**AWS Serverless Architecture and Basics Operations Using SAM template.**

**Abstract:**

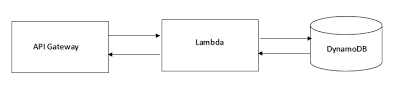
         Abstract microservices are an architectural and organizational approach to software development to speed up deployment cycles, foster innovation and ownership, and improve maintainability and scalability of software applications. Using a microservices approach, software is composed of small independent services that communicate over well-defined APIs.

**Amazon API Gateway:**

        API Gateway is a service for creating and managing APIs, helps to manage stages and easy deployment.

Gateway integration supports for lambda, mock data,http and aws other service integration. One of the best feature is SDK and swagger definition.

* Swagger definition: It's makes easy for defining all resource end points easily, supports json/yaml format.
* Once you done with API's creation, create stage using deploy option available in resource level, from stage download SDK (presently supports for android, iOS, JavaScript and Java).
* Authorizers: Support for securing APIs - Custom Authorizer and AWS\_IAM(cognito pool).
* Models - what we call as pojos in Java holds object properties.

[](https://4.bp.blogspot.com/-Cxq-j2blPkE/WTeLaUkXoiI/AAAAAAAADBE/fd6p7LSDdro30K1mu41skvr5WNbB4z7cgCLcB/s1600/sam.PNG)

**Lambda:**

    Lambda is service for writing your business logic or what we call usually retrieving data from database.

    We can write Business logic using different technologies (Java, C#, Node and Python).

* No operating systems to choose, secure, patch, or manage.
* No servers to right size, monitor, or scale out.
* Easy deployment and monitoring logs.

**Amazon DynamoDB:**

DynamoDB is NoSQL, Fully managed NoSQL database service by Amazon, key-value store. Supports up to 400kb each items. Designed to address the core problems of database management, performance, scalability, and reliability.

* Required Attribute: partition Key.
* Optional Attribute: Range Key.
* Except partition key DynamoDB is schema less.

**AWS CloudFormation**

AWS CloudFormation provides a service for managing resources.

1. How to deploy?

To upload already existing resource (template) to CloudFormation use Launch CloudFormer.

1. AWS SAM

The AWS Serverless Application Model (AWS SAM, previously known as Project Flourish) extends AWS CloudFormation to provide a simplified way of defining the Amazon API Gateway APIs, AWS Lambda functions, and Amazon DynamoDB tables needed by your serverless application.

Pros:

* Deploy APIs easily using swagger definition in API Gateway.
* Deploy Lambda Functions easily in Lambda Service.
* Create Tables easily in DynamoDB.

Cons:

* SAM templates supports for any one stage creation in API Gateway.
* SAM doesn’t support for versioning and aliasing in Lambda.
* SAM doesn’t support for creating Table with **Range** Key.

Resource Types:

* AWS::Serverless::Function 🡪 To create Lambda.
* AWS::Serverless::Api 🡪 To create APIs and stage.
* AWS::Serverless::SimpleTable 🡪 To Create Tables.

1. Sample Article CRUD Operations example using AWS SAM.



* To Create packed template use the below command

aws cloudformation package

--template-file articletemplate.yaml

--s3-bucket articleresource

--output-template-file article-packaged-template.yaml

* To deploy cloud formation template to AWS CloudFormation Stack

aws cloudformation deploy

--template-file article-packaged-template.yaml

--stack-name ArticleStack

--capabilities CAPABILITY\_IAM

OR

To deploy manually follow the step 5 flow.

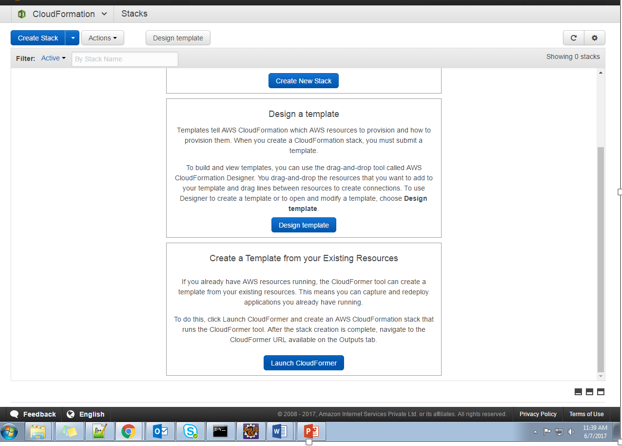
Note: AWS CLI should be installed.

