A BEGINNER'S GUIDE TO MASTERING THE AWS SERVERLESS APPLICATION MODEL



Serverless applications are changing the way companies do business by enabling them to deploy much faster and more frequently—a competitive advantage.

Amazon's AWS Serverless Application Model (AWS SAM) has been a game changer in this space, making it easy for developers to create, access and deploy applications, thanks to simplified templates and code samples.

(This tutorial is part of our AWS Guide. Use the right-hand menu to navigate.)

What you'll learn in this guide

This guide is a primer for developers who are interested in learning how to program in AWS SAM. We'll discuss the following key AWS SAM topics and features:

- Key Terminology
- Who uses AWS SAM?
- The AWS Serverless Application Model
- What AWS resources are used in AWS SAM?
- AWS Serverless Application Repository
- The benefits of programming in AWS SAM
- AWS SAM for Serverless Applications
- Serverless Application Model Examples

Key terminology

Here are three key terms you need to understand to start programming in AWS SAM:

- **Function-as-a-Service (FaaS)**. Think of FaaS as a ready-to-implement framework that can be easily tailored to the needs of an enterprise business. FaaS allows customers to develop and run code without the need to provision and manage infrastructure.
- **Compute service.** An on-demand FaaS that specializes in serverless computing.
- **Serverless application.** An application, programmed in the cloud, that requires no server maintenance.

(Learn more about serverless architecture.)

Who uses AWS SAM?

AWS SAM is an important resource for any developer who:

- Is ready to use serverless computing
- Wants to learn more about serverless architecture

The resources available within AWS SAM make it easy for any programmer to get their feet wet with low-cost, efficient serverless computing services provided by Amazon.

What is AWS SAM (Serverless Application Model)?

The AWS Serverless Application Model (SAM) is designed to make the creation, deployment, and execution of serverless applications as simple as possible. This can be done using AWS SAM templates with just a few choice code snippets—this way, practically anyone can create a serverless app.

What AWS resources are used for AWS SAM?

Let's look at the resources that are used in the AWS Serverless Application Model.

AWS SAM CLI, YAML & CloudFormation

AWS SAM applications can be created using the AWS SAM CLI command line tool. SAM CLI lets you build, test, and deploy applications using either:

- SAM CLI templates
- The AWS Cloud Development Kit (CDK)

SAM CLI can also be used for application deployment.

You can define and model SAM application templates using YAML.

AWS SAM templates are an extension of AWS CloudFormation templates. At deployment, SAM transforms the SAM syntax into CloudFormation syntax. CloudFormation sets up and configures the server infrastructure that your serverless applications will run under, allowing you to concentrate on your application rather than your infrastructure.

AWS Lambda

The AWS Lambda FaaS platform exists so that developers can run code without administering servers. Using the Lambda compute service, developers can upload deployable code. Then, Amazon handles the administration, charging only for the accumulated runtime used.

The Lambda console makes it easy to create applications with just a few clicks. Lambda supports many languages through Lambda runtime environments, including C#, Go, Java, .Net, Node.js, and Python.

Developers can create single, simple, and scalable Lambda functions that can be invoked in several ways, including:

- **Automatic invocation using triggers in Lambda or another AWS service.** Using triggers, Lambda functions can be invoked in response to lifecycle events, external events, or from a schedule.
- **Event source mapping from another AWS resource.** Event source mappings read items from an Amazon Kinesis or Amazon DynamoDB stream; an Amazon SQS queue; or another AWS resource and send those items to a Lambda function.
- **Other AWS services that can directly invoke Lambda functions.**

Of note, functions created on Lambda's architecture are scalable for optimum performance.

(Compare AWS Lambda with AWS ECS.)

Amazon API Gateway

Using the Amazon API Gateway service, developers can create, deploy, secure, and monitor the frontend of their serverless application. API Gateway extensions can be used in your AWS SAM templates.

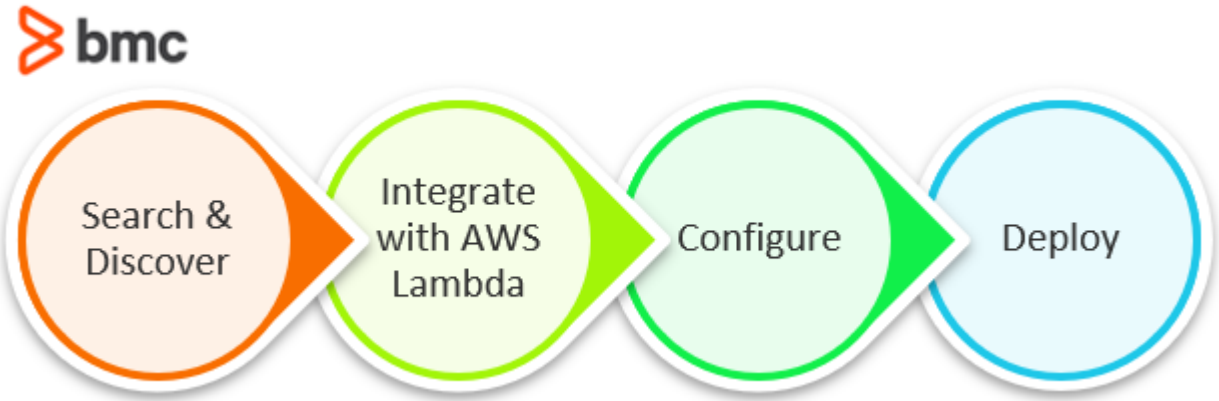
Like AWS Lambda, which is designed to create the backend of an application, Amazon API Gateway takes the complexity out of writing and deploying code that executes the front-door entry point to your application, including authorization. The Gateway APIs work with AWS Lambda backends, as well as Amazon EC2 and other web application services.

