**Content**

[1. SAM (Serverless Application Model) 2](#_Toc15987075)

[1. The file 2](#_Toc15987076)

[2. AWS SAM CLI 3](#_Toc15987077)

[3. StepFunctions 3](#_Toc15987078)

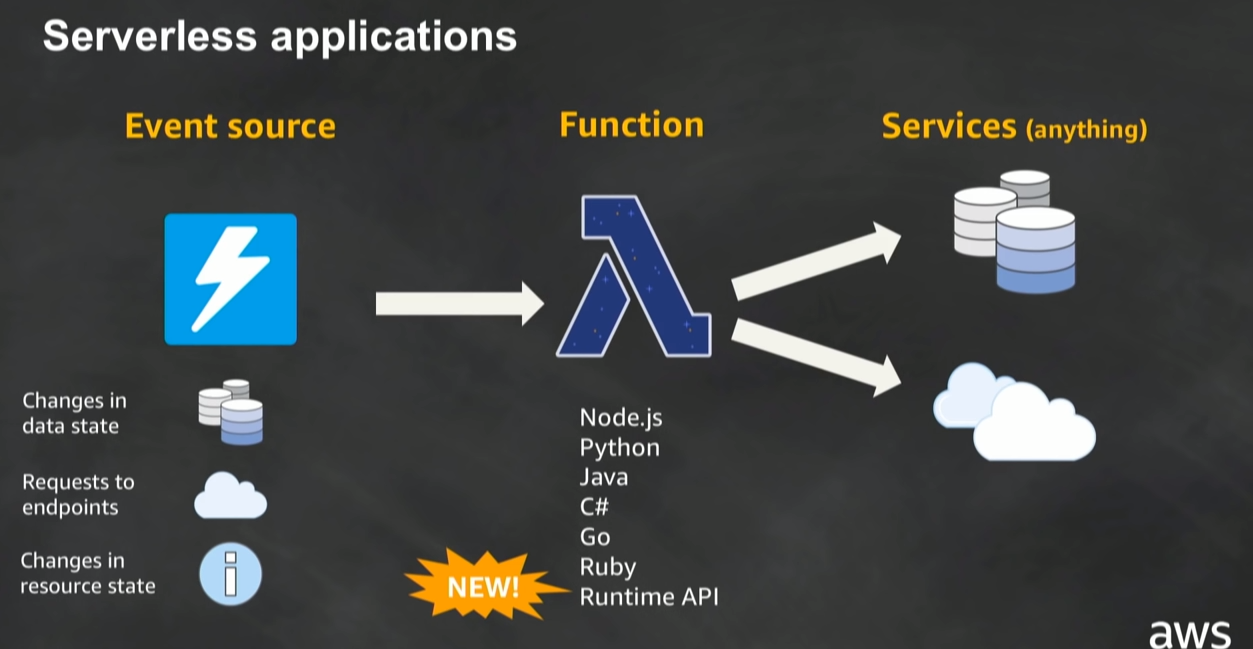
[2. Lambda – versions and aliases 4](#_Toc15987079)

[3. API Gateway 5](#_Toc15987080)

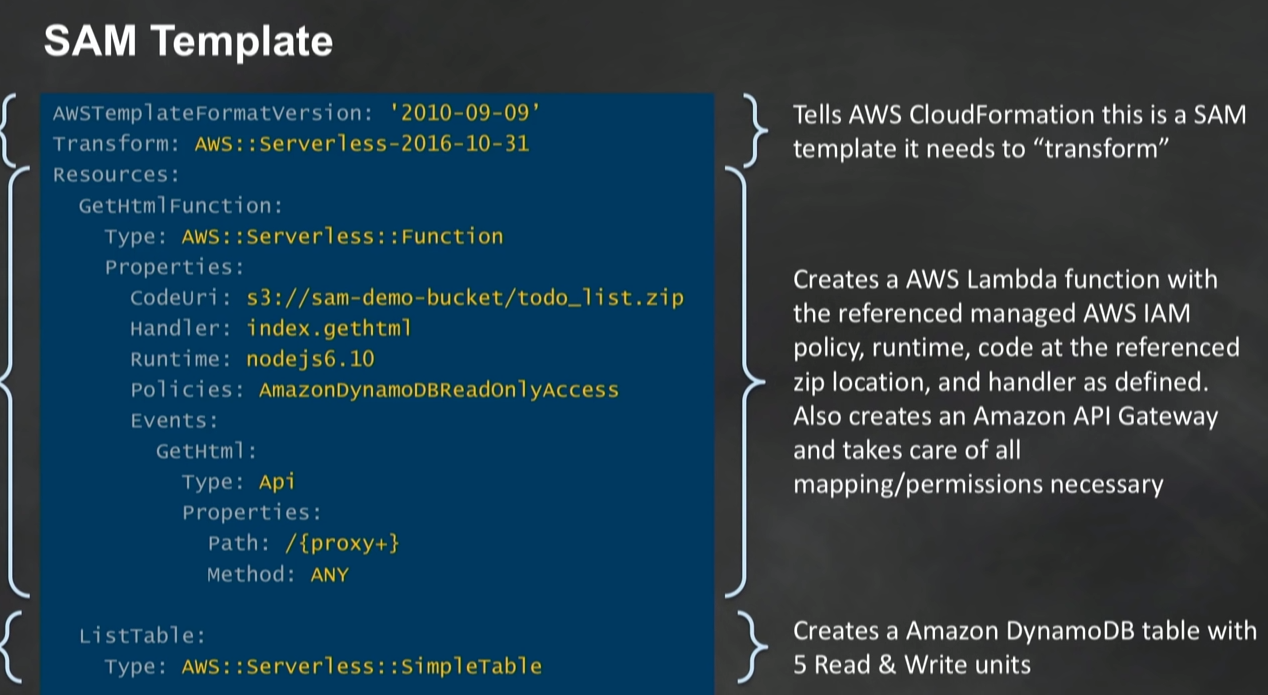
[1. API Gateway Private Endpoints 5](#_Toc15987081)