1. Lambda Source Code:

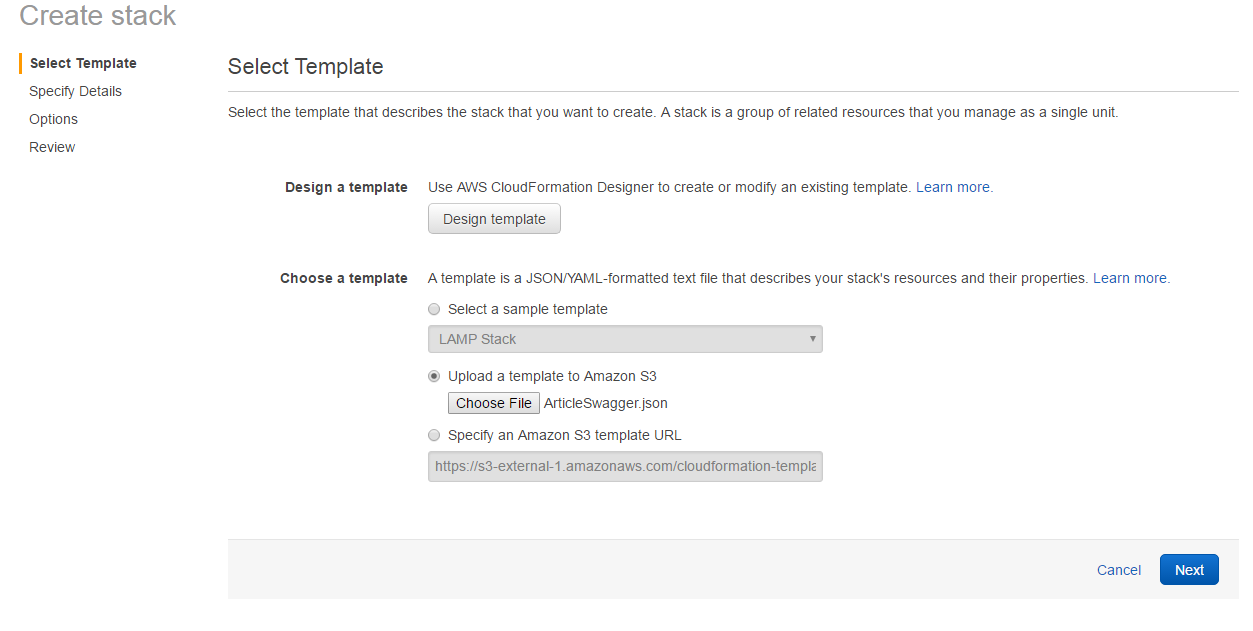
Take Latest code from source code repository and generate zip with name article.zip.

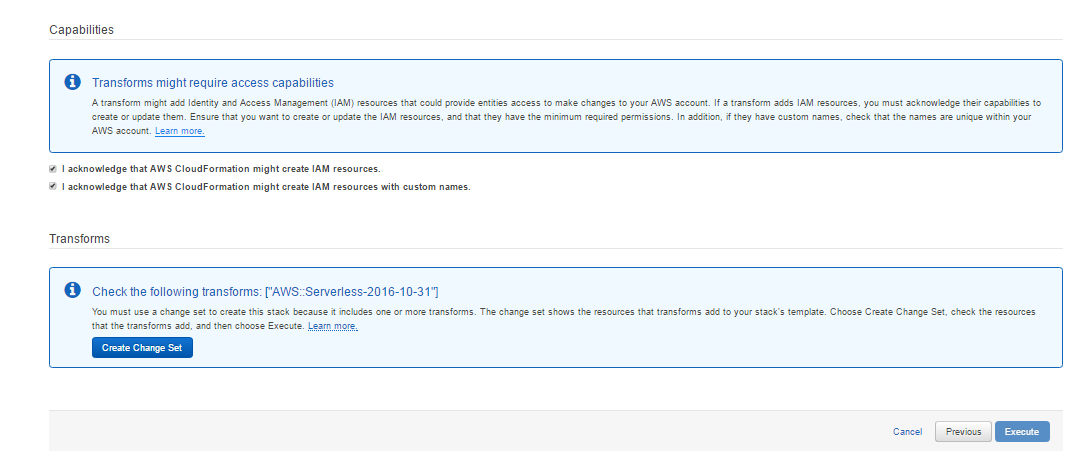
Source code path: <https://github.com/Garaddik/ArticleServerlessApp.git>

