



AWS
re:Invent

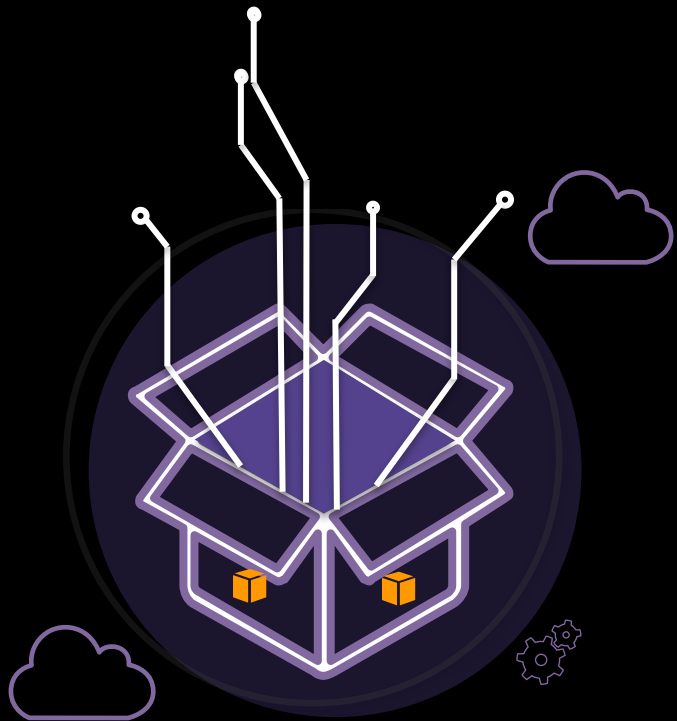
What's New with AWS Lambda

Julien Simon, Principal Technical Evangelist, AWS

julsimon@amazon.fr

@julsimon

Capabilities of a serverless platform



Cloud
Logic Layer



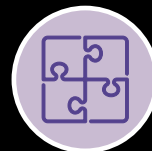
Orchestration and
State Management



Responsive
Data Sources



Application
Modeling
Framework



Developer
Ecosystem



Integrations
Library



Security and
Access Control



Reliability and
Performance



Global
Scale

CI/CD for serverless apps

- New features

- AWS SAM

- SAM in AWS CloudFormation

- Serverless CI/CD pipelines

 - with AWS CodePipeline and AWS

 - CodeBuild

Environment variables for Lambda functions ^{New}

You can define Environment Variables as key/value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more.](#)

Environment variables

var1

value1



var2

value2



Key

Value

```
var AWS = require('aws-sdk');  
  
exports.handler = function(event, context, callback) {  
  var bucketName = process.env.S3_BUCKET;  
  callback(null, bucketName);  
};
```

AWS Serverless Application Model (“SAM”) ^{New}

- A common language for describing the contents of a serverless app.
- CloudFormation now “speaks serverless” with native support for SAM.
- New CloudFormation tools to package and deploy Lambda-based apps.
- Export Lambda blueprints and functions in SAM from the AWS Lambda console.



AWS Serverless Application Model ^{New}

AWSTemplateFormatVersion: '2010-09-09'

Transform: AWS::Serverless-2016-10-31

Resources: GetHtmlFunction:

Type: AWS::Serverless::Function

Properties:

CodeUri: s3://flourish-demo-bucket/todo_list.zip

Handler: index.gethtml

Runtime: nodejs4.3

Policies: AmazonDynamoDBReadOnlyAccess

Events:

GetHtml: Type: Api

Properties: Path: /{proxy+} Method: ANY

ListTable: Type: AWS::Serverless::SimpleTable

Functions



APIs

Storage

AWS Serverless Application Model New

REPLACES:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources: GetHtmlFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: s3://flourish-demo-bucket/todo_list.zip
    Handler: index.gethtml
    Runtime: nodejs4.3
    Policies: AmazonDynamoDBReadOnlyAccess
    Events:
      GetHtml: Type: Api
  Properties: Path: /{proxy+} Method: ANY
  ListTable: Type: AWS::Serverless::SimpleTable
```

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  GetHtmlFunctionGetHtmlPermissionProd:
    Type: AWS::Lambda::Permission
    Properties:
      Action: lambda:invokeFunction
      Principal: apigateway.amazonaws.com
      FunctionName:
        Ref: GetHtmlFunction
      SourceArn:
        Fn::Sub: arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:${ServerlessRestApi}/${ANY}
      ServerlessRestApiProdStage:
        Type: AWS::ApiGateway::Stage
      Properties:
        DeploymentId:
          Ref: ServerlessRestApiDeployment
        RestApiId:
          Ref: ServerlessRestApi
        StageName: Prod
        ListTable:
          Type: AWS::DynamoDB::Table
      Properties:
        ProvisionedThroughput:
          WriteCapacityUnits: 5
          ReadCapacityUnits: 5
      AttributeDefinitions:
        - AttributeName: id
        AttributeType: S
      KeySchema:
        - KeyType: HASH
        AttributeName: id
      GetHtmlFunction:
        Type: AWS::Lambda::Function
      Properties:
        Handler: index.gethtml
        Code:
          S3Bucket: flourish-demo-bucket
          S3Key: todo_list.zip
      Role:
        Fn::GetAtt:
          - GetHtmlFunctionRole
        - Arn
      Runtime: nodejs4.3
      GetHtmlFunctionRole:
        Type: AWS::IAM::Role
      Properties:
        ManagedPolicyArns:
          - arn:aws:iam::aws:policy/AmazonDynamoDBReadOnlyAccess
          - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
        AssumeRolePolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Action:
                - sts:AssumeRole
              Effect: Allow
              Principal:
                Service:
                  - lambda.amazonaws.com
              ServerlessRestApiDeployment:
                Type: AWS::ApiGateway::Deployment
              Properties:
                RestApiId:
                  Ref: ServerlessRestApi
                Description: 'RestApi deployment id: 127e3bf91142ab1ddc5f5446adb094442581a90d'
                StageName: Stage
                GetHtmlFunctionGetHtmlPermissionTest:
                  Type: AWS::Lambda::Permission
```

```
Properties:
  Action: lambda:invokeFunction
  Principal: apigateway.amazonaws.com
  FunctionName:
    Ref: GetHtmlFunction
  SourceArn:
    Fn::Sub: arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:${ServerlessRestApi}/${ANY}
  ServerlessRestApi:
    Type: AWS::ApiGateway::RestApi
  Properties:
    Body:
      info:
        version: '1.0'
        title:
          Ref: AWS::StackName
        paths:
          "/{proxy+}":
            x-amazon-apigateway-any-method:
              x-amazon-apigateway-integration:
                httpMethod: ANY
                type: aws_proxy
                uri:
                  Fn::Sub: arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${GetHtmlFunction.Arn}/invocations
            responses: {}
            swagger: '2.0'
```

SAM: Open Specification ^{New}

A common language to describe the content of a serverless application *across the ecosystem*.

