AWS re:Invent

SVS321-R

AWS Lambda layers deep dive

Nitin Vashishtha
Solutions Architect
Amazon Web Services

AWS re:Invent

aws

Serverless applications

Event source

Changes in data state

Requests to endpoints

Changes in resource state

Function

Node.js
Python
Java
C#
Go
Ruby
Runtime API

Services

Anatomy of an AWS Lambda function

Handler() function

Function to be executed upon invocation

Event object

Data sent during Lambda function invocation

Context object

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

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello World!')
    }
```

Anatomy of an AWS Lambda function

```
Function myhandler(event, context) {
    <Event handling logic> {
           result = SubfunctionA()
      } else {
             result = SubfunctionB()

    return result;
}

Function subFunctionA(thing){
 ## logic here
}

Function subFunctionB(thing){
 ## logic here
}
```

Anatomy of an AWS Lambda function

```
Function myhandler(event, context) {
    <Event handling logic> {
           result = SubfunctionA()
      } else {
             result = SubfunctionB()

    return result;
}
```

```
Function subFunctionA(thing){
 ## logic here
}

Function subFunctionB(thing){
 ## logic here
}
```

Functions will then grow in complexity with business logic sub-functions

Anatomy of an AWS Lambda function

```
Import sdk
Import http-lib
Import ham-sandwich

Pre-handler-secret-getter()
Pre-handler-db-connect()

Function myhandler(event, context) {
    <Event handling logic> {
                result = SubfunctionA()
            }else {
                        result = SubfunctionB()

    return result;
}

Function Pre-handler-secret-getter() {
}

Function Pre-handler-db-connect(){
}
```

```
Function subFunctionA(thing){
 ## logic here
}

Function subFunctionB(thing){
 ## logic here
}
```

Business logic sub-functions

Anatomy of an AWS Lambda function

```
Import sdk
Import http-lib
Import ham-sandwich
Pre-handler-secret-getter()
Pre-handler-db-connect()
```

Dependencies, configuration information, common helper functions

```
Function myhandler(event, context) {
    <Event handling logic> {
                result = SubfunctionA()
            }else {
                        result = SubfunctionB()

    return result;
}
```

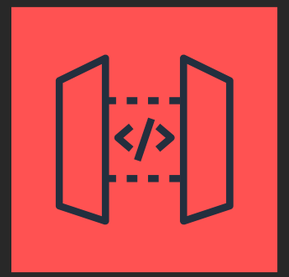
```
Function Pre-handler-secret-getter() {
}