1. Loud Formation

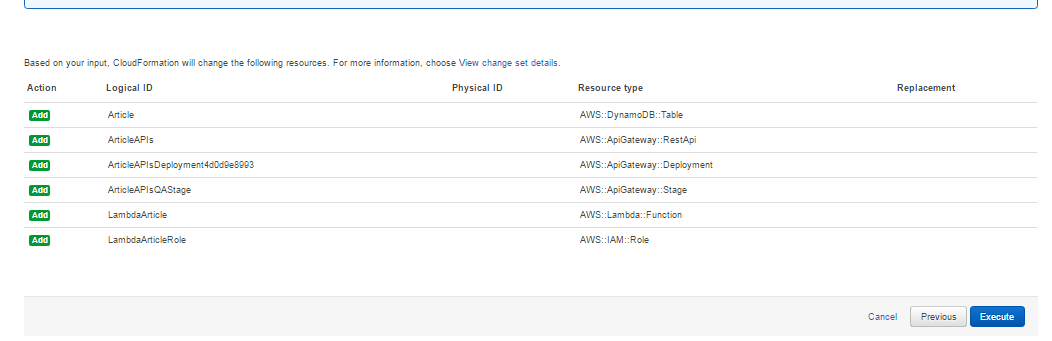


Select Launch CloudFormer

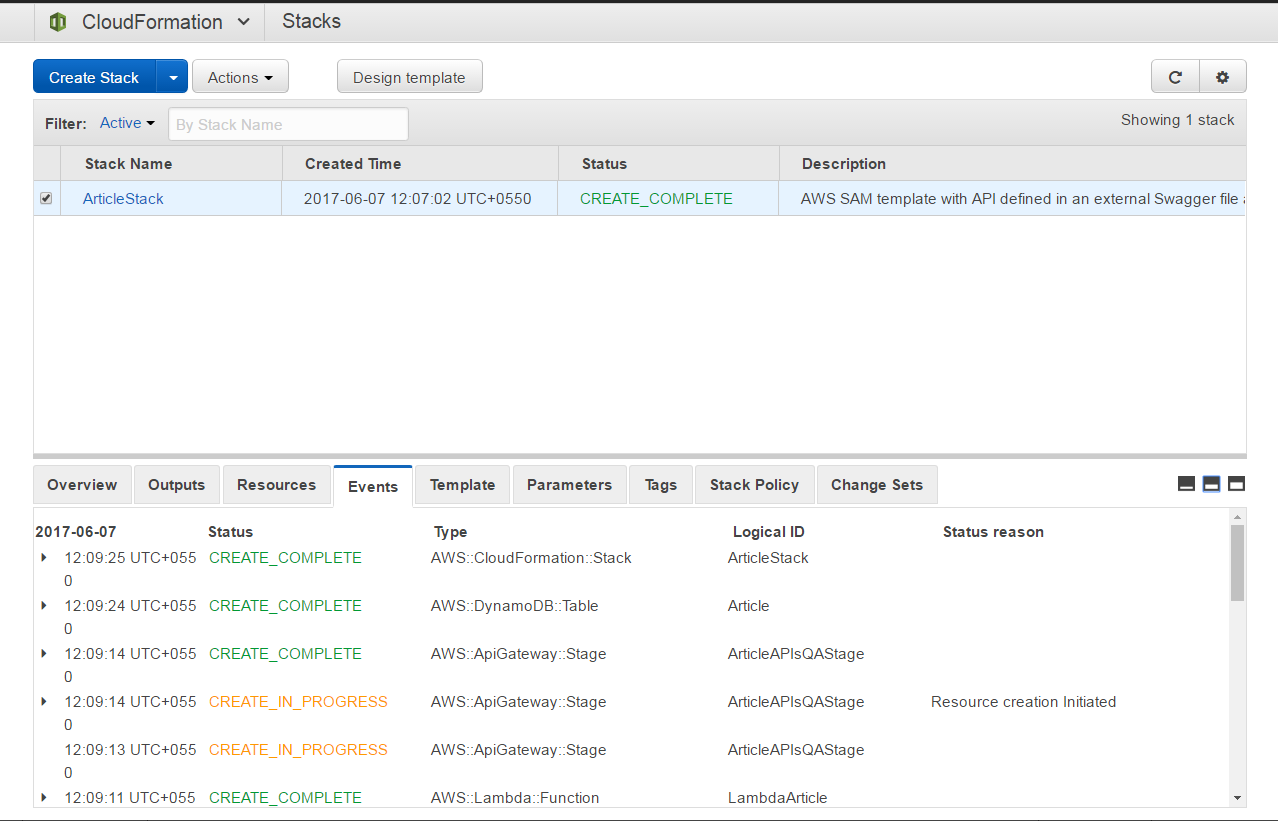




Select check box, click on create change set and it will take some time to generate resource definition.



Ready with resource definition generate, Click one execute Button and it will take some time to generate all resource in specified Services presently we are creating resources in API gateway , Lambda, and DynamodDB.



Here we go, just using Two CLI commands all the resources created.

**Conclusion:**

Creating application components that are easy to maintain, decoupled, and scalable. When you create a logic tier where integration occurs via Amazon API Gateway and computation occurs within AWS Lambda, you are on your way to realizing those goals while reducing the amount of effort to achieve them. Together, these services provide a HTTPS API front end for your clients and Stub to easily integrate and consume APIs.

**References:**

* <https://github.com/awslabs/serverless-application-model>
* <https://github.com/awslabs/serverless-application-model/blob/master/HOWTO.md>
* <https://github.com/awslabs/serverless-application-model/blob/master/versions/2016-10-31.md>
* <https://aws.amazon.com/cloudformation/details/#designer>
* <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>
* <https://docs.aws.amazon.com/amazondynamodb/latest/gettingstartedguide/Welcome.html>
* <https://aws.amazon.com/documentation/apigateway/>
* <https://github.com/awslabs/serverless-application-model>
* <https://aws.amazon.com/cloudformation/>
* <https://docs.aws.amazon.com/lambda/latest/dg/deploying-lambda-apps.html#lambda-app-cloud-formation>
* <http://swagger.io/specification/>