Apache 2.0 licensed
GitHub project

AWS Serverless Application Model (SAM)

Version 2016-10-31

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

The AWS Serverless Application Model (SAM) is licensed under [The Apache License, Version 2.0](#).

Introduction

AWS SAM is a model used to define serverless applications on AWS.

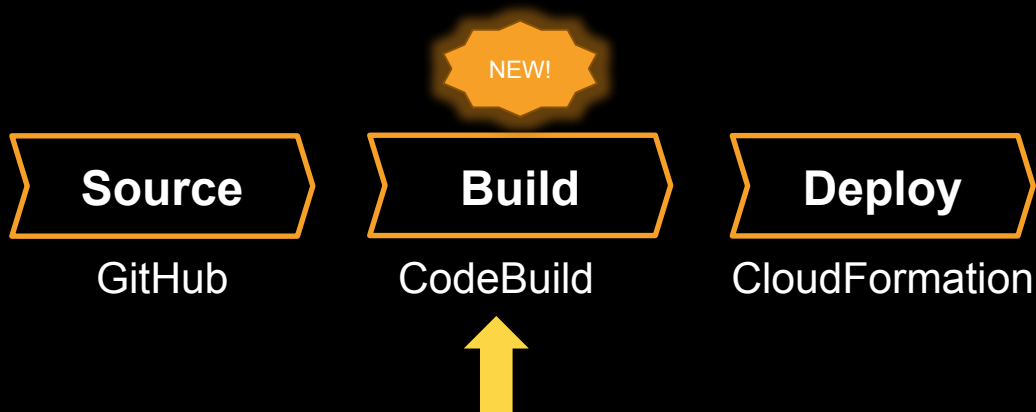
Serverless applications are applications composed of functions triggered by events. A typical serverless application consists of one or more AWS Lambda functions triggered by events such as object uploads to [Amazon S3](#), and API actions. Those functions can stand alone or leverage other resources such as [Amazon DynamoDB](#) buckets. The most basic serverless application is simply a function.

Serverless CI/CD pipeline



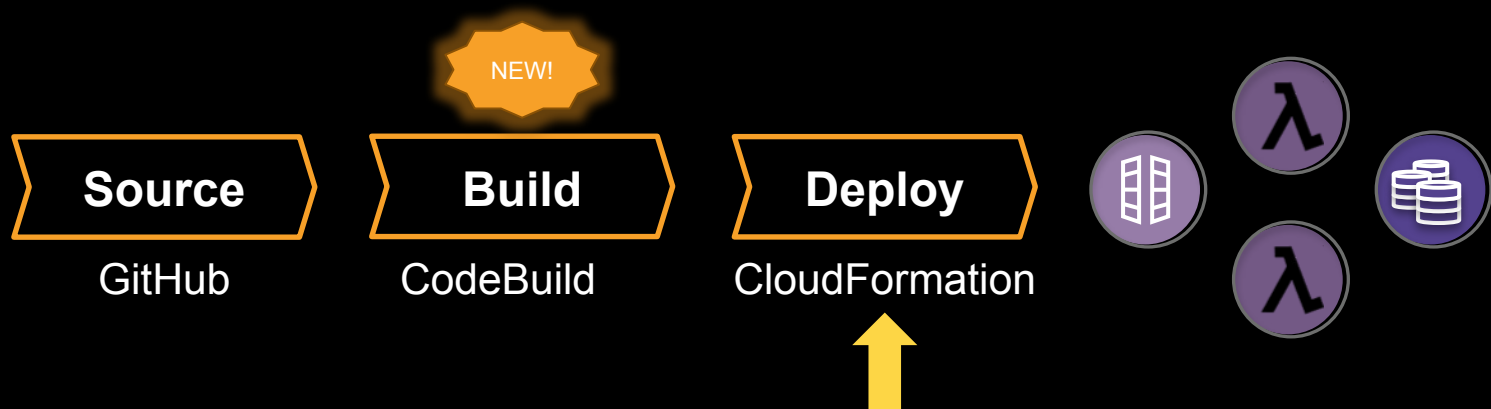
- Pull source directly from GitHub or CodeCommit using CodePipeline

Serverless CI/CD pipeline



- Pull source directly from GitHub or AWS CodeCommit using AWS CodePipeline
- **Build and package serverless apps with AWS CodeBuild**
 - npm, pip, Java compilation, BYO Docker...

Serverless CI/CD pipeline



- Pull source directly from GitHub or AWS CodeCommit using AWS CodePipeline
- Build and package serverless apps with AWS CodeBuild
- **Deploy your completed Lambda app with AWS CloudFormation**

Tracing serverless apps with AWS X-Ray

**How do I diagnose
Lambda apps?**

Introducing X-Ray *Preview*

Gain visibility into events traveling through services

Trace calls and timing from Lambda functions to other AWS services

Easy configuration

Lambda support coming soon



Xray provides tracing and monitoring capabilities for your Lambda function.

