



AWS Application Composer では始める サーバーレスアプリケーション

#ちょっぴりDD

山崎 宏紀 (Hiroki Yamazaki)

Solutions Architect

Amazon Web Services Japan G.K.

山崎 宏紀 (Yamazaki Hiroki)

アマゾンウェブサービスジャパン

技術統括本部 ソリューションアーキテクト

興味:

- コンピューティング, 離散アルゴリズム
- Infrastructure as Code, Serverless



好きな AWS サービス:

- AWS Lambda, AWS CDK



趣味:

- 将棋, ボードゲーム, 水泳, ボルダリング (new!)



アジェンダ

- AWS Application Composer とは
- AWS Application Composer が解決する課題
- Demo
- まとめ

注意事項

- 2022 年 12 月現在、AWS Application Composer は Preview サービスです。現時点では、評価目的、テスト用途でご利用ください
- Preview 期間中、仕様は予告なく変更される可能性があります。予めご了承ください。本日は 2022 年 12 月時点の仕様に基づき解説します。

AWS Application Composer



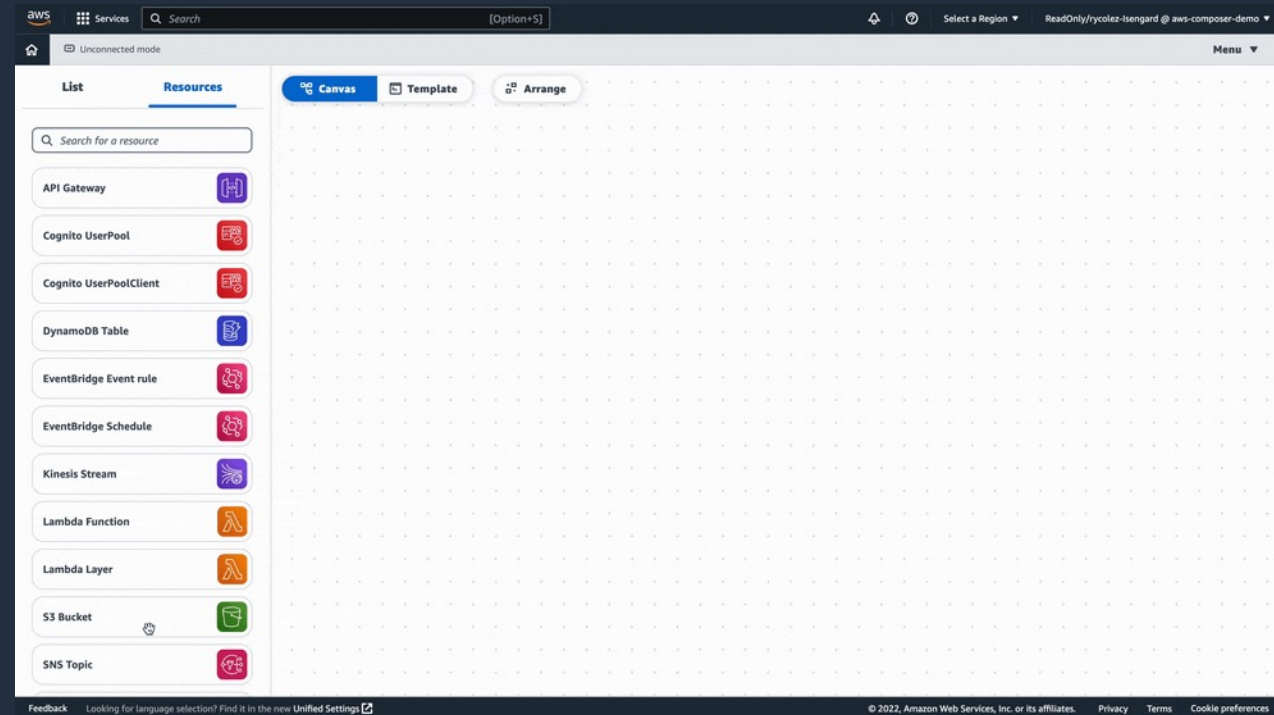
AWS Application Composer (Preview)

GUI でサーバーレスアプリケーションを構築し、IaC のコードを生成

- 新規作成だけでなく、既存 CloudFormation, SAM テンプレートも編集可能
- 東京をはじめ 6 つのリージョンでプレビュー開始

SAM: Serverless Application Model

IaC: Infrastructure as Code



キャンバスと CloudFormation テンプレート

The screenshot displays the AWS CloudFormation console interface. On the left, a sidebar lists various AWS resources. The main area is split into two views: a 'Canvas' view and a 'Template' view.

