# Serverless computing with Rust

Shing Lyu (呂行)

2019/8/17 COSCUP

# Agenda

- What is Serverless?
- Serverless in AWS
- High level architecture
- Rust runtime for AWS Lambda
  - Lambda-http vs lambda-aws
- Exposing the API with API Gateway
- Accessing the database
- Logging
- Performance critical tasks

# About Me

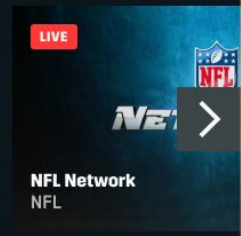
- 呂　行 Shing Lyu

- Backend Engineer @ DAZN
- Author of *Building Reusable Code with Rust*
- https://shinglyu.com/

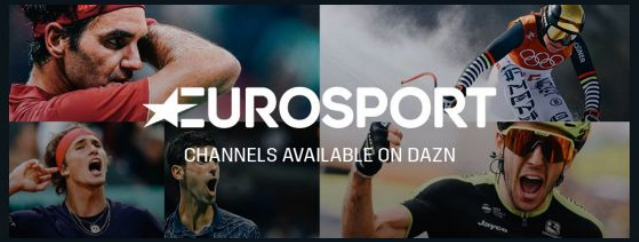
## What's on







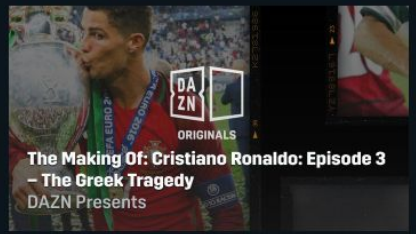

## Don't miss



## Making of



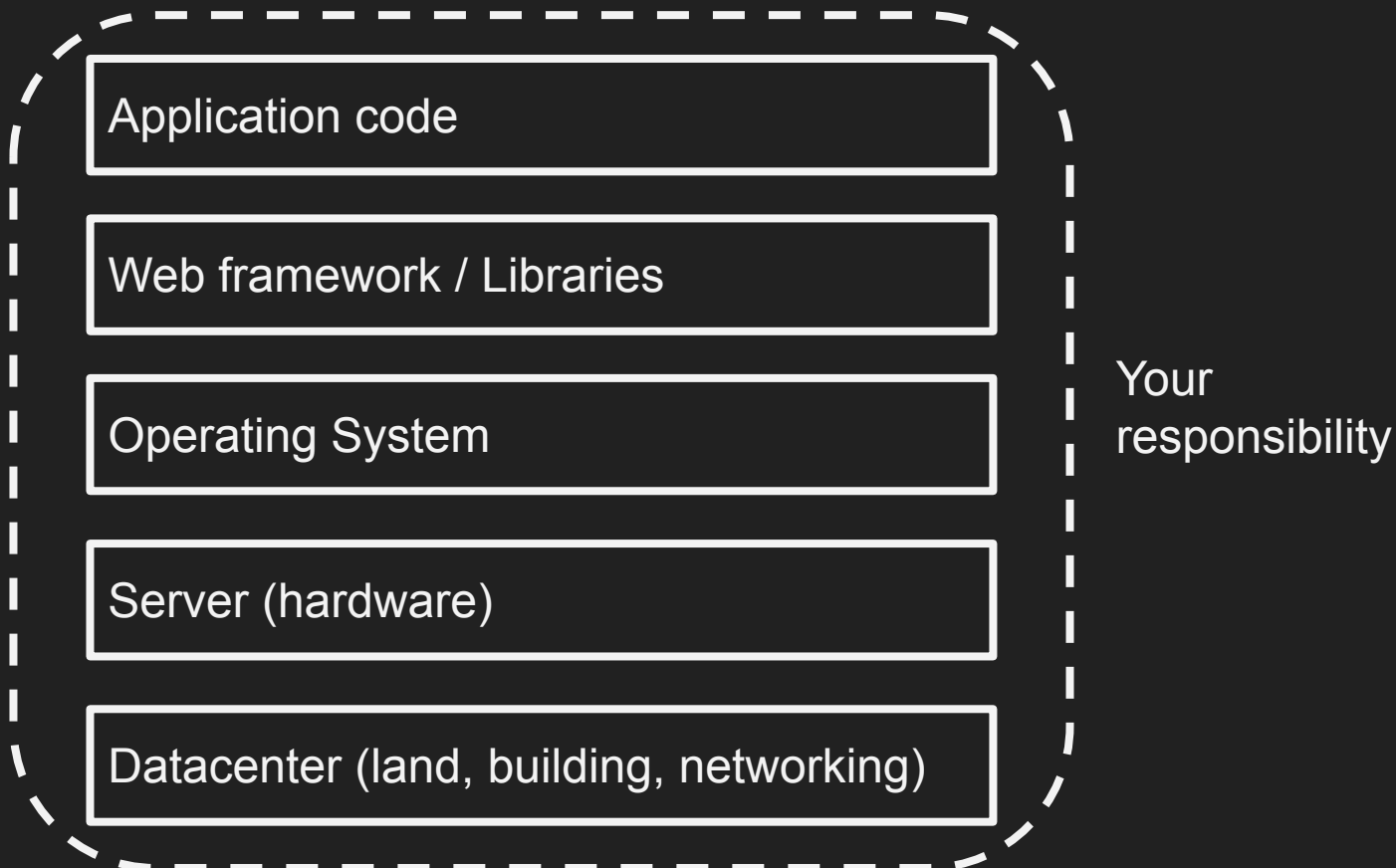




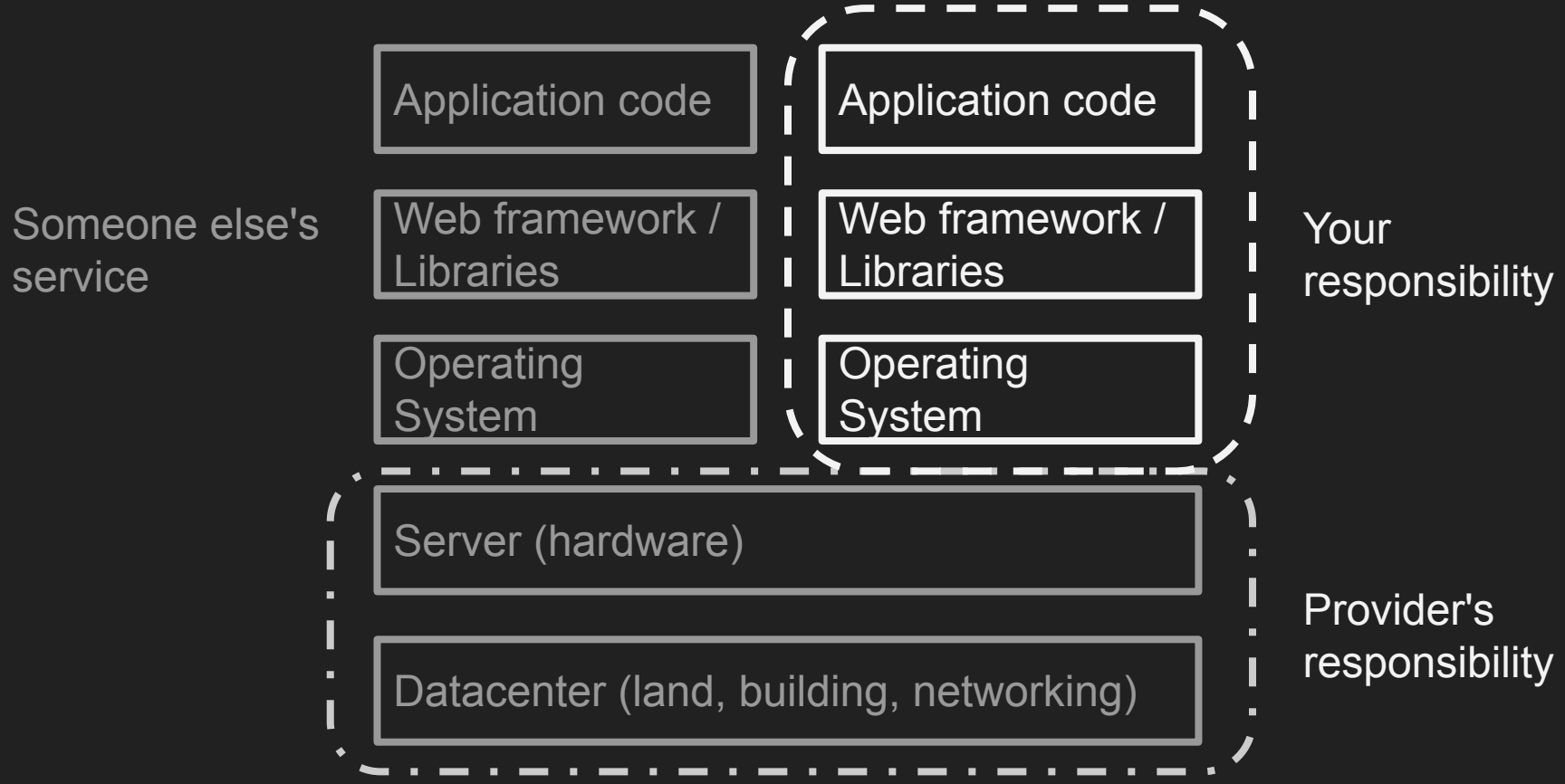

## #FeelMehr


Estonia
Baltic Sea
Riga
Latvia
North Sea
Edinburgh
Copenhagen
Denmark
Lithuania
United Kingdom
Vilnius
Isle of Man
Minsk
Минск
Manchester
Hamburg
Dublin
Belarus
Ireland
Liverpool
Berlin
Poland
Amsterdam
Warsaw
Netherlands
Brussels
Cologne
Germany
Kyiv
Київ
Belgium
Frankfurt
Prague
Luxembourg
Czechia
Paris
Slovakia
Munich
Vienna
Budapest
Austria
Moldova
Chisinau
Hungary
Switzerland
France
Slovenia
Zagreb
Milan
Romania
Belgrade
Београд
Croatia
Bucharest
Bosnia and
Herzegovina
Serbia
Bay of Biscay
Monaco
Sarajevo
Sofia
София
Italy
Montenegro
Kosovo