Enable active tracing ☐ ⓘ

Easy setup

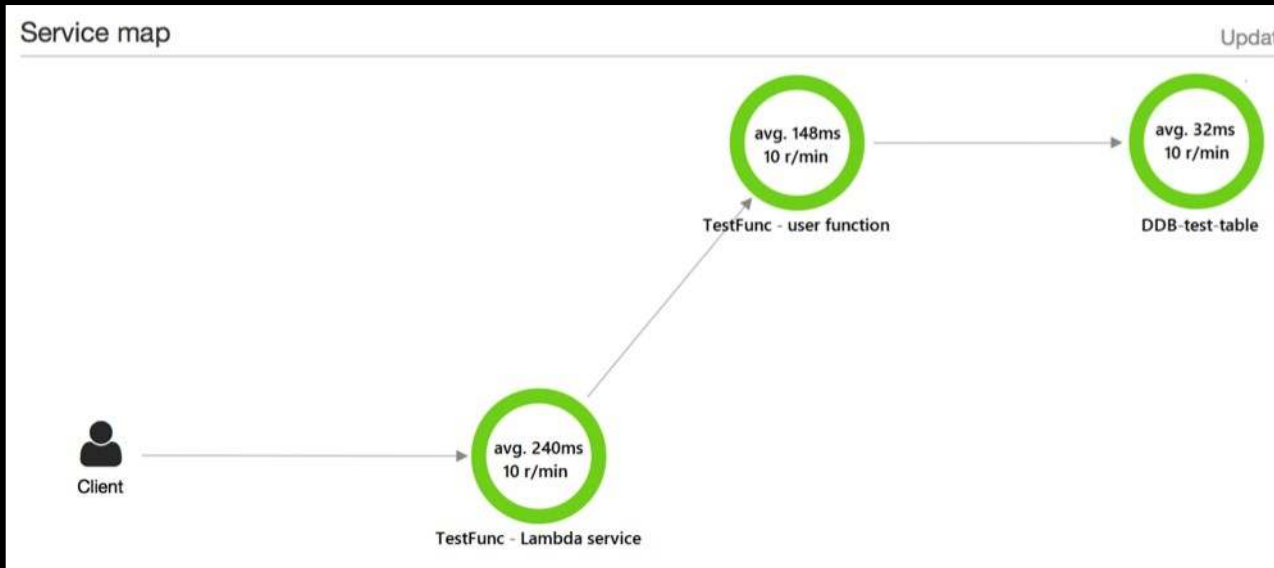
```
1 const AWSXRay = require('aws-xray-sdk');
2
3 exports.handler = AWSXRay.captureLambda((event, context, callback) => {
4   const segment = context.xrayContext.segment;
5   // TODO implement
6   callback(null, 'Hello from Lambda');
7 });
```

Introducing X-Ray *Preview*

View the dynamic topology of your application

See actual dependencies
among microservice
components

Easily detect and
diagnose missing
events and throttles

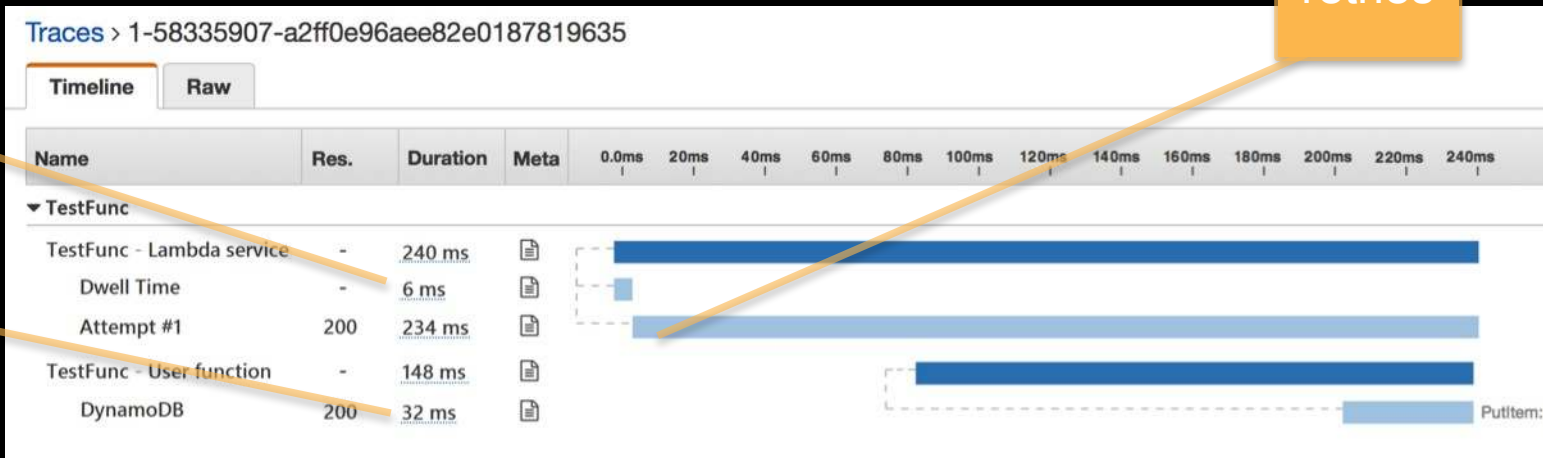


Introducing X-Ray *Preview*

See dwell time and retries for async invokes

Profile performance of calls your code makes to other AWS services

- Detect failures in event processing
- Easily find and fix performance issues



dwell times

service call times

retries

New Lambda features

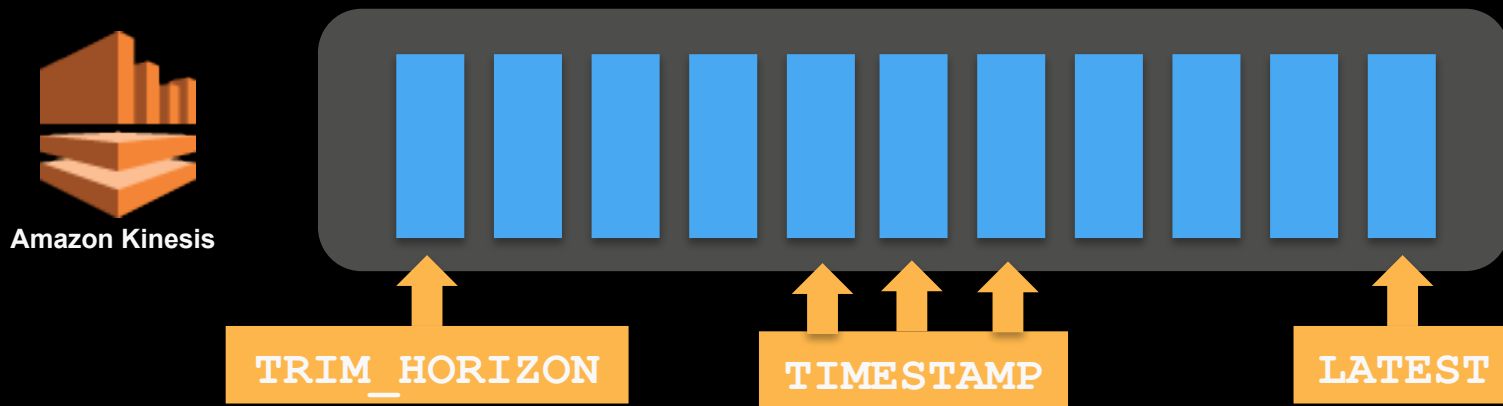
AT_TIMESTAMP Amazon Kinesis iterator

C# with .NET Core

Dead letter queue

AT_TIMESTAMP Amazon Kinesis iterator New

- Process streaming data in Amazon Kinesis at any point in time
- Stop and start processing without rewinding or losing data