Amazon keeps its API Gateway service low-cost and affordable for developers by only charging for:

- Calls made to APIs
- Data transfers out of them

AWS Serverless Application Repository

The AWS Serverless Application Repository is a searchable ecosystem that allows developers to find serverless applications and their components for deployment. Its helps simplify serverless application development by providing ready-to-use apps.



Here are the basic steps:

1. **Search and discover.** A developer can search the repository for code snippets, functions, serverless applications, and their components.
2. **Integrate with the AWS Lambda console.** Repository components are already available to developers.
3. **Configure.** Before deploying, developers can set environment variables, parameter values, and more. For example, you can go the plug-and-play route by adding repository components to a larger application framework, or you can deconstruct and tinker with the code for further customization. If needed, pull requests can also be submitted to repository authors.
4. **Deploy.** Deployed applications can be managed from the AWS Management Console. A developer can follow prompts to name, describe and upload their serverless applications and components to the ecosystem where they can be shared internally and with other developers across the ecosystem. This feature makes AWS SAM a truly open-source environment.

Benefits of programming in AWS SAM

You can build serverless applications for almost any type of backend service without having to worry about scalability and managing servers. Here are some of the many benefits that building serverless applications in AWS SAM has to offer.

Low cost & efficient

AWS SAM is low-cost and efficient for developers because of its pay-as-you-go structure. The platform only charges developers for usage, meaning you never pay for more of a service than you use.

Simplified processes

The overarching goal of AWS SAM is ease of use. By design, it's focused on simplifying application development so that programmers have more freedom to create in the open-source ecosystem.

Quick, scalable deployment

AWS SAM makes deployment quick and simple by allowing developers to upload code to AWS and letting Amazon handle the rest. They also provide robust testing environments, so developers don't miss a beat. All of this occurs on a platform that is easy to scale, allowing apps to grow and change

to meet business objectives.

Convenient & accessible

Undoubtedly, AWS SAM offers a convenient solution for developing in the cloud. Its serverless nature also means that it is a universally accessible platform. The wide reach of the internet makes it easy to execute code on-demand from anywhere.

Decreased time to market

Overall, choosing a serverless application platform saves time and money that would otherwise be spent managing and operating servers or runtimes, whether on-premises or in the cloud. Because developers can create apps in a fraction of the time (think hours—not weeks or months), they are able to focus more of their attention on accelerating innovation in today's competitive digital economy.

AWS SAM for Serverless Applications

It's clear that AWS SAM is a highly efficient, highly scalable, low-cost, and convenient solution for cloud programming.

But for those who haven't yet made the switch, there are some concerns that arise from developing using AWS SAM, including:

- A general lack of control over the ecosystem that developers are coding in.
- Vendor lock-in that may occur when you sign up for any FaaS.
- Session timeouts that require developers to rewrite code, making it more complex instead of simplifying the process.
- AWS Lambda timeouts: Lambda functions are limited by a timeout value that can be configured from 3 seconds to 900 seconds (15 minutes). Lambda automatically terminates functions running longer than its time-out value.

The first two points are common drawbacks of any outsourcing strategy. The latter two concerns, however, might mean you'll implement a few new workarounds. Regardless, the many benefits of programming in AWS SAM cannot be overlooked.

Serverless Application Model examples

To better understand AWS Serverless Application Model (SAM), it helps to see examples of how it is used in the real world.

1. Data processing pipeline

Use AWS SAM to automate ingesting data, along with transforming and storing it. A typical pipeline uses AmazonS3, AWS Lambda, AWS Glue, and Amazon Kinesis, Amazon DynamoDB, or Amazon Redshift. Uploading a file to a bucket in S3 triggers a Lambda function for pre-processing data. Then it goes to Kinesis, for real-time analytics, or into a database like Amazon DynamoDB. AWS Glue can extract, transform, and load data into applications. This kind of pipeline is scalable, fault-tolerant, and provides cost efficiencies in processing high-data volumes.

2. Serverless web application

Eliminate the burden of managing infrastructure with the serverless web application. The approach may use a package of services, including Amazon API Gateway, AWS Lambda, Amazon S3, Amazon DynamoDB, and secure authentication by Amazon Cognito. For low-latency global access, deploy these services through Amazon CloudFront. Serverless web applications are ideal for modern web apps because they automatically scale to accommodate traffic and help reduce operating costs.

3. Event-driven microservices

Architecting and integrating services that communicate through events and take action in real-time is an approach that e-commerce platforms, entities managing financial transactions, and organizations that need real-time analytics use. It utilizes services like Amazon EventBridge, AWS Lambda, Amazon SQS, and DynamoDB. An event happens, like a customer placing an order, which is published on EventBridge for routing to Lambda. That service may validate the order, process payment, and update the inventory. A service like Amazon SQS provides asynchronous processing. DynamoDB stores all data from the transaction. This approach offers scalability, isolates faults, and makes it possible to develop services independently.

Learn more about The 15 Best Practices for Building a Microservices Architecture.

Related reading

- BMC DevOps Blog
- What's AWS VPC? Amazon Virtual Private Cloud Explained
- The AWS Well-Architected Framework: 5 Pillars & Best Practices
- Serverless Best Practices
- MongoDB vs DynamoDB: Comparing NoSQL Databases
- AWS re:Invent: The Complete Guide