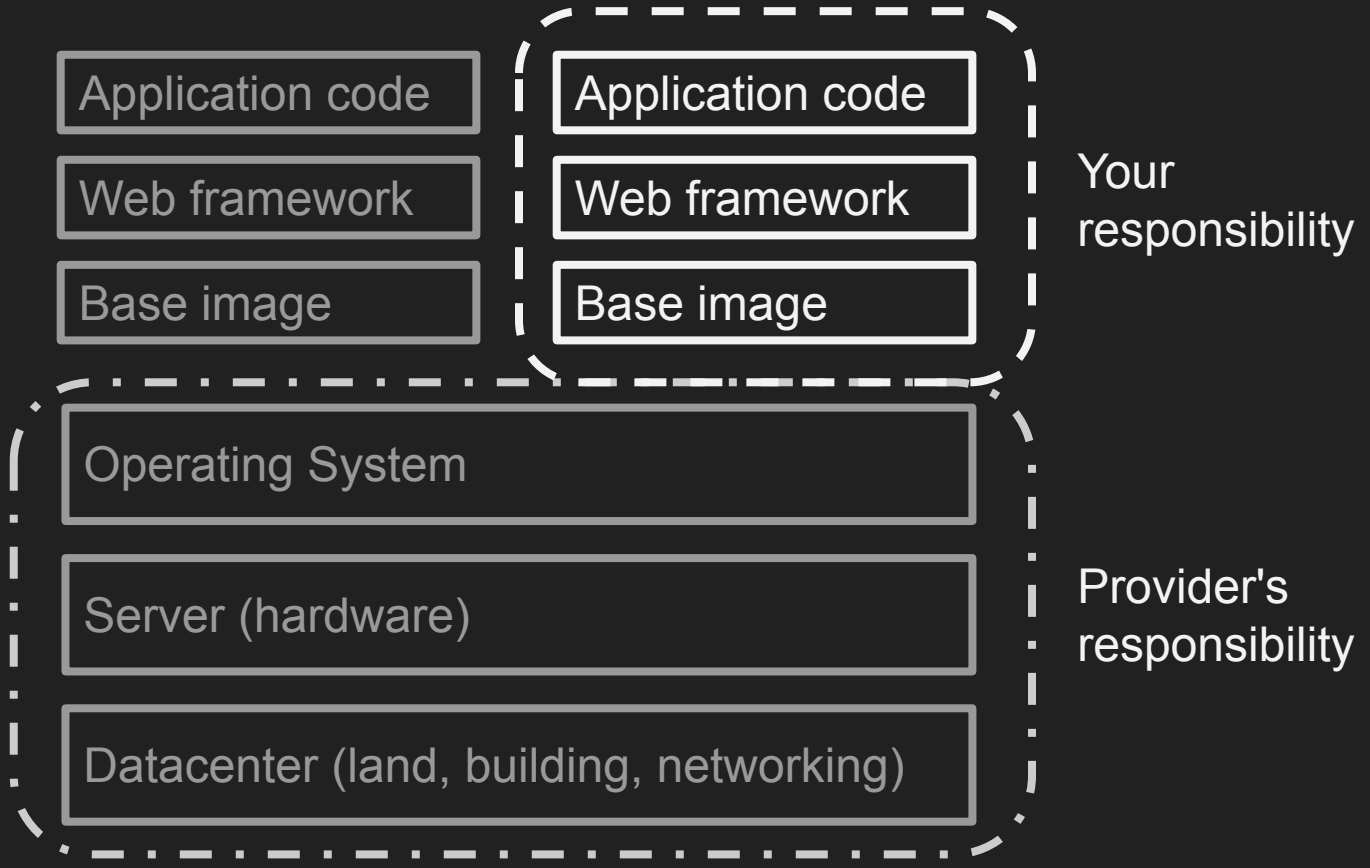
# What is Serverless? - the old days



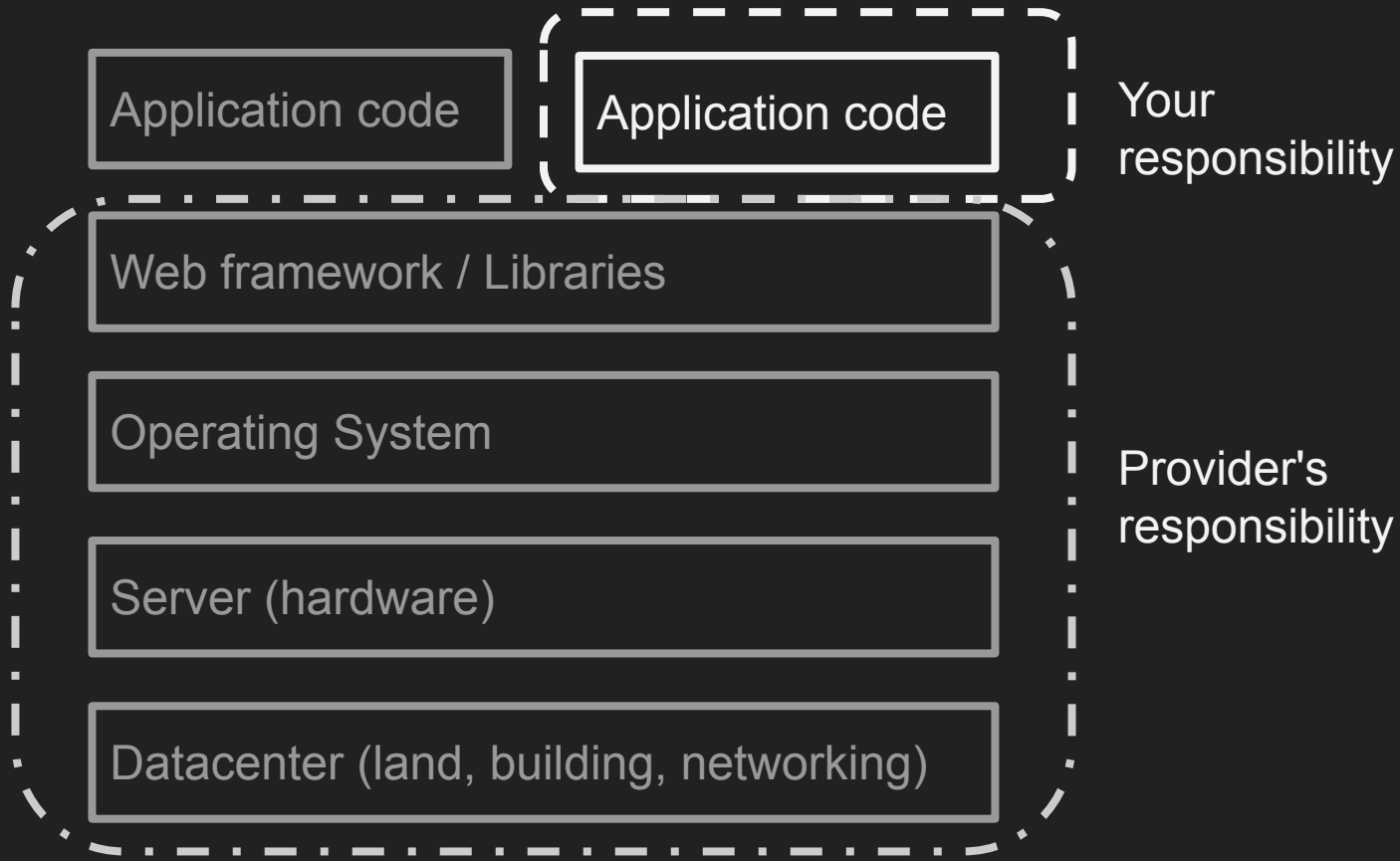
# What is Serverless? - VM / VPS



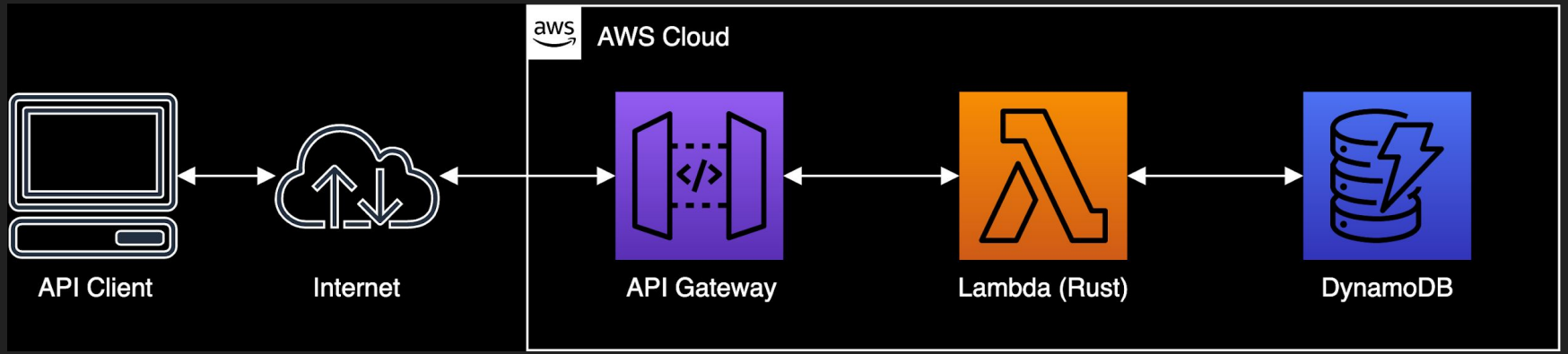
# What is Serverless? - Container



# What is Serverless? - serverless



# Architecture



AWS Open Source Blog

# Rust Runtime for AWS Lambda

by Stefano Buliani | on 29 NOV 2018 | in AWS Lambda, Open Source | Permalink | Comments | Share

中文版

AWS Lambda, which makes it easy for developers to run code for virtually any type of application or backend service with zero administration, has just announced the Runtime APIs. The Runtime APIs define an HTTP-based specification of the Lambda programming model which can be implemented in any programming language. To accompany the API launch, we have open sourced a **runtime for the Rust language**. If you're not familiar with Rust, it's a programming language for

# lambda_runtime 0.2.1

Homepage Documentation Repository Dependent crates

Cargo.toml `lambda_runtime = "0.2.1"`

## Rust Runtime for AWS Lambda

build passing

This package makes it easy to run AWS Lambda Functions written in Rust. This workspace includes multiple crates:

- docs 0.2.2 **`lambda-runtime-client`** is a client SDK for the Lambda Runtime APIs. You probably don't need to use this crate directly!
- docs 0.2.1 **`lambda-runtime`** is a library that makes it easy to write Lambda functions in Rust.
- docs 0.1.1 **`lambda-http`** is a library that makes it easy to write API Gateway proxy event focused Lambda functions in Rust.

### Example function

The code below creates a simple function that receives an event with a `greeting`