1. SAM (Serverless Application Model)

AWS CloudFormation extension, optimized for serverless. Special serverless resource types: functions, APIs, tables, Layers and Applications. Supports anything AWS CloudFormation supports.



* 1. The file



<https://bit.y/2xWycnj>

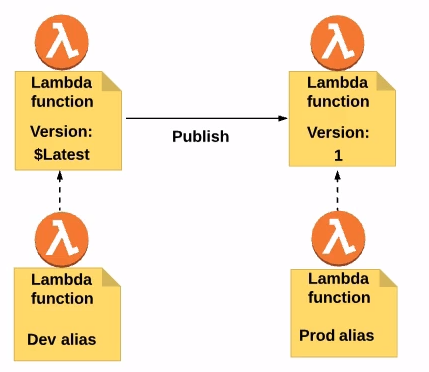
* 1. AWS SAM CLI

CLI tool for local development, debugging, testing, deploying and monitoring serverless applications. Uses open source docker-lambda images to mimic Lambda’s execution environment such as timeout, memory limits, runtimes.

* 1. StepFunctions

1. Lambda – versions and aliases

* You can publish one or more versions of your Lambda function
* Each Lambda Function has a unique ARN.
* After you publish a version it can not be charged
* The latest version of the function is tagged as $Latest
* Lambda supports creating aliases for each Lambda function version
* An alias is a pointer to a specific Lambda function version
* Each alias has unique arn
* Alieses can be modified. **You can update an alias to point to different version.**
* Lambda Canary: shit traffic between 2 versions based on weights.



1. API Gateway

You can create APIs for use in your own client applications (apps) or you can make your APIs available to third-party app developers.

* REST – stateless
* WebSocket – stateful API specific entity – route. (real time chat app: onConnect, onDisconnect, sendMessage)

A REST API receives message from the client and forwards those to the backend resource. It also forwards responses from the backend to the client. But a Web Socket API supports two-way communication between client apps and your backend. The backend can send callback messages to connected clients. The backed services can independenty send messages to its clients. When REST there is no response withour a request. When WEB Socket it can be.

Swagger – helps developers to build RESTful web services.

Authorization – API Key

* 1. API Gateway Private Endpoints

Requirements:

* A VPC with at least one subnet and DNS resolution enabled.
* A VPC endpoint configured such:

1. Service name = „com.amazonaws.{region}.execute-api“
2. Enable Private DNS Name = enabled
3. A security group set to allow TCP Port 443 inbound from either an OP range in your VPC or another security group in your VPC

* An API Gateway with

1. Endpoint type = „Private“
2. A resource policy giving access to your API from the VPC endpoints.
   1. API Gateway features

Why authentication? What if a user checks the source code of the backend application and steals our url?

* + 1. API keys and usage plans

If you plan to create an API which is shared with other developers, not the users of your app, but other developers creating their own apps. Like Google Maps API – you can send them coordinates and get back some information. IF you want to use them, you need to register and you will get an API Key by them. Its that key which you need to pass with any request you send to the API, because you identify yourself with that key and Google can track your usage / limit you. In AWS you can do the same. You can event setup Usage Plans and assign it to given Keys. Its not important if you are going to use your API on your own.

* + 1. Custom Domain Names

You will not have the generic aws url, but you can connect the domain you own.

* + 1. Client Certificates

Client certificates are important if you plan on forwarding incoming requests to yet another http endpoint and on the another http endpoint you want to validate that http request that endpoint gets insteed stams from the API gateway. Certificates proves this.

You can use API Gateway to generate an SSL certificate and use its public key in the backend to verify that HTTP requests to your backend system are from API Gateway. This allows your HTTP backend to control and accept only requests originating from Amazon API Gateway, even if the backend is publicly accessible.

**Rotate and Expiring Client Certificate** - the client certificate generated by API Gateway is valid for 365 days. You must rotate the certificate before a client certificate on an API stage expires to avoid any downtime for the API. You can check the expiration date of certificate by calling clientCertificate:by-id of the API Gateway REST API or the AWS CLI command of get-client-certificate and inspecting the returned expirationDate property.