Function Pre-handler-db-connect(){
}
```

Common helper functions

```
Function subFunctionA(thing){
 ## logic here
}

Function subFunctionB(thing){
 ## logic here
}
```

Business logic sub-functions

Anatomy of a serverless application



Amazon API Gateway

/orders
/forums
/search
/lists
/user
/...

Now, assume that I have an API-based workload

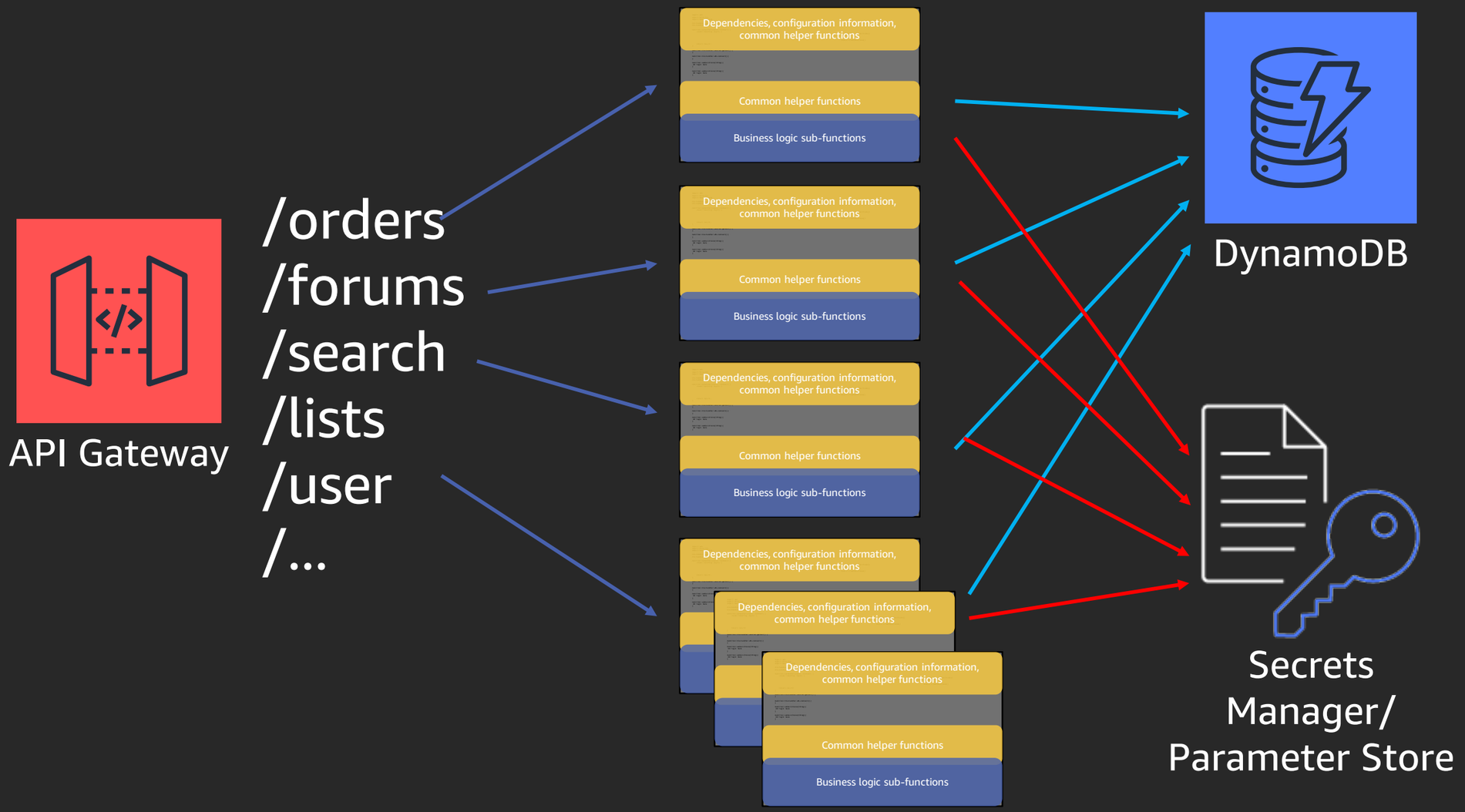
Anatomy of a serverless application

It needs to read/write to a database and get keys and configuration from external services

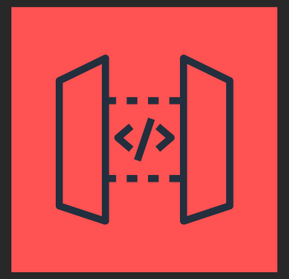
Amazon DynamoDB

AWS Secrets Manager/ Parameter Store

Anatomy of a serverless application



Anatomy of a serverless application



API Gateway

/orders
/forums
/search
/lists
/user
/...

There could be a lot of duplicated code here!

DynamoDB

Secrets Manager/ Parameter Store

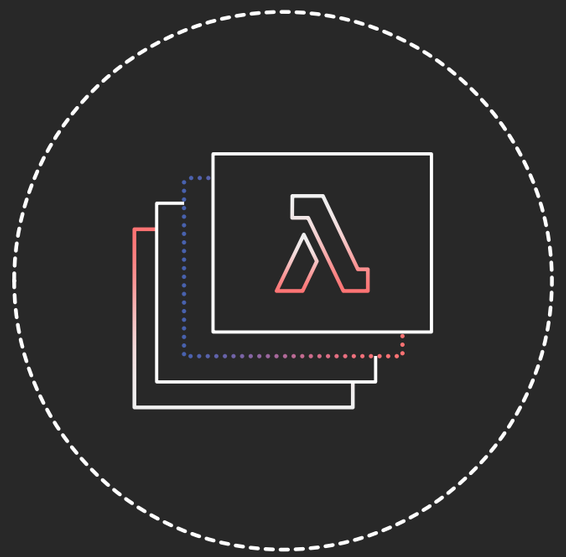
Anatomy of a serverless application

We want something more like this

DynamoDB

Secrets Manager/ Parameter Store

Lambda layers



Let functions easily share code; upload layer once, reference within any function

Layer can be anything: Dependencies, training data, configuration files, etc.

Promote separation of responsibilities and let developers iterate faster on writing business logic

Built-in support for secure sharing by ecosystem

Lambda Layers Use Cases

- Custom code, that is used by more than one function

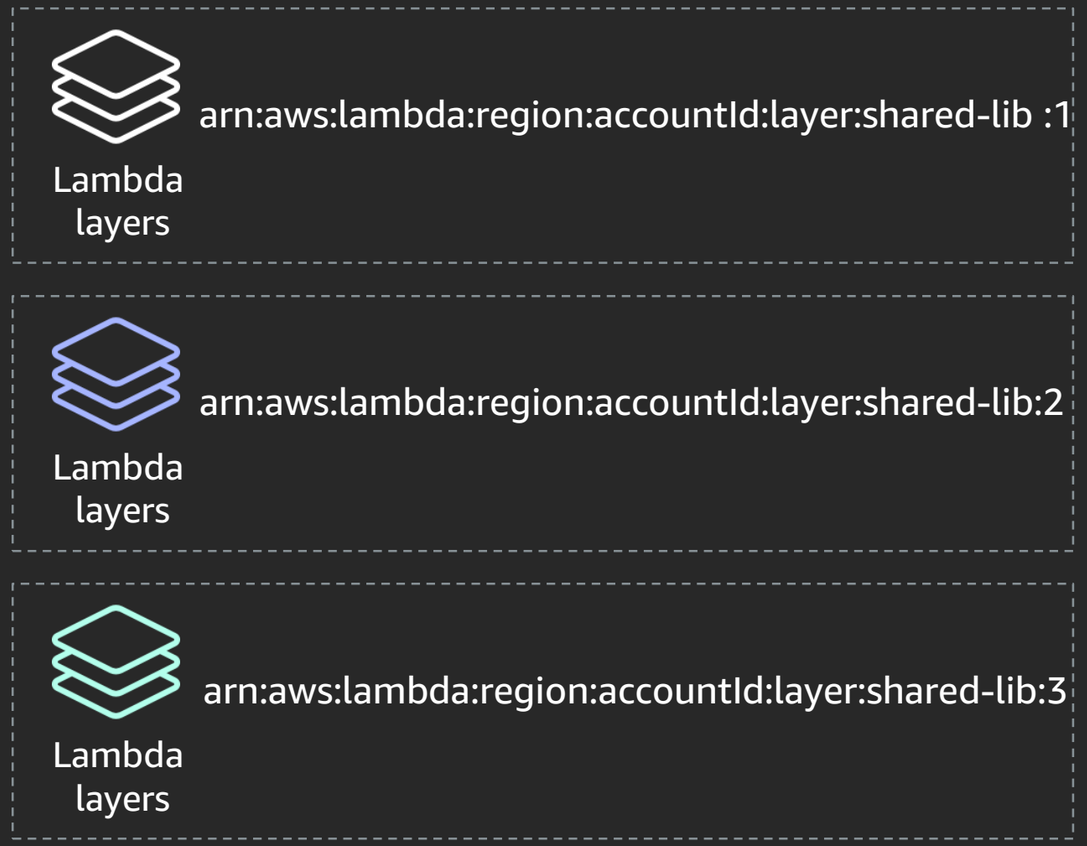
- Libraries, modules, frameworks to simplify the implementation of your business logic

Lambda Layers Benefits

- Enforce separation of concerns, between dependencies and your custom business logic
- Make your function code smaller and more focused on what you want to build
- Speed up deployments, because less code must be packaged and uploaded, and dependencies can be reused

Using Lambda layers

- Put common components in a zip file and upload it as a Lambda layer
- Layers are immutable and can be versioned to manage updates
- When a version is deleted or permissions to use it are revoked, functions that used it previously will continue to work, but you won’t be able to create new ones