Last Updated
**3 months ago**

maintenance actively-developed

build passing

Crate Size
**4.23 kB**

### Authors

- David Barsky
- Stefano Buliani

### License

Apache-2.0

### Keywords

rust runtime lambda aws

# lambda_http 0.1.1

Homepage Documentation Repository Dependent crates

Cargo.toml `lambda_http = "0.1.1"`

## Rust Runtime for AWS Lambda

build passing

This package makes it easy to run AWS Lambda Functions written in Rust. This workspace includes multiple crates:

- docs 0.2.2 **`lambda-runtime-client`** is a client SDK for the Lambda Runtime APIs. You probably don't need to use this crate directly!
- docs 0.2.1 **`lambda-runtime`** is a library that makes it easy to write Lambda functions in Rust.
- docs 0.1.1 **`lambda-http`** is a library that makes it easy to write API Gateway proxy event focused Lambda functions in Rust.

### Example function

The code below creates a simple function that receives an event with a `greeting`

Last Updated
**3 months ago**

maintenance actively-developed

build passing

Crate Size
**14.56 kB**

### Authors

- Doug Tangren

### License

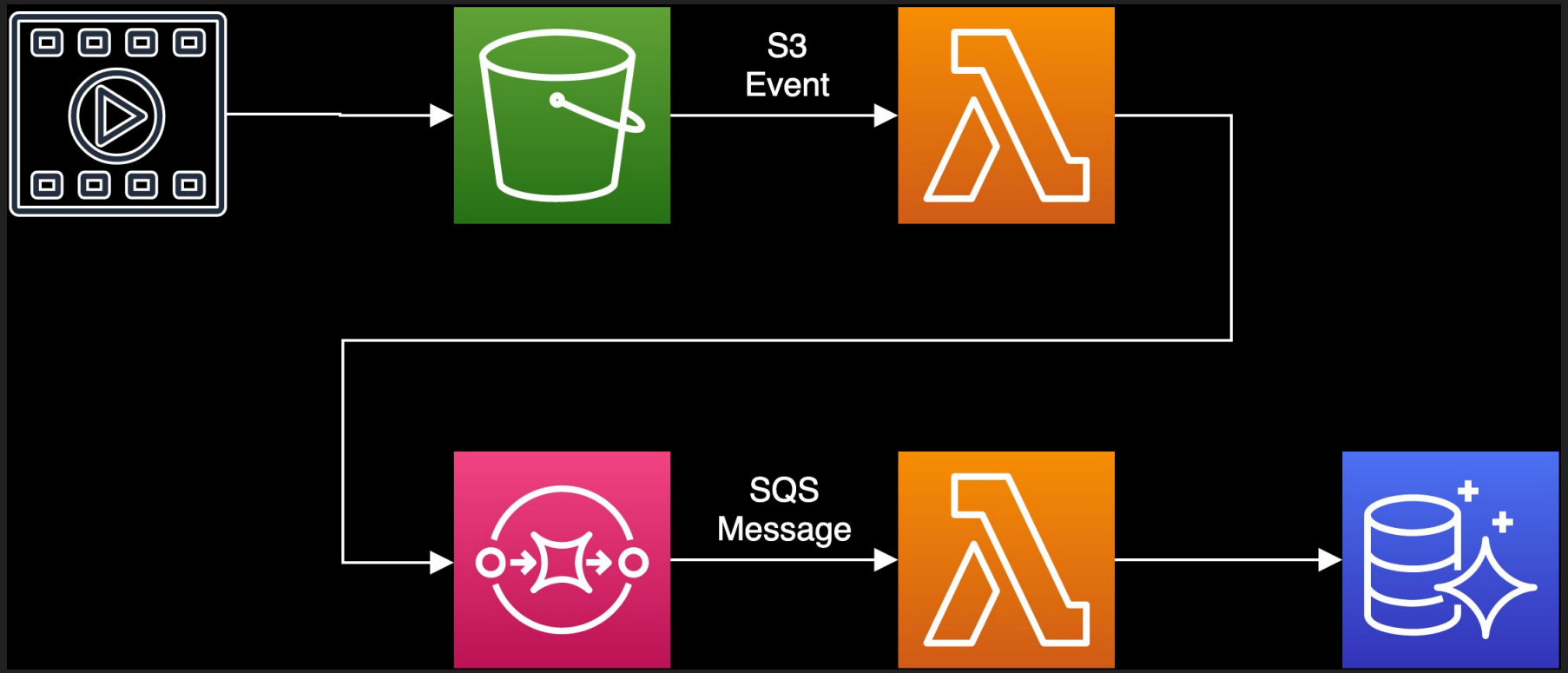
Apache-2.0

### Keywords

api lambda alb aws apigateway

### Owners

# Non-HTTP events



# Hello world

```
use std::error::Error;

use lambda_runtime::{error::HandlerError, lambda, Context};
use log::{self, error};
use serde_derive::{Deserialize, Serialize};
use simple_error::bail;
use simple_logger;


fn main() -> Result<(), Box<dyn Error>> {
    simple_logger::init_with_level(log::Level::Debug)?;

    lambda!(my_handler);

    Ok(())
}
```

