INNOVATE

MODERN APPLICATIONS EDITION

27&28 October 2021

Getting started with serverless applications

Caragh Lyons
Senior Solutions Architect
Amazon Web Services

Who am I?

Caragh Lyons

Senior Solutions Architect

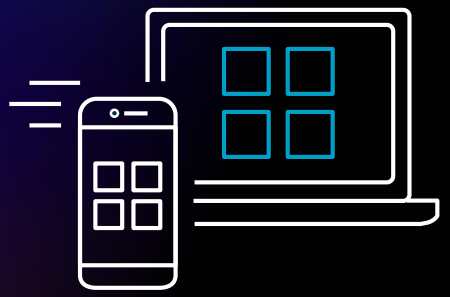
AWS Sydney Australia

”My background is in Application Architecture and for me, Serverless is a game changer. It means I can focus on my application and not the infrastructure”

Agenda

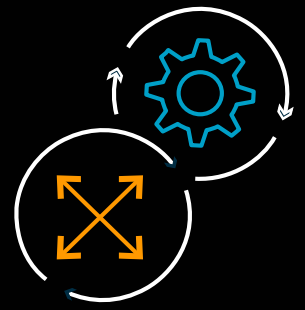
- Application architecture
- Introduction to Serverless
- *Demo*: AWS Lambda
- Serverless Application Model (SAM)
- *Demo*: Serverless application

What do *you* need to drive success?

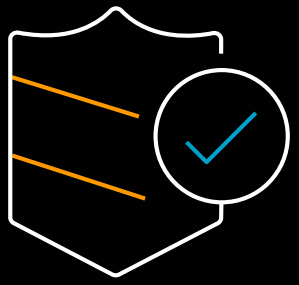


Get to market faster

Lower total cost of ownership



High performance and scalability

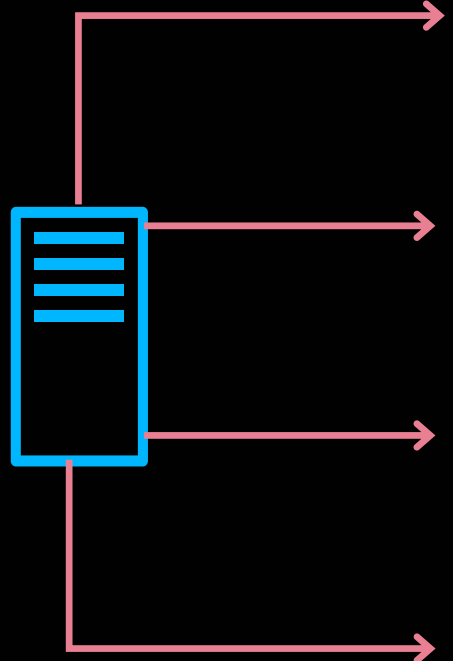


Security and isolation by design

Application architecture journey

The single server architecture

Many application architecture s start here



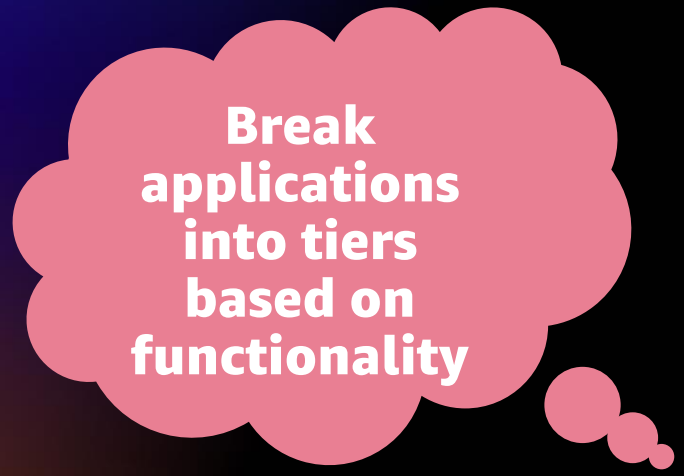
Proxy/Firewall: NGINX, Apache, F5, Zeus

Server: IIS, Apache, Tomcat

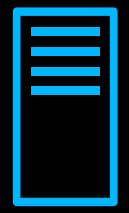
Code: Node, .Net, Ruby, Java, Go, Python