Canvas View: Shows a visual representation of the CloudFormation template. It includes an API Gateway resource named 'Api' with endpoints for GET /items, POST /items, GET /items/{id}, PUT /items/{id}, and DELETE /items/{id}. This API Gateway is connected to five Lambda Function resources: ListItem, CreateItem, GetItem, UpdateItem, and DeleteItem. All five Lambda functions are connected to a single DynamoDB Table resource named 'Items'.

Template View: Shows the CloudFormation YAML code for the 'Api' resource. The code is as follows:

```
1 Transform: AWS::Serverless-2016-10-31
2 Resources:
3   ListItem:
4     Type: AWS::Serverless::Function
5     Properties:
6       Description: !Sub
7         - Stack ${AWS::StackName} Function ${ResourceName}
8         - ResourceName: ListItem
9       CodeUri: src/api
10      Handler: index.handler
11      Runtime: nodejs18.x
12      MemorySize: 3008
13      Timeout: 30
14      Tracing: Active
15     Events:
16       ApiGETItems:
17         Type: Api
18         Properties:
19           Path: /Items
20           Method: GET
21           RestApiId: !Ref Api
```

At the bottom of the console, the status bar indicates 'YAML Ln 1, Col 1' with 0 errors and 0 warnings.

AWS Application Composer が 解決する課題

サーバーレスアプリケーション開発における課題

- サーバーレス初心者が複数の AWS サービスからアプリケーションを構成する場合、**急な学習曲線**を経験する可能性
- まず、どのサービスが必要なのか、それらをどのように連携させるのか、どのように良いデフォルトを設定するのかの理解が必要
- アプリケーションが複雑化すると、**アーキテクチャがどのように変化したか**を伝えたり、**可視化**したりすることが困難に

AWS Application Composer のメリット

Concept/Compose

**Collaborate/
Iterate**

Deploy

数分で動くアーキテクチャを構成

- ドラッグ & ドロップで動く
ビジュアルエディター
- 直感的な構成
- 共通のタスクを簡素化
 - IAM ポリシー
 - ベストプラクティスに
則った設定

チームコラボレーションと イテレーションを加速

- 何百行もの設定に目を通すこと
なくアーキテクチャを把握し、
チームコラボレーションを促進

IaC による透過的なデプロイ

- IaC とアーキテクチャの
ビジュアル表現は同期
- ビジュアルアーキテクチャを
デプロイ可能な構成に変換し、
CloudFormation テンプレート
による IaC を提供