# Hello world (2) -handler function

```
fn my_handler(
    e: CustomEvent,
    c: Context
) -> Result<CustomOutput, HandlerError> {
    Ok(CustomOutput {
        message: format!("Hello, {}!", e.first_name),
    })
}
```

# Hello world (3) - event format

```
#[derive(Deserialize)]
struct CustomEvent {
    #[serde(rename = "firstName")]
    first_name: String,
}
#[derive(Serialize)]
struct CustomOutput {
    message: String,
}
```

# Hello world (4) - Cargo.toml

```
[package]
name = "rust-serverless-example"
// ...
[dependencies]
lambda_runtime = "0.2.1"
// …
[[bin]]
name = "bootstrap"
path = "src/main.rs"
```

# Compile for AWS Lambda

- MacOS
  - rustup target add x86_64-unknown-linux-musl
  - brew install filosottile/musl-cross/musl-cross`
  - Configure the linker in ./cargo/config
    - [target.x86_64-unknown-linux-musl]
      linker = "x86_64-linux-musl-gcc"
  - ln -s /usr/local/bin/x86_64-linux-musl-gcc /usr/local/bin/musl-gcc
- cargo build
- ./target/x86_64-unknown-linux-musl/release/bootstrap ⇒ package.zip

# Create function Info

Choose one of the following options to create your function.

**Author from scratch**
Start with a simple Hello World example.

**Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.

**Browse serverless app repository**
Deploy a sample Lambda application from the AWS Serverless Application Repository.

## Basic information

Function name
Enter a name that describes the purpose of your function.

rust-serverless-example

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.

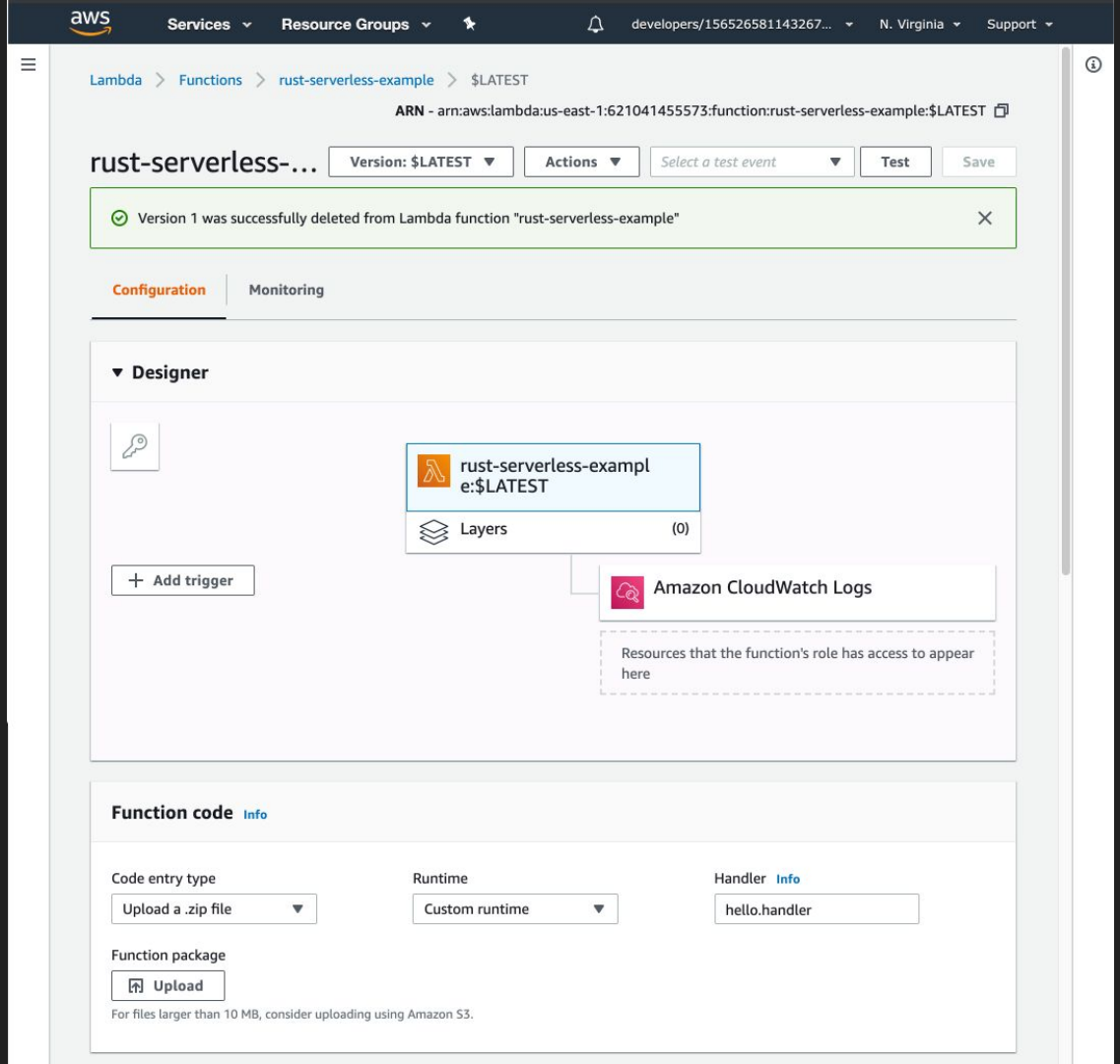
Provide your own bootstrap

## Permissions Info

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▶ Choose or create an execution role

Cancel Create function


aws
Services
Resource Groups
developers/156526581143267...
N. Virginia
Support
Lambda > Functions > rust-serverless-example > $LATEST
ARN - arn:aws:lambda:us-east-1:621041455573:function:rust-serverless-example:$LATEST
rust-serverless-...
Version: $LATEST
Actions
Select a test event
Test
Save
Version 1 was successfully deleted from Lambda function "rust-serverless-example"
Configuration
Monitoring
Designer
rust-serverless-exampl
e:$LATEST
Layers (0)
Add trigger
Amazon CloudWatch Logs
Resources that the function's role has access to appear here
Function code Info
Code entry type
Upload a .zip file
Runtime
Custom runtime
Handler Info
hello.handler
Function package
Upload
For files larger than 10 MB, consider uploading using Amazon S3.