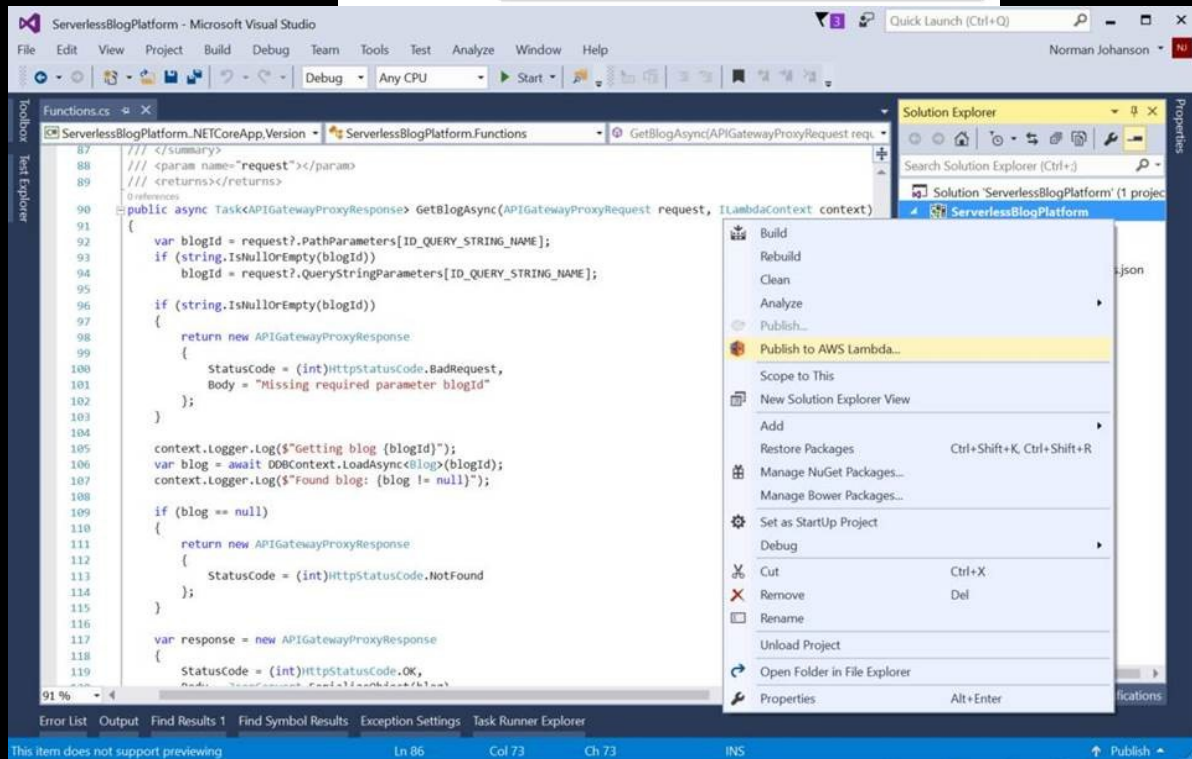
C# and .NET Core ^{New}

- Write Lambda functions in C#
- netcoreapp 1.0 on Amazon Linux
- Built-in logging and metrics
- Supports common AWS event types (S3, SNS)

Name* csharp-lambda

Description My C# Lambda function!