Database: SQL, MySQL, PostgreSQL, Mongo

The tiered architecture



Proxy/Firewall



Server

Database

The tiered architecture

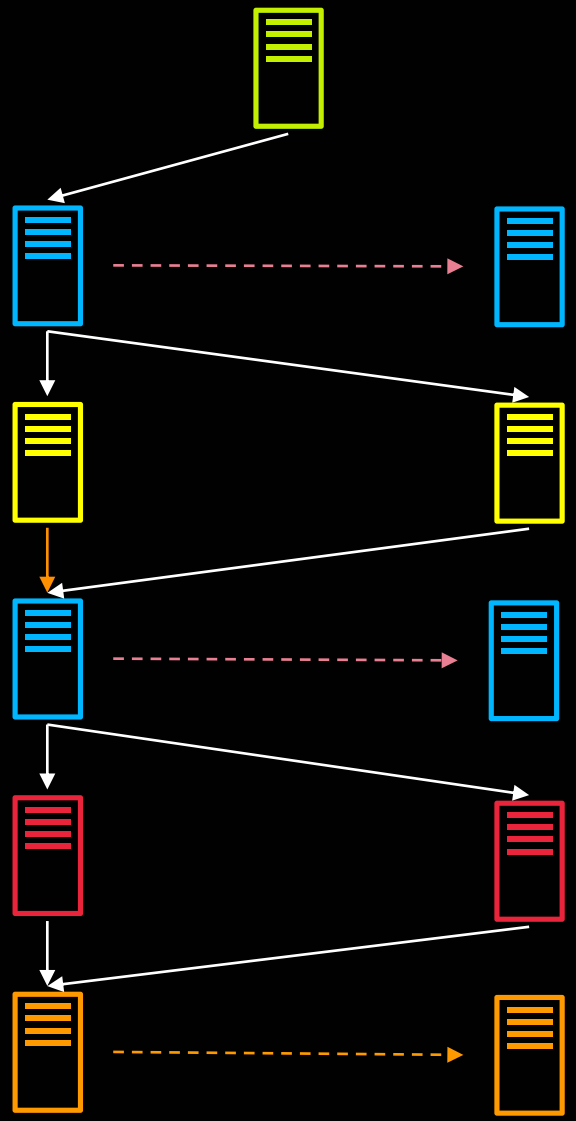
Proxy/Firewall

Client

Backend

Database

The redundant tiered architecture



Proxy/Firewall

Client load balancer

Client

Backend load balancer

Backend

Database

Servers

What size servers are right for my budget?

Servers

What size servers are right for my budget?

Servers

How many servers should I budget for?

What size servers are right for my budget?

Servers

How can I control access from my servers?

How will the application handle server hardware failure?

How many servers should I budget for?

Servers

How should my app withstand a server failing?

How can I tell if a server has been compromised?

How can I increase utilization of my servers?

Which OS should my servers run?

How much remaining capacity do my servers have?

How should I implement dynamic configuration changes on my servers?

When should I decide to scale up my servers?

What size servers are right for my budget?

How will I keep my server OS patched?

How can I control access from my servers?

Which packages should be baked into my server images?

(Arghhhhhhhh!)

How will new code be deployed to my servers?

What size server is right for my performance?

How will the application handle server hardware failure?

How many users create too much load for my servers?

Which users should have access to my servers?

Should I tune OS settings to optimize my application?

How many servers should I budget for?

When should I decide to scale out my servers?

aws

How should my app withstand a server failing?

How can I tell if a server has been compromised?

How can I increase utilization of my servers?

Which OS should my servers run?

How much remaining capacity do my servers have?

How should I implement dynamic configuration changes on my servers?

When should I decide to scale up my servers?

What size servers are right for my budget?

How will I keep my server OS patched?

In most cases, developer's time is spend on the operation and maintenance of applications, rather than the actual business problem

How will the application handle server hardware failure?

Which users should have access to my servers?

Should I tune OS settings to optimize my application?

How many users create too much load for my servers?

When should I decide to scale out my servers?

How many servers should I budget for?

aws

Hello serverless!

What serverless means ?