Demo



デモの流れ

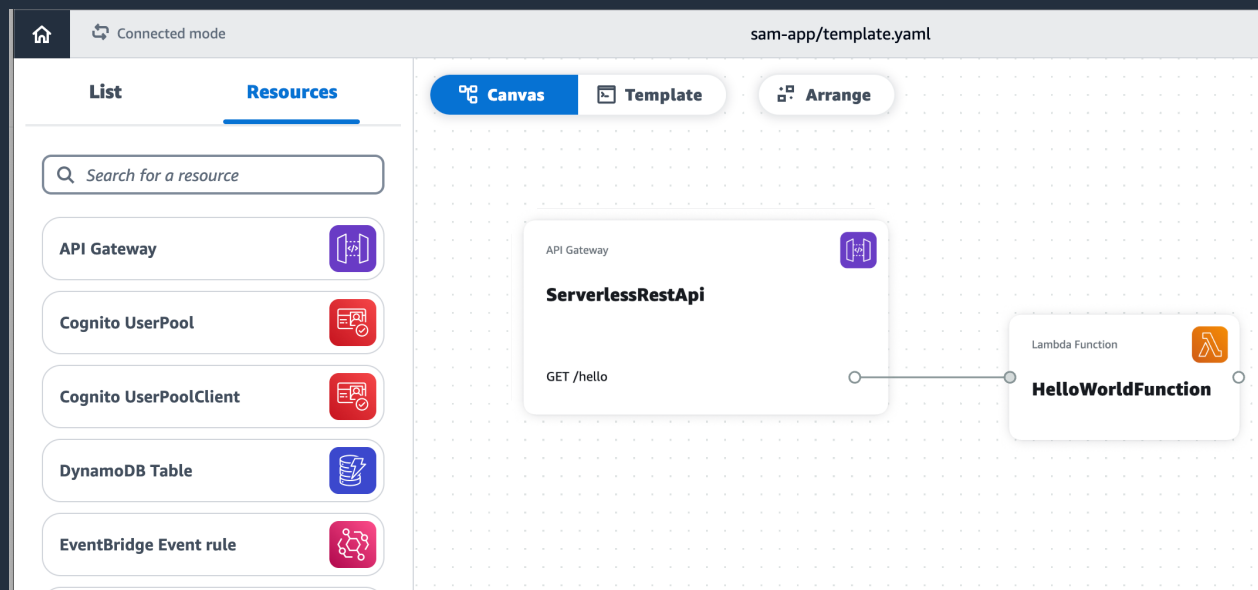
SAM で作成したサンプルアプリケーションに AWS Application Composer で機能を追加し、デプロイするまでをお見せします

```
1  AWSTemplateFormatVersion: '2010-09-09'
2  Transform: AWS::Serverless-2016-10-31
3  Description: |
4    sam-app
5    Sample SAM Template for sam-app
6  Globals:
7    Function:
8      Timeout: 3
9      MemorySize: 128
10 Resources:
11   AWS: Add Debug Configuration
12   HelloWorldFunction:
13     Type: AWS::Serverless::Function
14     Properties:
15       CodeUri: hello_world/
16       Handler: app.lambda_handler
17       Runtime: python3.9
18       Architectures:
19         - x86_64
```

インポート



テンプレート生成



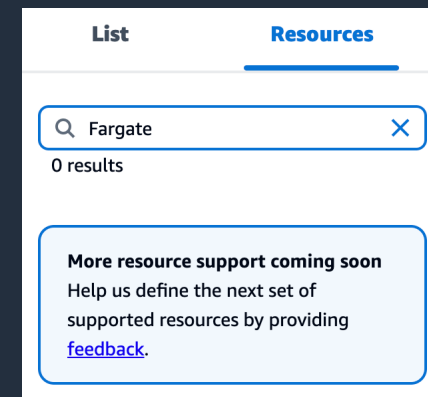
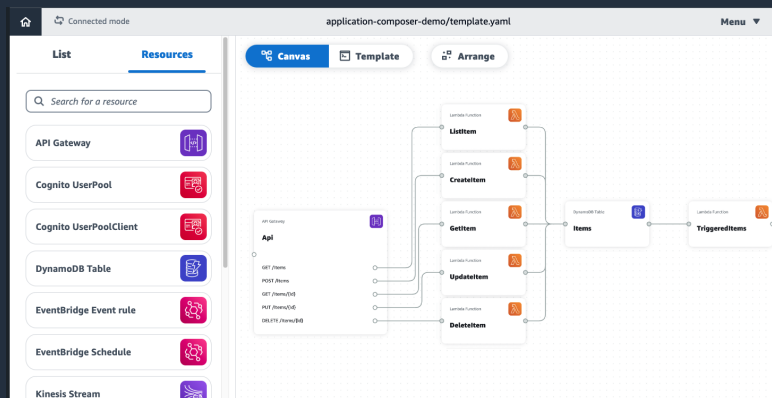
SAM テンプレート

AWS Application Composer

まとめ

まとめ

- 新サービス **AWS Application Composer (Preview)** を紹介
- **サーバーレスアプリケーション**をドラッグ&ドロップで開発し、**laC** を生成してデプロイ
- ぜひご自身でお試しく下さい！(Feedback もお待ちしております)



参考リンク: <https://aws.amazon.com/jp/about-aws/whats-new/2022/12/aws-application-composer-preview/>



Thank you!