## Function code Info

Code entry type

Upload a .zip file

Runtime

Custom runtime

Handler Info

hello.handler

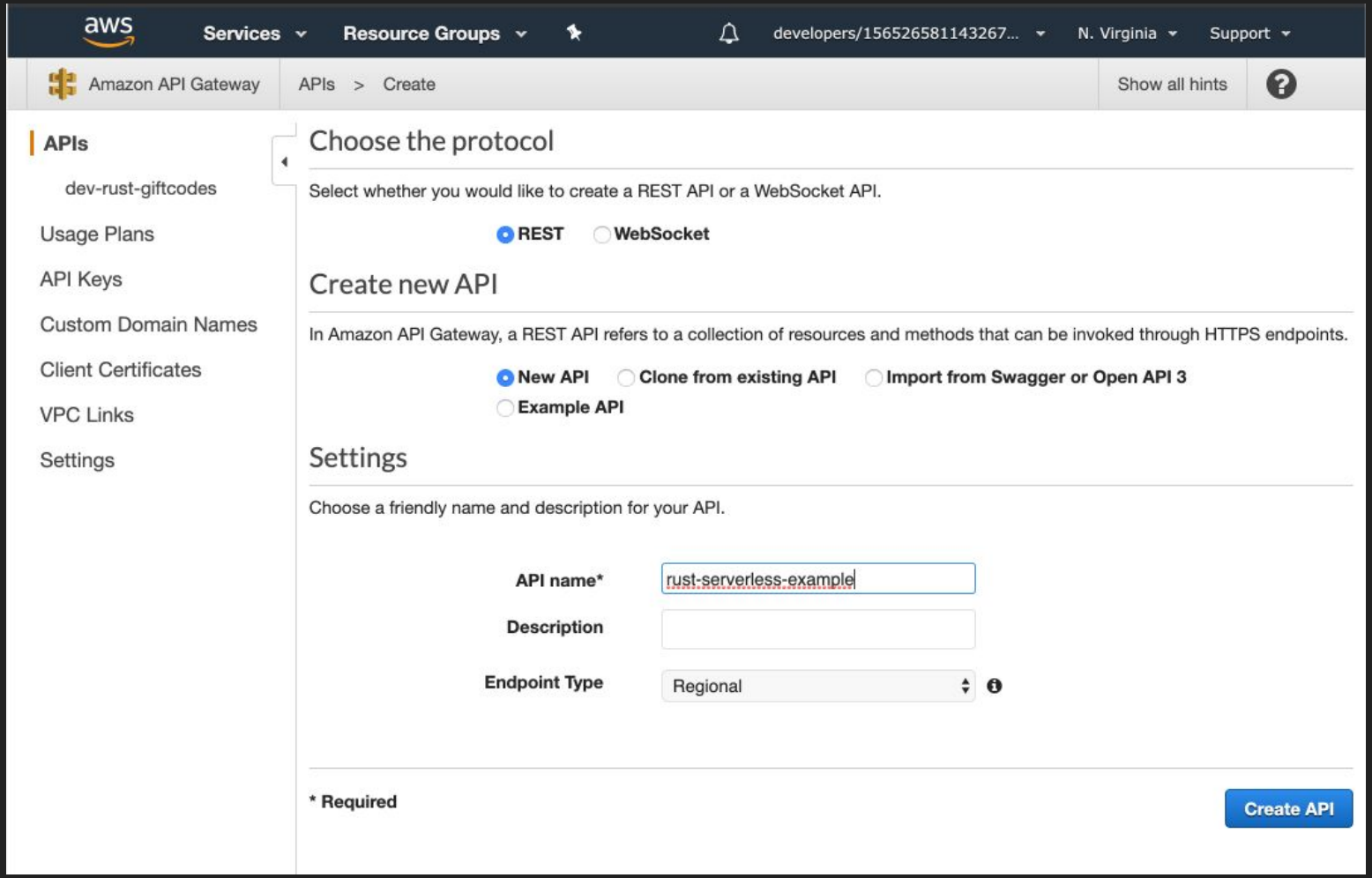
Function package

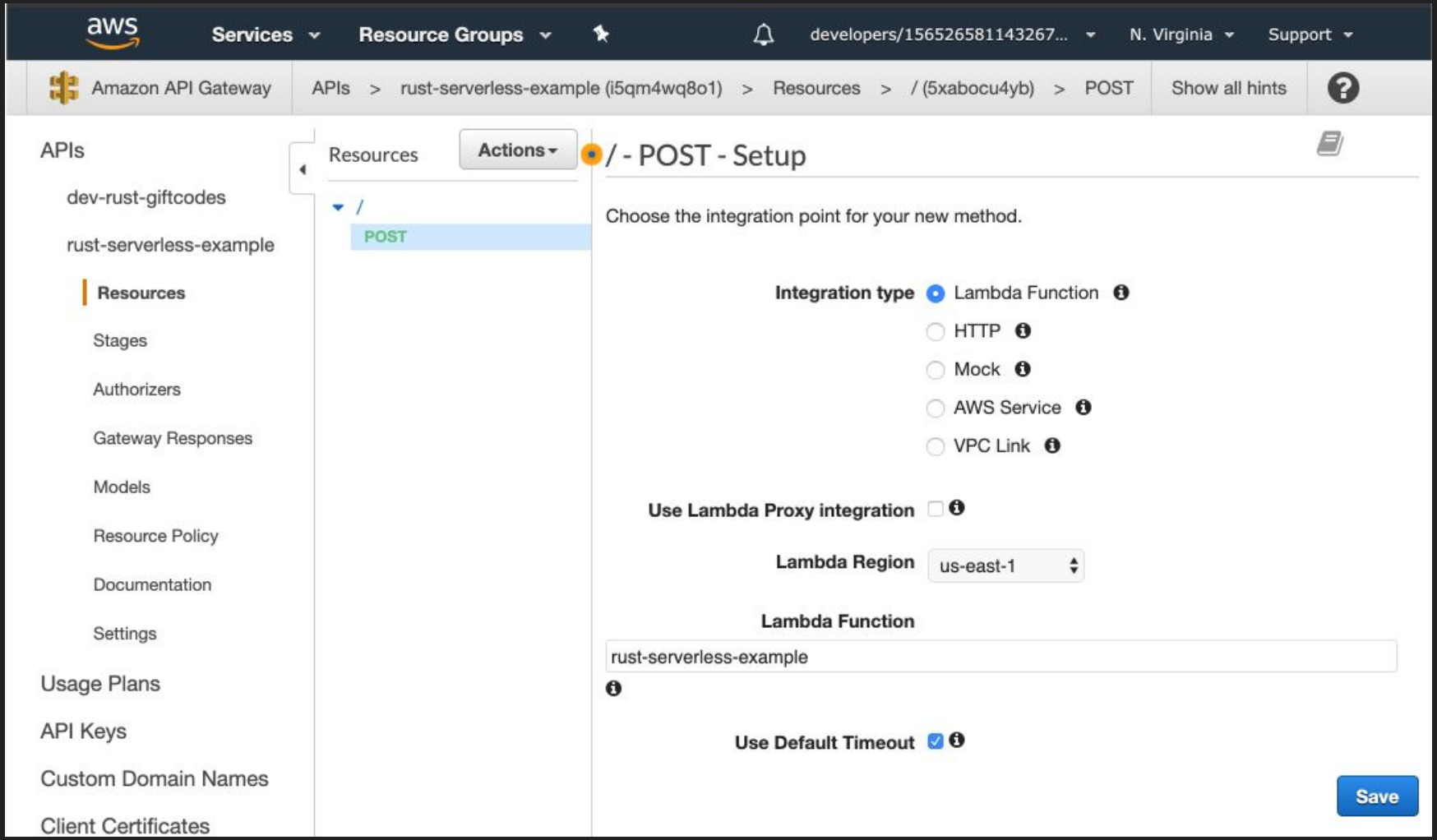
Upload

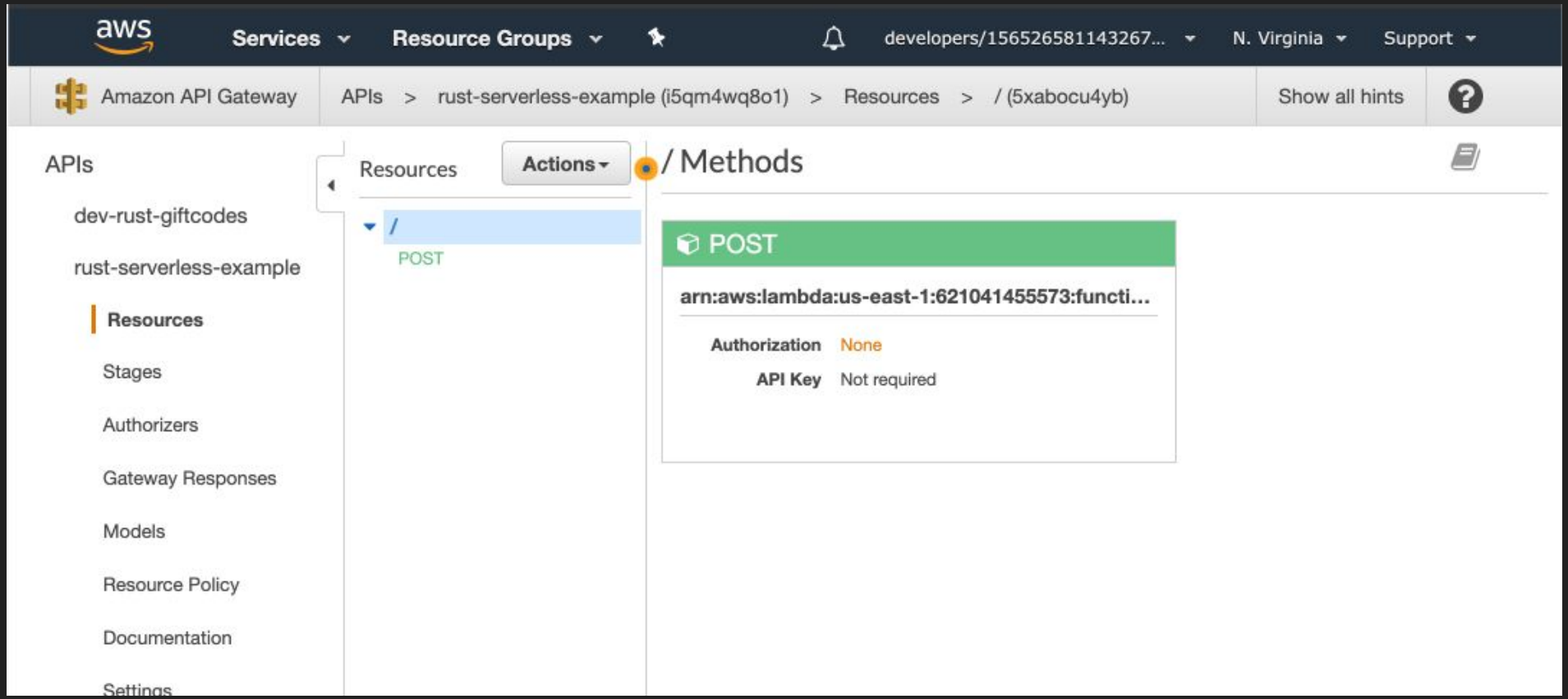