No servers to provision or manage

Scales with usage

Never pay for idle

Availability and fault tolerance built in

AWS Lambda

AWS Lambda

Event-driven compute

Function as a Service (FaaS)

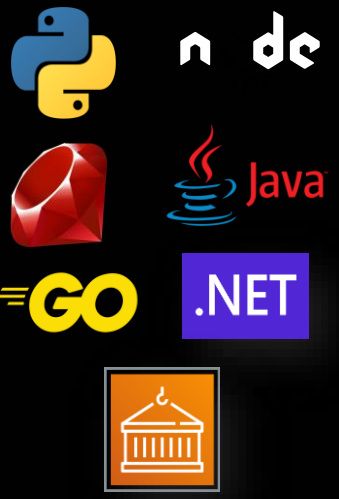
Serverless FaaS

Serverless applications

Function

Serverless applications

Function



Serverless applications

Event Source

Function

Changes in resource state

Requests to endpoints

Changes in data state

aws

Serverless applications

Event Source

Function

Services
(anything)

Changes in resource state

Requests to endpoints

Changes in data state

aws

Anatomy of Lambda Function

```python
import json

def lambda_handler(event, context):

    message = f"Hello {event['first_name']}, welcome to the wrold of serverless!"

    return {
        'statusCode': 200,
        'message' : json.dumps(message)
    }
```

Anatomy of Lambda Function

Handler function

Function to be executed upon invocation

```python
import json

def lambda_handler(event, context):

    message = f"Hello {event['first_name']}, welcome to the wrold of serverless!"

    return {
        'statusCode': 200,
        'message' : json.dumps(message)
    }
```

Anatomy of Lambda Function

Event object

Data sent during Lambda Function Invocation

Handler function

Function to be executed upon invocation

```python
import json

def lambda_handler(event, context):

    message = f"Hello {event['first_name']}, welcome to the wrold of serverless!"

    return {
        'statusCode': 200,
        'message' : json.dumps(message)
    }
```

Anatomy of Lambda Function

Event object

Data sent during Lambda Function Invocation

Context object

Methods available to interact with runtime information (request ID, log group, etc.)

Handler function

Function to be executed upon invocation

```
import json

def lambda_handler(event, context):

    message = f"Hello {event['first_name']}, welcome to the wrold of serverless!"

    return {
        'statusCode': 200,
        'message' : json.dumps(message)
    }
```

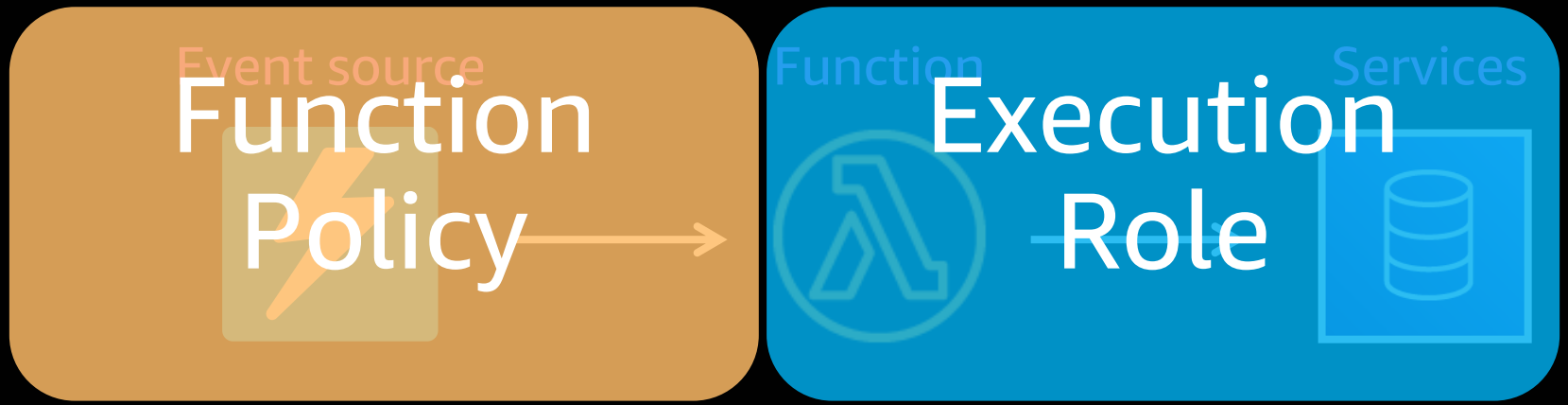
AWS Lambda Permissions Model

Function policies:

- “Actions on bucket X can invoke Lambda function Z"
- Resource policies allow for cross account access
- Used for sync and async invocations

Execution role:

- “Lambda function A can read from DynamoDB table users”
- Define what AWS resources/API calls can this function access via IAM
- Used in streaming invocations