- You can reference up to five layers, one of which can optionally be a custom runtime



How Lambda layers work

Order is important because each layer is a zip file, and they are all extracted in the same path

- /opt
- Each layer can potentially overwrite the previous one

This approach can be used to customize the environment

For example, the first layer can be a custom runtime, and the second layer can add specific versions of the libraries that you need

The storage of your Lambda layers takes part in the Lambda function storage per region limit (75 GB)

Including library dependencies in a layer

Runtime	Folders
Node.js	nodejs/node_modules nodejs/node8/node_modules (NODE_PATH)
Python	python python/lib/python3.7/site-packages (site directories)
Java	java/lib (CLASSPATH)
Ruby	ruby/gems/2.5.0 (GEM_PATH) ruby/lib (RUBY_LIB)
All	bin (PATH) lib (LD_LIBRARY_PATH)

Lambda Layers Permissions

- Layers can be used within
  - An AWS account
  - Shared between accounts
  - Shared publicly with the broad developer community
- AWS is publishing a public layer which includes NumPy and SciPy, two popular scientific libraries for Python
  - This prebuilt and optimized layer can help you start very quickly with data processing and machine learning applications

Lambda Layers API

- Updated
  - CreateFunction
  - UpdateFunctionConfiguration
- New
  - ListLayers
  - ListLayerVersions
  - PublishLayerVersion
  - DeleteLayerVersion
  - GetLayerVersion
  - GetLayerVersionPolicy
  - AddLayerVersionPermission
  - RemoveLayerVersionPermission

Model function environments with AWS Serverless Application Model (SAM)

- Open-source framework for building serverless applications on AWS
- Shorthand syntax to express functions, APIs, databases, and event source mappings
- Transforms and expands SAM syntax into AWS CloudFormation syntax on deployment
- Supports all AWS CloudFormation resource types

https://aws.amazon.com/serverless/sam/

SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.get
      Runtime: nodejs6.10
      CodeUri: src/
      Policies: AmazonDynamoDBReadOnlyAccess
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /resource/{resourceId}
            Method: get
```