Runtime* C#

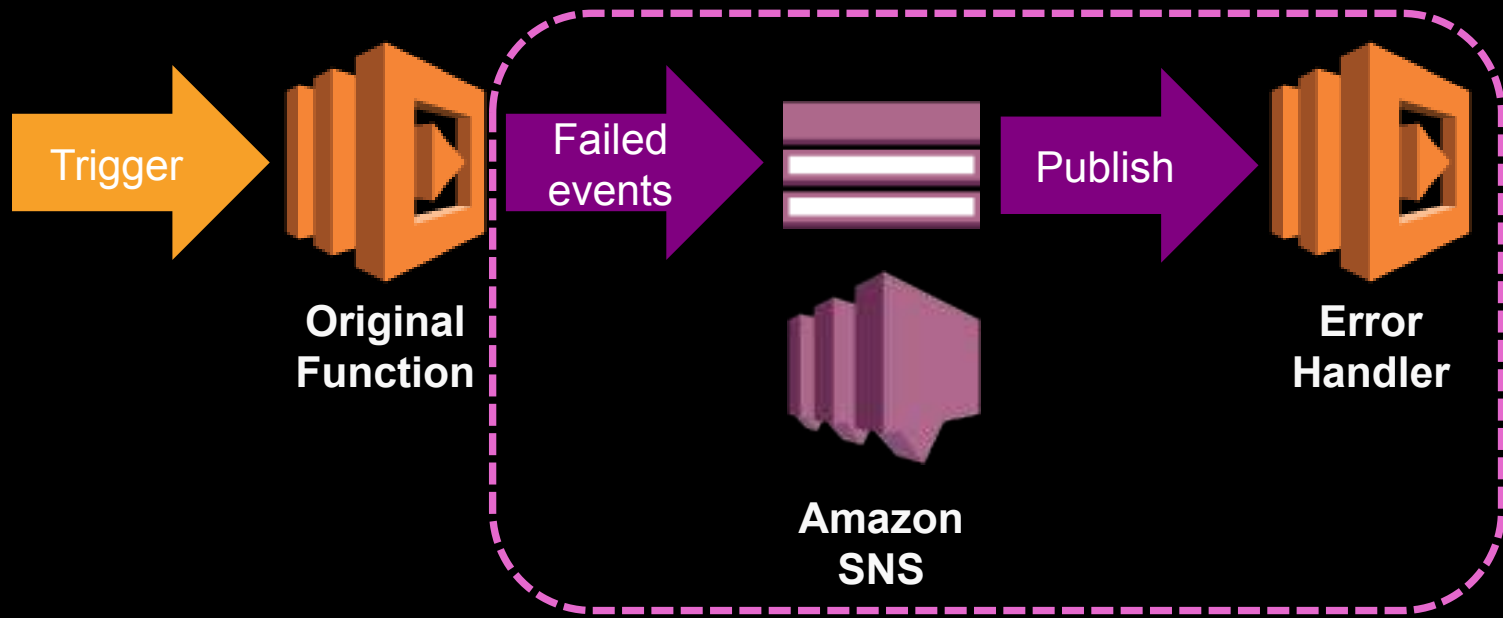


Dead-letter queue for events New

Easily create reliable end-to-end event processing solutions

- Sends all unprocessed events to your SQS queue or SNS topic: 3 strikes rule
- Preserves events *even if your code has an issue* or the call was throttled
- Per-function
- Works for all async invokes, *including S3 and SNS events*





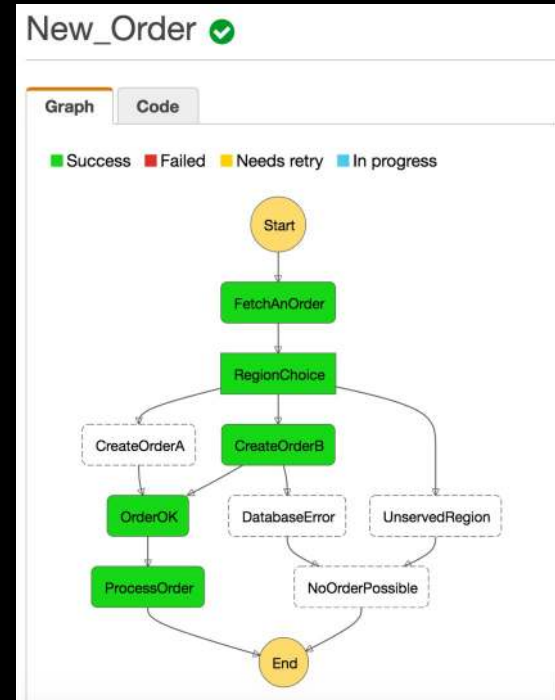
Lambda DLQ in action

Orchestrating Lambda functions

AWS Step Functions **New**

Reliably orchestrate multiple Lambda functions

Attempt a function more than 3X
Add callbacks to asynchronous functions
Handle situations that require waiting
Chain function execution ($A \rightarrow B \rightarrow C$)
Supports long-running workflows



New API Gateway features

Binary encoding

Documentation support

AWS Marketplace SaaS integration

Developer Portal Reference Implementation

Binary encoding ^{New}

Uses Content-Type and Accept headers

Serve images,
audio, and other
binary content

Binary Support

You can configure binary support for your API by specifying which media types should be treated as binary types. API Gateway will look at the **Content-Type** and **Accept** HTTP headers to decide how to handle the body.

Binary media types

- image/gif
- application/octet-stream
- add another here...

Add binary media type **Cancel** **Save**

Automatically base64-encodes Lambda integrations

API documentation ^{New}

- Document your APIs – edit doc parts directly in the API Gateway console
- Swagger import/export – fully round-trip-able
- Supports tech writers – independent update and publish flow

Documentation

Create Documentation PartImport DocumentationPublish Documentation

Add documentation to help developers understand how to interact with your API. Documentation parts can be shared across multiple resources and methods by specifying a wildcard value (*) for method or status code, eg. documentation for a 200 response can be used in multiple locations. You can also import documentation by supplying a Swagger definition file, and publish documentation to a stage. For more information, reference the [documentation](#).

TypeAllPathMethodAllNameStatus Code200

TypeResponse (status code)

Path /

Method *

Status Code200

```
{
  "description": "Successful operation"
}
```

EditClone

TypeResponse Header

Path /

MethodOPTIONS

Status Code200

NameAccess-Control-Allow-Headers

```
{
  "description": "Used in response to a preflight request"
}
```

EditClone

API Gateway and AWS Marketplace integration **New**

- Use API Gateway to simplify building and operating APIs
- Sell your APIs on the AWS Marketplace
- Easy discovery and procurement for your API's consumers
- Track API usage by consumer / key
- Automated billing through AWS



URL Reputation APIs



Speech understanding APIs

Monetize your microservices!