Demo

AWS Amplify

AWS Cloud Development Kit (AWS CDK)

Serverless Application Model (SAM)

SAM comes in 2 parts

SAM comes in 2 parts

SAM Template

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 templates

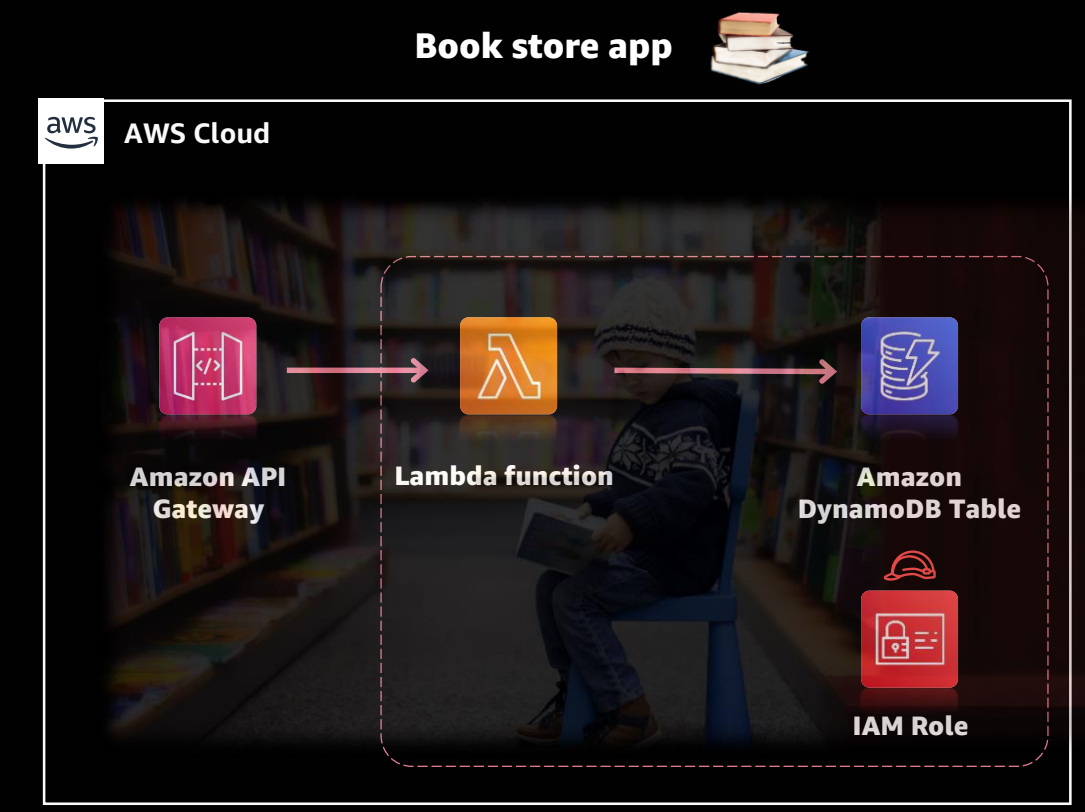
```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:

  AddBookFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: books_lib_app/
      Handler: app.lambda_handler
      Runtime: python3.8
      Events:
        ListBooks:
          Type: Api
          Properties:
            Path: /books
            Method: get
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref BooksTable
  BooksTable:
    Type: AWS::Serverless::SimpleTable
```

In 21 lines it creates:

- Serverless compute (Lambda function)
- Application integration (API Gateway)
- Security permission (IAM role)
- Persistent data store (DynamoDB table)