For files larger than 10 MB, consider uploading using Amazon S3.

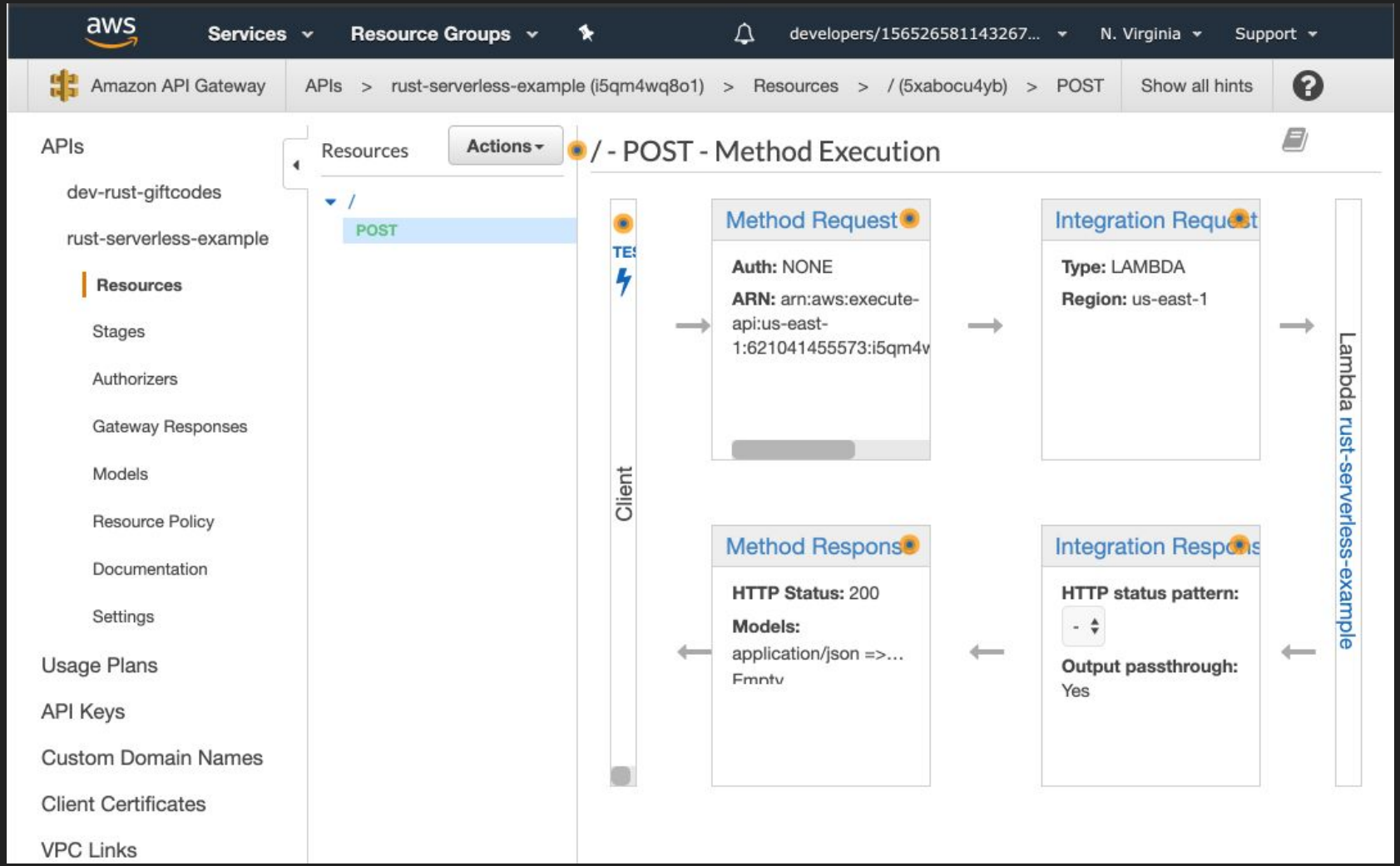
## Environment variables

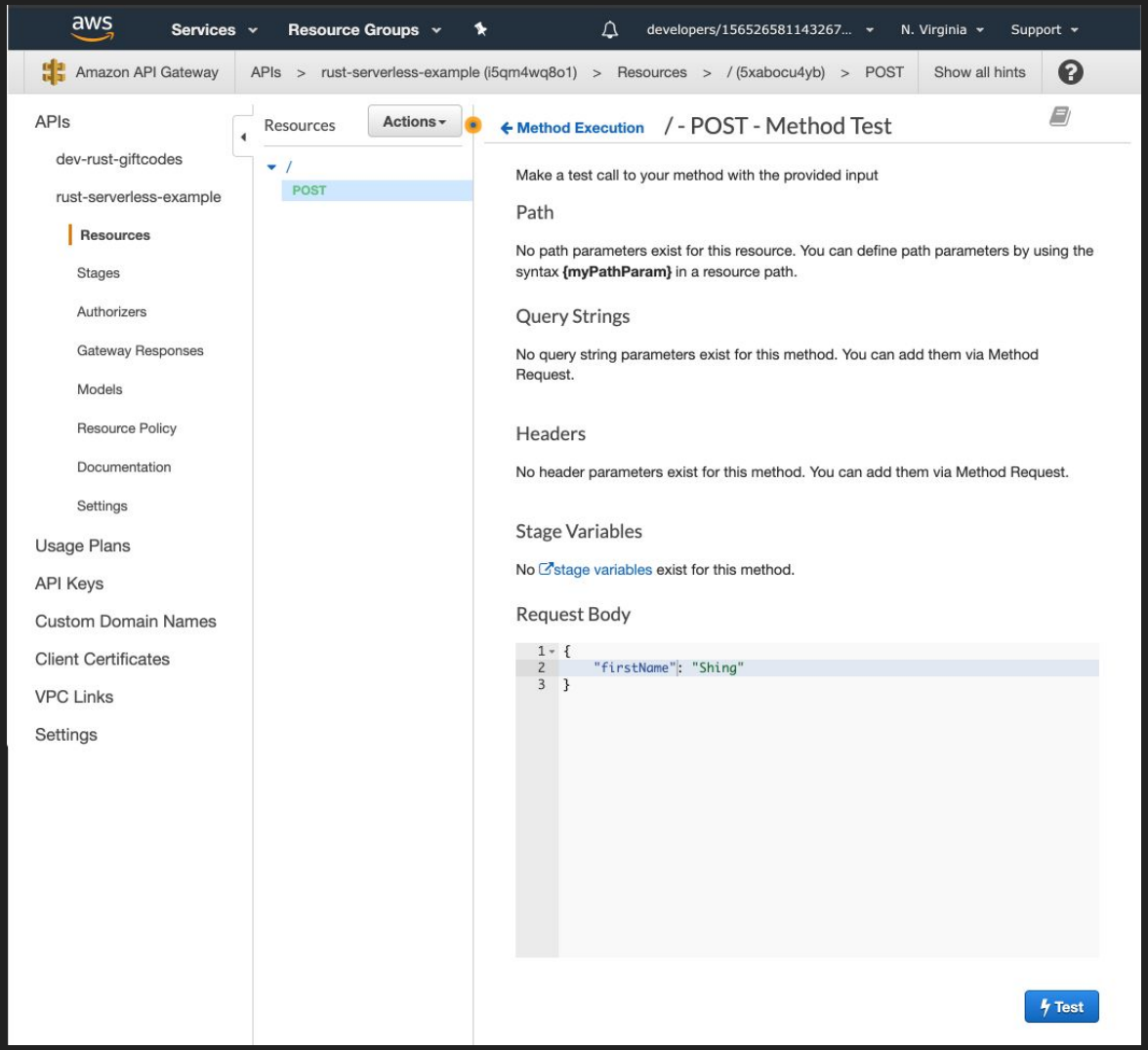
You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more