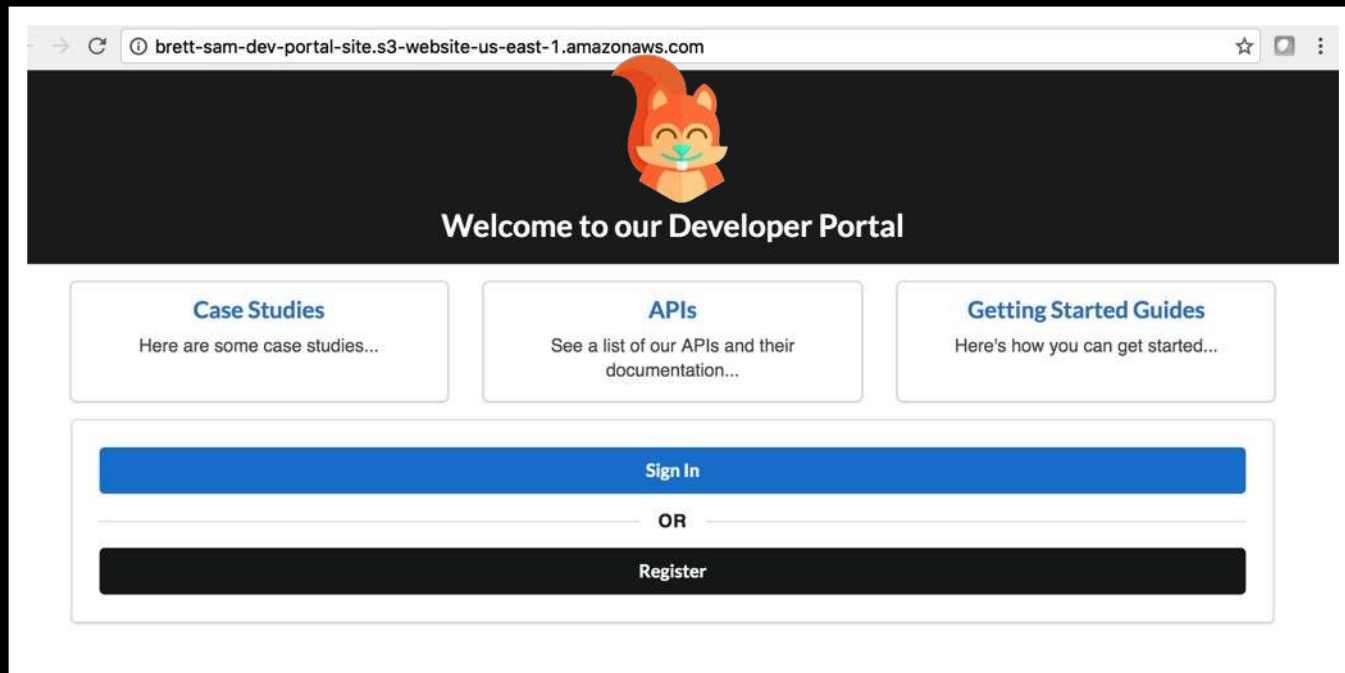
API Gateway Developer Portal

Open source reference implementation

New

SAM-based implementation available on GitHub

Help developers
consume your
APIs
Vend API Keys
AWS Marketplace
integration
Supports Cognito
authN



New places you can use Lambda functions

Lambda Bots

Amazon Kinesis Firehose

On-prem storage

Devices

Edge/CDN

Lambda Bots and Amazon Lex *Preview*

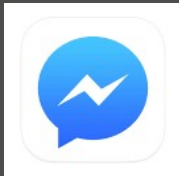
- Text and speech
- Lambda functions run business logic
- Facebook, AWS Mobile Hub
- Slack and Twilio integration coming soon

I'd like to book a hotel

The screenshot displays the Amazon Lex console interface for configuring a bot named 'BookStay'. The left sidebar shows the 'Intents' list with 'BookHotel' selected. The main area shows the 'Sample utterances' and 'Parameters' for the 'BookHotel' intent. A blue speech bubble with the text 'I'd like to book a hotel' points to the first sample utterance. The 'Parameters' table lists required parameters for the intent.

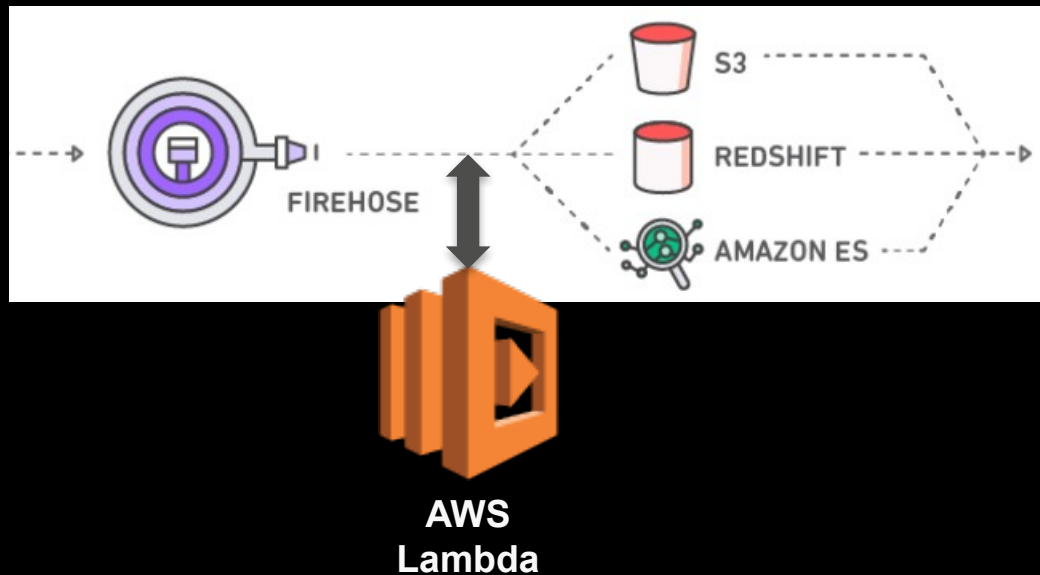
Required	Name	Parameter type	Prompt
<input type="checkbox"/>	e.g. Location	e.g. AMAZON.CITY	e.g. What city?
<input checked="" type="checkbox"/>	CheckinDate	AMAZON.DATE	What day do you want to check in?
<input checked="" type="checkbox"/>	Nights	AMAZON.NUMBER	How many nights will you be staying?
<input checked="" type="checkbox"/>	Location	AMAZON.US_CITY	What city will you be staying in?

Below the console, there is a 'Test App' window showing a chat interface with the bot. The chat history shows the user saying 'I would like to book a hotel' and the bot asking 'What city will you be staying in?'. The user has responded with 'Seattle'. The bot then asks 'What day do you want to check in?'. The user has responded with 'What day do you want to check in?'. The bot then asks 'How many nights will you be staying?'. The user has responded with 'What city will you be staying in?'. The bot then asks 'What type of room would you like, queen, king or del...'. The user has responded with 'Type or speak to your application ...'.