SAM templates

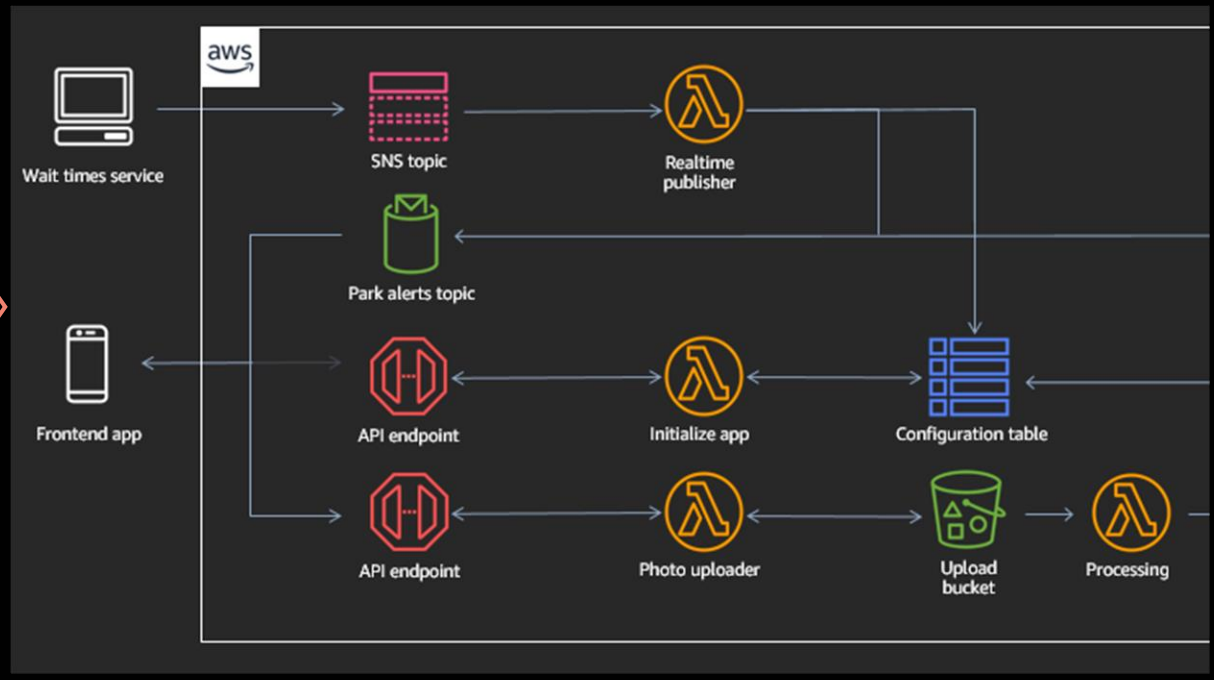
```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:

  AddBookFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: books_lib_app/
      Handler: app.lambda_handler
      Runtime: python3.8
      Events:
        ListBooks:
          Type: Api
          Properties:
            Path: /books
            Method: get
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref BooksTable
  BooksTable:
    Type: AWS::Serverless::SimpleTable
```



Demo

(This time with SAM)

Stitch and build



Small pieces, loosely joined

Serverless is more than compute

Serverless is more than compute

COMPUTE


AWS Lambda


AWS Fargate

DATA STORES


Amazon Simple Storage Service (Amazon S3)