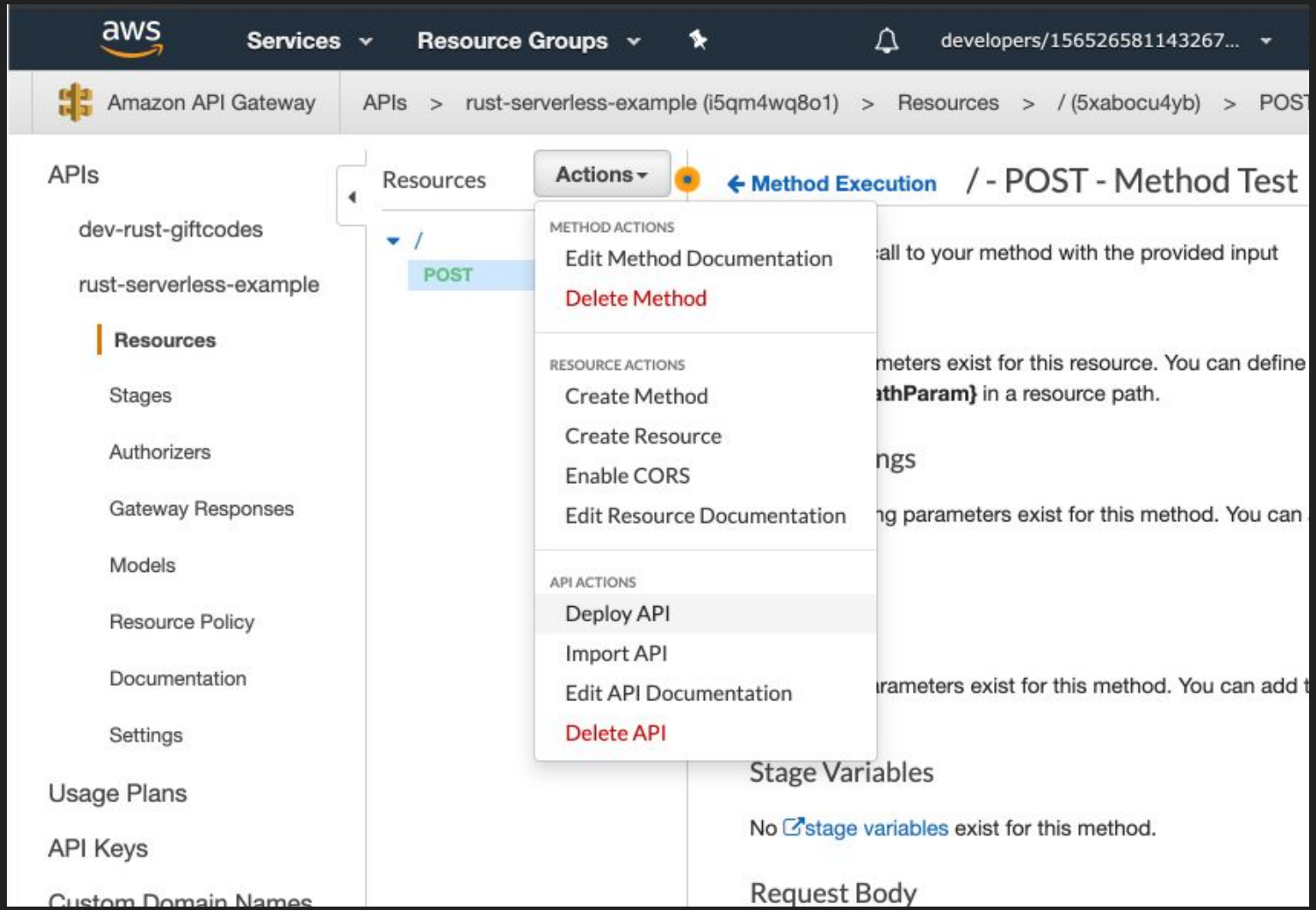

aws
Services
Resource Groups
developers/156526581143267...
N. Virginia
Support
Amazon API Gateway
APIs > Create
Show all hints
APIs
dev-rust-giftcodes
Usage Plans
API Keys
Custom Domain Names
Client Certificates
VPC Links
Settings
Choose the protocol
Select whether you would like to create a REST API or a WebSocket API.
REST
WebSocket
Create new API
In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.
New API
Clone from existing API
Import from Swagger or Open API 3
Example API
Settings
Choose a friendly name and description for your API.
API name*
rust-serverless-example
Description
Endpoint Type
Regional
* Required
Create API

