルトがあります
- 文字列補間は必要ありません

getSecret()

secure な **string** の **module** のパラメータに、**Key Vault** からの既存のシークレットを使用する。例

```
module secretModule './secretModule.bicep' = {
    name: 'secretModule'
    params: {
        myPassword: kv.getSecret(secret1Name, secret1Version)
        mySecondPassword: kv.getSecret(secret2Name)
    }
}
```

Known limitations

- **object** と **arrays** を1行に書けない (例 `['a', 'b', 'c']`) ([#586](#)).
- 今のBicep は改行に厳しすぎる。過剰なルールを緩めたい。([#146](#))
- すべてのリソースで個々にapiVersionを指定する必要があります。
議論は [#622](#) にあります。

まとめ

bicep は、ARM Template の生産性を上げるツールとして秀逸。
ARM（腕） 、 bicep（上腕二頭筋） という名前付けらしい。読み方
は、 バイセップ

- 今回のコンテンツ、
 - GitHub [20210529-build202-bicep](#)
 - slide share [20210529-build202-bicep](#)
- Powerd by Marp。ありがとうございました。

終