Amazon Aurora Serverless


Amazon DynamoDB

INTEGRATION


Amazon EventBridge


Amazon API Gateway

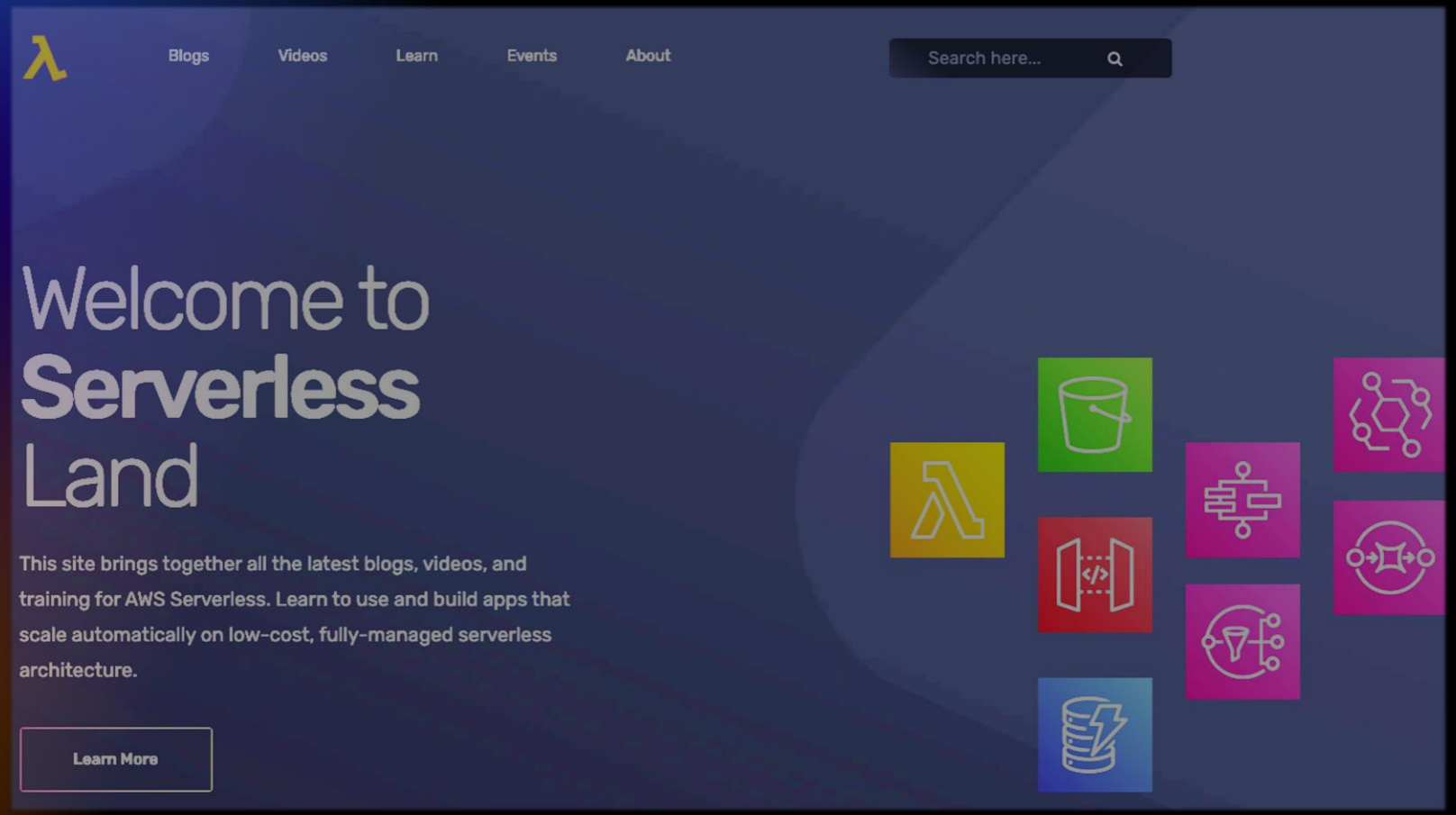

Amazon Simple Queue Service (Amazon SQS)


Amazon Simple Notification Service (Amazon SNS)


AWS Step Functions


AWS AppSync

Start building with AWS



https://serverlessland.com

Visit the Modern Applications Resource Hub for more resources

Dive deeper with these resources to help you develop an effective plan for your modernization journey.

- Build modern applications on AWS e-book
- Build mobile and web apps faster e-book
- Modernize today with containers on AWS e-book
- Adopting a modern Dev+Ops model e-book
- Modern apps need modern ops e-book
- Determining the total cost of ownership: Comparing Serverless and Server-based technologies paper
- Continuous learning, continuous modernization e-book
- … and more!

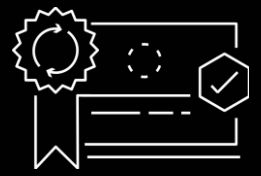
https://bit.ly/3yfOvbK

Visit resource hub »

AWS Training and Certification

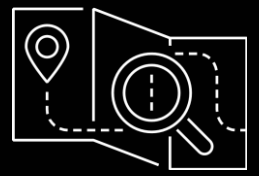
Accelerate modernization with continuous learning

Free digital courses, including:
Architecting serverless solutions
Getting started with DevOps on AWS



Earn an industry-recognized credential:
AWS Certified Developer – Associate
AWS Certified DevOps – Professional

Hands-on classroom training (available virtually) including:
Running containers on Amazon Elastic Kubernetes Service (Amazon EKS)
Advanced developing on AWS



Create a self-paced learning roadmap
AWS ramp-up guide - Developer
AWS ramp-up guide - DevOps

Take Developer and DevOps training today

Learn more about Modernization training for you and your team

Thank you for attending AWS Innovate Modern Applications Edition

We hope you found it interesting! A kind reminder to **complete the survey.**
Let us know what you thought of today's event and how we can improve the event experience for you in the future.


aws-apj-marketing@amazon.com


twitter.com/AWSCloud


facebook.com/AmazonWebServices


youtube.com/user/AmazonWebServices


slideshare.net/AmazonWebServices


twitch.tv/aws

Thank you!