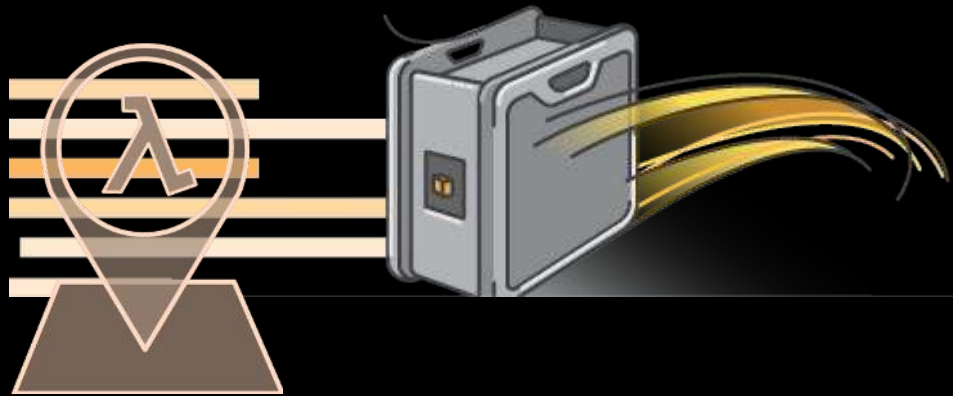
Amazon Kinesis Firehose integration **Coming Soon**

- Simple, real-time data streaming
- Transform, audit, or aggregate records in flight with Lambda
- Flexible buffering
- Lambda and Firehose both scale automatically



AWS Snowball Edge ^{New}

- Fast, simple, secure data transfer from on-prem to/from AWS Cloud
- 100 TB capacity
- Local S3 storage APIs
- **Local Lambda functions**
- Transcode multimedia content, compress in-real time, custom auditing



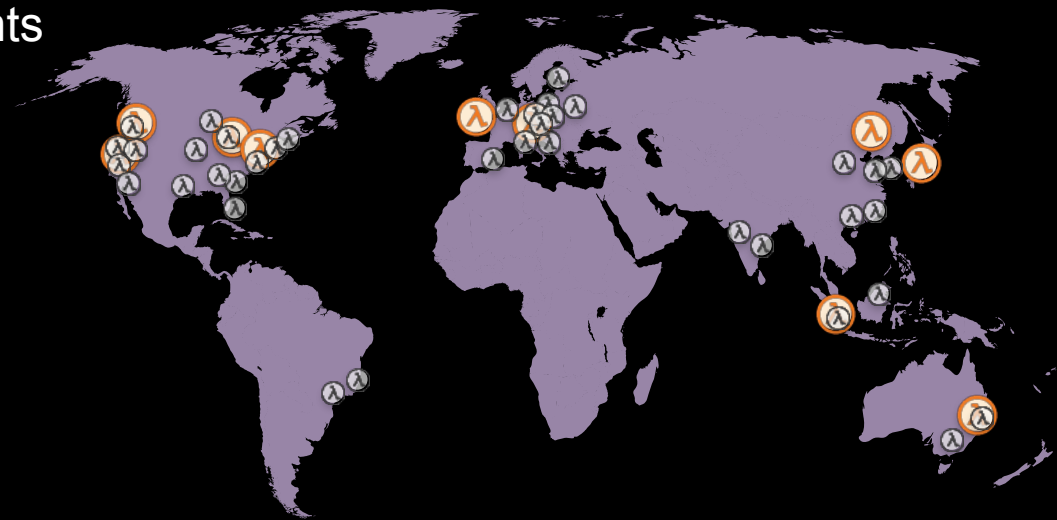
AWS Greengrass *Preview*

- Greengrass extends AWS processing onto devices
- Low-latency, near-real time
- **Lambda functions run right on the device**
- Cloud storage and compute via AWS IoT service
- BYOH – 1GHz, 128MB, x86 or ARM, Linux



Lambda@Edge Preview

- Low-latency request/response customization
- Supports viewer and origin events
- Preview limitations:
 - Node.js only
 - **50** ms max
 - Headers only
- Pricing: \$0.60/M requests and \$0.00000625125 per 128MB-s
 - 4K requests free/month

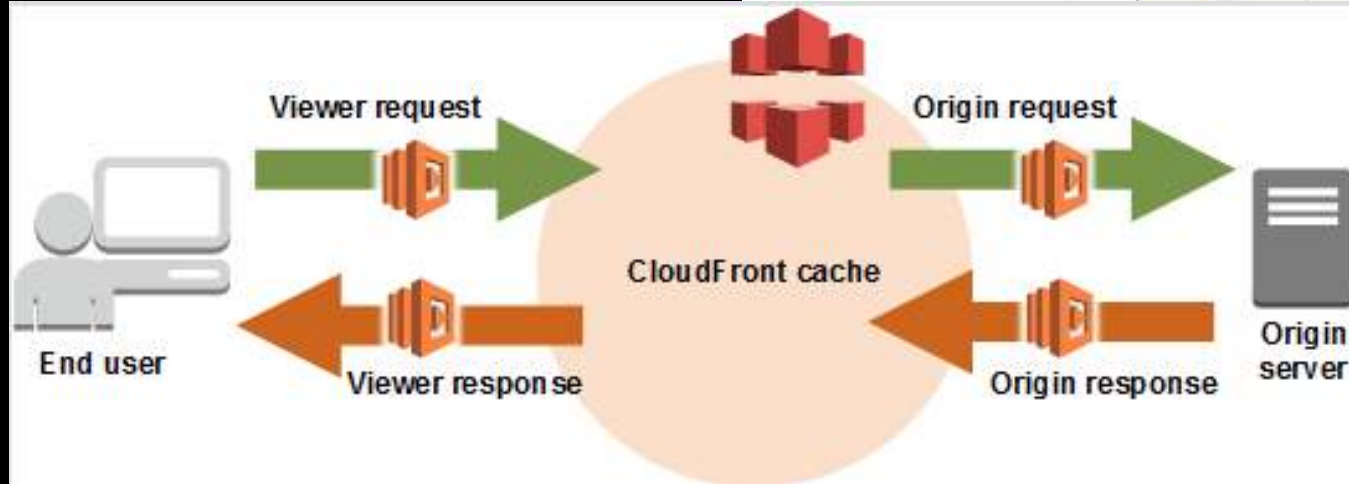
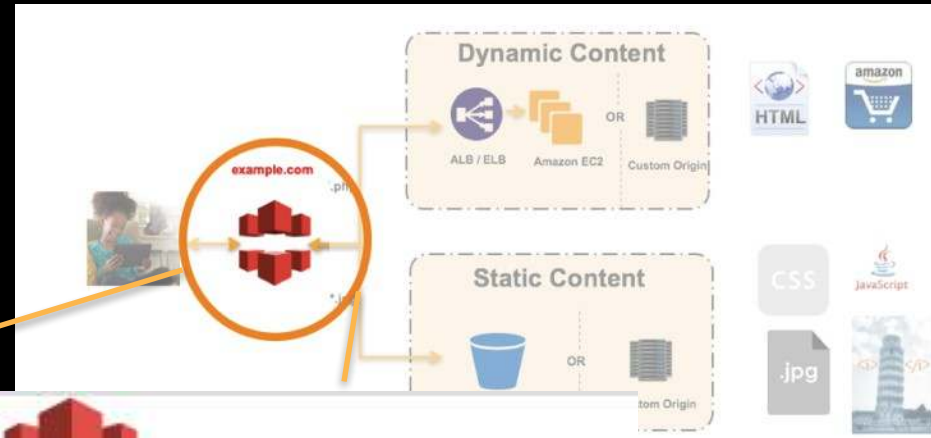


Sign up to join the preview!