Shorthand syntax to express functions, tables, and events

SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.get
      Runtime: nodejs6.10
      CodeUri: src/
      Policies: AmazonDynamoDBReadOnlyAccess
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /resource/{resourceId}
            Method: get
```

Just 18 lines of SAM syntax creates many resources:

- Lambda function
- API Gateway
- Amazon DynamoDB table
- AWS Identity and Access Management (IAM) roles

AWS SAM template properties

AWS::Serverless::Function

AWS::Serverless::Api

AWS::Serverless::SimpleTable

AWS::Serverless::LayerVersion

AWS::Serverless::Application

From AWS SAM version 2016-10-31

```
Properties:
  LayerName: MyLayer
  Description: Layer description
  ContentUri: 's3://my-bucket/my-layer.zip'
  CompatibleRuntimes:
    - nodejs6.10
    - nodejs8.10
  LicenseInfo: 'MIT-0 license.'
  RetentionPolicy: Retain
```

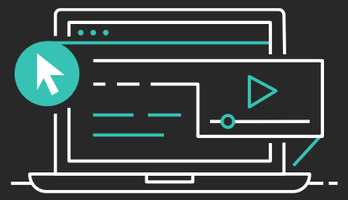
Layers in the AWS Serverless Application Model (SAM)

```
AWSTemplateFormatVersion: '2010-09-09’
Transform: AWS::Serverless-2016-10-31
Description: Sample SAM Template using Layers

Globals:
   Function:
      Timeout: 600

Resources:
  OneLayerVersionServerlessFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: app.one_layer_hanlder
      Runtime: python3.6
      CodeUri: hello_world
      Layers:
        - <LayerOneVersionArn>
        - <LayerTwoVersionArn>
```

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      Description: Layer description
      ContentUri: 's3://my-bucket/my-layer.zip’
      CompatibleRuntimes:
        - nodejs6.10
        - nodejs8.10
      LicenseInfo: 'Available under the MIT-0 license.’
      RetentionPolicy: Retain
```

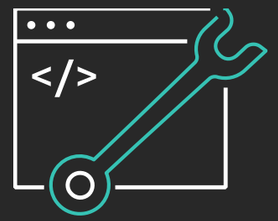
Learn Serverless with AWS Training and Certification

Resources created by the experts at AWS to help you learn modern application development



Free, on-demand courses on serverless, including:

- Introduction to Serverless Development
- Getting in the Serverless Mindset
- AWS Lambda Foundations
- Amazon API Gateway for Serverless Applications
- Amazon DynamoDB for Serverless Architectures



Additional digital and classroom trainings cover modern application development and computing

Visit the Learning Library at https://aws.training

aws training and certification


We want
code!

