Building Serverless Applications with SAM

Min San

Meet SAM!

Why SAM ?

- Local Execution and debugging
- Resource Creation
- Events Configuration
- IAM Policies
- Packing
- Deployment
- Logs

What is SAM ?

A serverless application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Note that a serverless application is more than just a Lambda function—it can include additional resources such as APIs, databases, and event source mappings

https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html

Serverless Application Model

CloudFomration Extension optimized for serverless

New Serverless Resources types: Funcion, SimpleTable, API, Layers

Support Anything CloudFormation Support

Support Both YAML and JSON

SAM templates

Using shorthand syntax to express resources and event source mappings, it provides infrastructure as code (IaC) for serverless applications.

SAM CLI

Provides tooling for local development, debugging, build, packaging, and deployment for serverless applications

SAM Template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
 GetProductsFunction:
  Type: AWS::Serverless::Function
  Properties:
   Handler: index.getProducts
   Runtime: nodejs10.x
   CodeUri: src/
   Policies:
    - DynamoDBReadPolicy:
       TableName: !Ref ProductTable
   Events:
    GetResource:
     Type: Api
     Properties:
      Path: /products/{productId}
      Method: get
 ProductTable:
  Type: AWS::Serverless::SimpleTable
```

Just 20 lines to create:

- Lambda function
- IAM role
- API Gateway
- DynamoDB table

SAM Resources

AWS::Serverless::Api
AWS::Serverless::HttpApi
AWS::Serverless::Function
AWS::Serverless::Application
AWS::Serverless::SimpleTable
AWS::Serverless::LayerVersion
AWS::Serverless::StateMachine

AWS::Serverless::Api

```yaml
Resources:
  ServerlessApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors: "'*'"
      Auth:
        DefaultAuthorizer: CognitoAuthorizer
        Authorizers:
          CognitoAuthorizer:
            UserPoolArn: !GetAtt UserPool.Arn
      GatewayResponses:
        UNAUTHORIZED:
          StatusCode: 401
          ResponseParameters:
            Headers:
              Access-Control-Expose-Headers: "'WWW-Authenticate'"
              Access-Control-Allow-Origin: "'*'"
              WWW-Authenticate: >-
                'Bearer realm="admin"'
```

Simple Table

```
FileMetadata:
  Type: AWS::Serverless::SimpleTable
  Properties:
    TableName: FileMetadata
    PrimaryKey:
      Name: Key
      Type: String
```

AWS::Serverless::Function

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function R
md#awsserverlessfunction
    Properties:
      CodeUri: hello-world/
      Handler: hello-world
      Runtime: go1.x
      Tracing: Active # https://docs.aws.amazon.com/lambda/lates
      Events:
        CatchAll:
          Type: Api # More info about API Event Source: https://
          Properties:
            Path: /hello
            Method: GET
      Environment: # More info about Env Vars: https://github.co
        Variables:
          PARAM1: VALUE
```

Events

- S3
- Api
- SNS
- SQS
- Kinesis
- HttpApi
- Cognito
- IoTRule
- Schedule
- AlexaSkill
- DynamoDB
- EventBridgeRule
- CloudWatchLogs
- CloudWatchEvent

```
Events:
  UploadEvent:
    Type: S3
    Properties:
      Bucket: !Ref ServerlessMeetup
      Events: s3:ObjectCreated:*
```

```
    Events:
      MySQSEvent:
        Type: SQS
        Properties:
          Queue: !GetAtt MySqsQueue.Arn
          BatchSize: 10
MySqsQueue:
  Type: AWS::SQS::Queue
```

Other events

```
MySqsQueue:
  Type: AWS::SQS::Queue
```

```
SrcBucket:
  Type: AWS::S3::Bucket
```

```
stream:
  Type: "AWS::Kinesis::Stream"
  Properties:
    ShardCount: 1
streamConsumer:
  Type: "AWS::Kinesis::StreamConsumer"
  Properties:
    StreamARN: !GetAtt stream.Arn
    ConsumerName: "TestConsumer"
```

IAM Policies

AWS managed IAM policies

Inline IAM policy document

AWS SAM policy templates

https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-policy-template-list.html

Inline IAM policy VS AWS SAM policy templates

```
Resources:
  MyFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: index.handler
      Runtime: nodejs8.10
      CodeUri: 's3://my-bucket/function.zip'
      Policies:
      - Statement:
        - Sid: SSMDescribeParametersPolicy
          Effect: Allow
          Action:
          - ssm:DescribeParameters
          Resource: '*'
        - Sid: SSMGetParameterPolicy
          Effect: Allow
          Action:
          - ssm:GetParameters
          - ssm:GetParameter
          Resource: '*'
```

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - SQSPollerPolicy:
          QueueName:
            !GetAtt MyQueue.QueueName
```

Globals

```yaml
Globals:
  Function:
    Runtime: nodejs6.10
    Timeout: 180
    Handler: index.handler
    Environment:
      Variables:
        TABLE_NAME: data-table

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          MESSAGE: "Hello From SAM"

  ThumbnailFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        Thumbnail:
          Type: Api
          Properties:
            Path: /thumbnail
            Method: POST
```

Deployment

```
Resources:
 MyLambdaFunction:
   Type: AWS::Serverless::Function
   Properties:
     Handler: index.handler
     Runtime: nodejs4.3
     CodeUri: s3://bucket/code.zip

     AutoPublishAlias: live

     DeploymentPreference:
       Type: Canary10Percent10Minutes
       Alarms:
         # A list of alarms that you want to monitor
         - !Ref AliasErrorMetricGreaterThanZeroAlarm
         - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
       Hooks:
         # Validation Lambda functions that are run before & after traffic shifting
         PreTraffic: !Ref PreTrafficLambdaFunction
         PostTraffic: !Ref PostTrafficLambdaFunction
```

SAM CLI

- Initialization - Create Project
- Validate
- Build
- Local invoke (Local event generate, Debugger)
- Package
- Deploy
- Logs
- Publish

Demo Time </>

Thank You!
Happy Hacking Guys