Lambda@Edge use cases

- **Customize content**, based on user/device properties
- **Validate visitors**: check tokens, filter bot traffic
- **Rewrite URLs**: inject ads, hide file structure
- **Run A/B testing**

CloudFront Triggers for Lambda@Edge Functions



New Lambda videos from re:Invent 2016

AWS re:Invent 2016: What's New with AWS Lambda (SVR202) <https://www.youtube.com/watch?v=CwxWhyGteNc>

AWS re:Invent 2016: Serverless Apps with AWS Step Functions (SVR201) <https://www.youtube.com/watch?v=75MRve4nv8s>

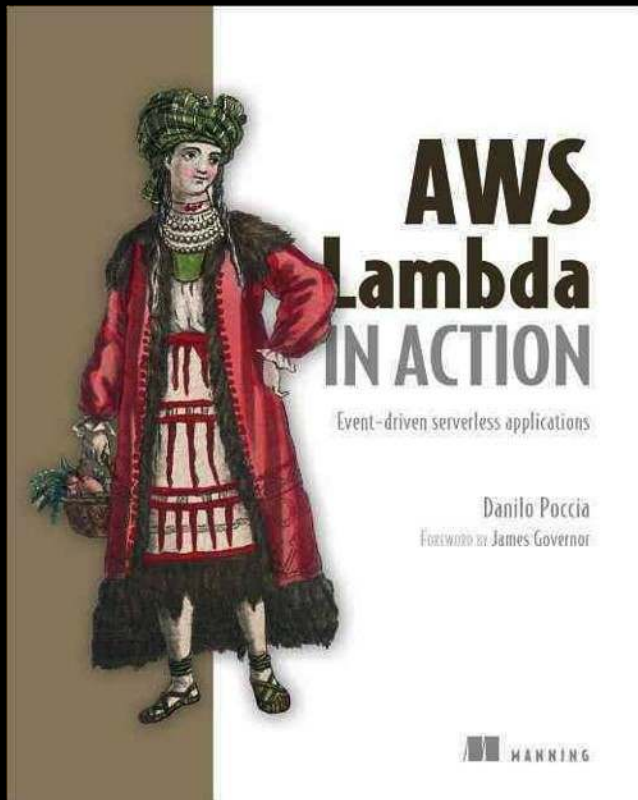
AWS re:Invent 2016: Real-time Data Processing Using AWS Lambda (SVR301) <https://www.youtube.com/watch?v=VFLKOy4GKXQ>

AWS re:Invent 2016: Serverless Architectural Patterns and Best Practices (ARC402) <https://www.youtube.com/watch?v=b7UMoc1iUYw>

AWS re:Invent 2016: Bringing AWS Lambda to the Edge (CTD206) <https://www.youtube.com/watch?v=j26novaqF6M>

AWS re:Invent 2016: Ubiquitous Computing with Greengrass (IOT201) <https://www.youtube.com/watch?v=XQQjX8GTEko>

The only Lambda book you need to read

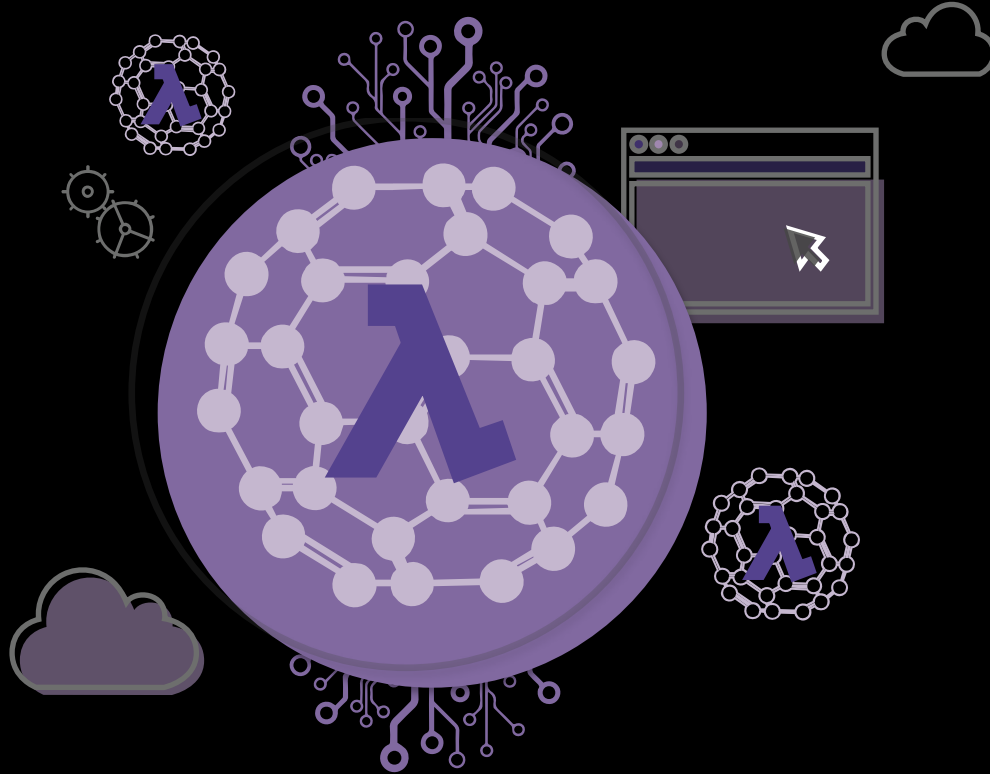


Written by AWS Technical
Evangelist Danilo Poccia

Released in December 2016

<https://www.amazon.com/Aws-Lambda-Action-Event-driven-Applications/dp/1617293717/>

Enjoy your serverless journey!





AWS
re:Invent

Thank you!

julsimon@amazon.fr
@julsimon