Demo

Q&A

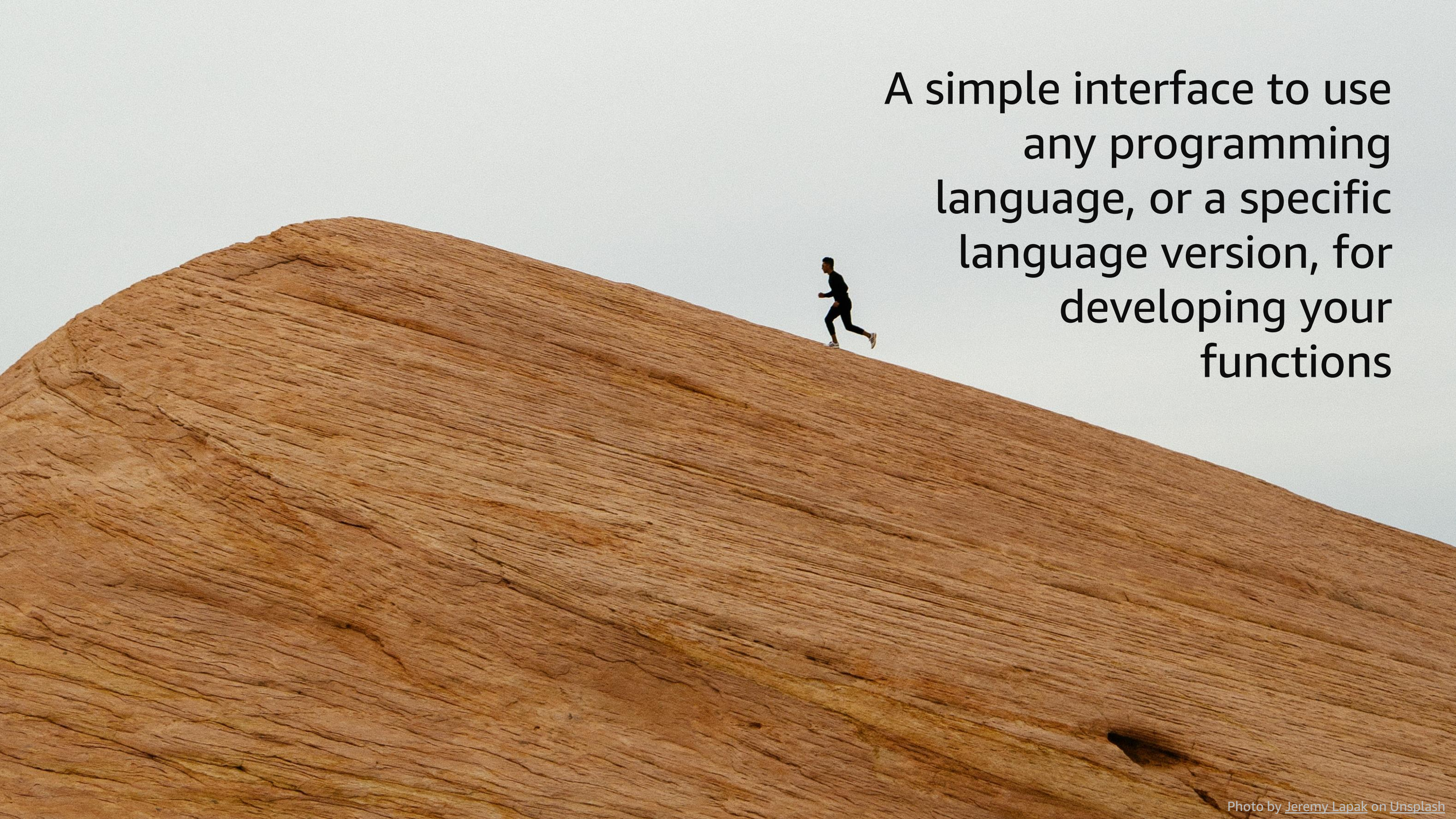
Appendix

Lambda Runtime API

AWS re:Invent

aws

A simple interface to use any programming language, or a specific language version, for developing your functions



Lambda Runtime API

- You can now select a **custom runtime** in the console (**provided** in the API/SDKs/CLI) as the runtime of a Lambda function

- With this selection, the function must include (in its code or in a layer) an executable file called **bootstrap**

  - The runtime bootstrap is responsible for the communication between your code and the Lambda environment

  - Your code can use any programming language

Open Source Runtimes

- C++ (AWS)
- Rust (AWS)

- Erlang (AlertLogic)
- Elixir (AlertLogic)
- Cobol (Blu Age)
- Node.js (NodeSource N|Solid)
- PHP (Stackery)

Please complete the session survey in the mobile app.

Thank you!

Nitin Vashishtha

nvashish@amazon.com

AWS re:Invent

aws