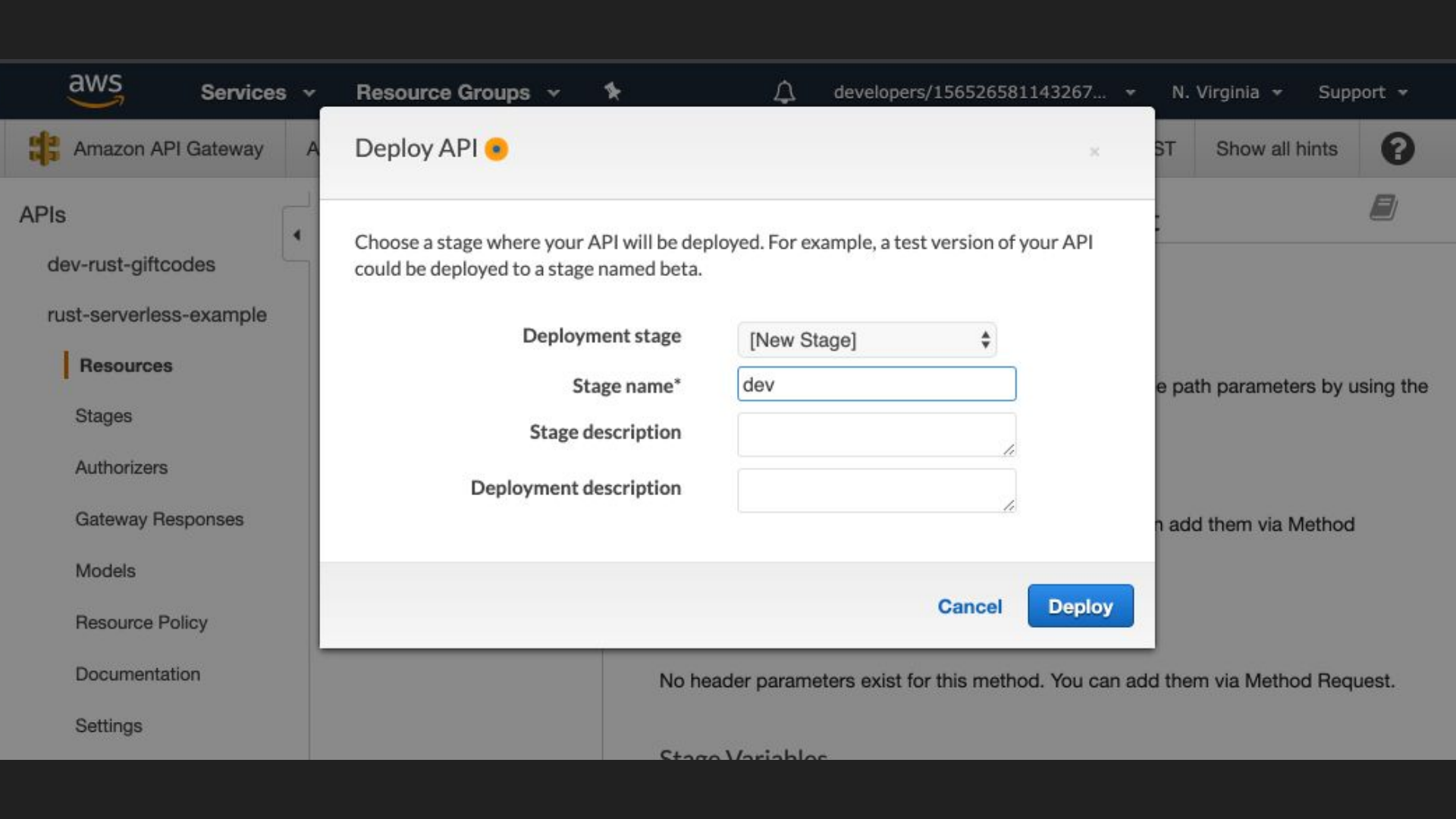

aws
Services
Resource Groups
developers/156526581143267...
N. Virginia
Support
Amazon API Gateway
APIs > rust-serverless-example (i5qm4wq8o1) > Resources > / (5xabocu4yb) > POST
Show all hints
APIs
dev-rust-giftcodes
rust-serverless-example
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Settings
Usage Plans
API Keys
Custom Domain Names
Client Certificates
Resources
Actions
/
POST
/ - POST - Setup
Choose the integration point for your new method.
Integration type
Lambda Function
HTTP
Mock
AWS Service
VPC Link
Use Lambda Proxy integration
Lambda Region
us-east-1
Lambda Function
rust-serverless-example
Use Default Timeout
Save

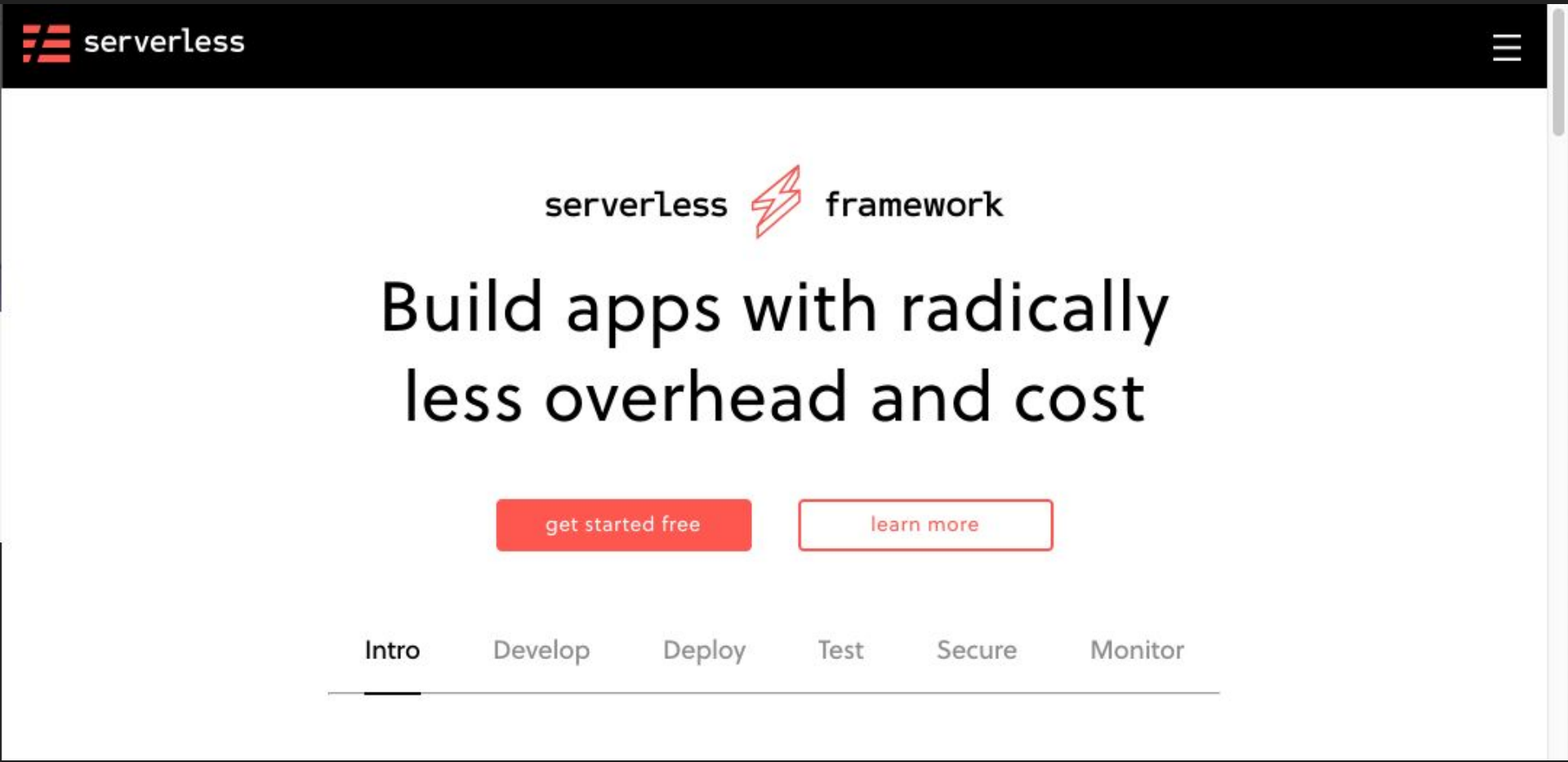

aws
Services
Resource Groups
developers/156526581143267...
N. Virginia
Support
Amazon API Gateway
APIs > rust-serverless-example (i5qm4wq8o1) > Resources > / (5xabocu4yb)
Show all hints
APIs
dev-rust-giftcodes
rust-serverless-example
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Settings
Resources
Actions
/
POST
/ Methods
POST
arn:aws:lambda:us-east-1:621041455573:functi...
Authorization None
API Key Not required


aws
Services
Resource Groups
developers/156526581143267...
N. Virginia
Support
Amazon API Gateway
APIs > rust-serverless-example (i5qm4wq8o1) > Resources > / (5xabocu4yb) > POST
Show all hints
APIs
dev-rust-giftcodes
rust-serverless-example
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Settings
Usage Plans
API Keys
Custom Domain Names
Client Certificates
VPC Links
Resources
Actions
/
POST
/ - POST - Method Execution
Client
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:621041455573:i5qm4v
Integration Request
Type: LAMBDA
Region: us-east-1
Lambda rust-serverless-example
Method Response
HTTP Status: 200
Models:
application/json =>...
Empty
Integration Response
HTTP status pattern:
-
Output passthrough:
Yes


aws
Services
Resource Groups
developers/156526581143267...
N. Virginia
Support
Amazon API Gateway
APIs > rust-serverless-example (i5qm4wq8o1) > Resources > / (5xabocu4yb) > POST
Show all hints
APIs
dev-rust-giftcodes
rust-serverless-example
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Settings
Usage Plans
API Keys
Custom Domain Names
Client Certificates
VPC Links
Settings
Resources
Actions
/
POST
Method Execution
/ - POST - Method Test
Make a test call to your method with the provided input
Path
No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.
Query Strings
No query string parameters exist for this method. You can add them via Method Request.
Headers
No header parameters exist for this method. You can add them via Method Request.
Stage Variables
No stage variables exist for this method.
Request Body
1 {
2 "firstName": "Shing"
3 }
Test


aws
Services
Resource Groups
developers/156526581143267...
Amazon API Gateway
APIs > rust-serverless-example (i5qm4wq8o1) > Resources > / (5xabocu4yb) > POST
APIs
dev-rust-giftcodes
rust-serverless-example
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Settings
Usage Plans
API Keys
Custom Domain Names
Resources
Actions
Method Execution
/ - POST - Method Test
/
POST
METHOD ACTIONS
Edit Method Documentation
Delete Method
RESOURCE ACTIONS
Create Method
Create Resource
Enable CORS
Edit Resource Documentation
API ACTIONS
Deploy API
Import API
Edit API Documentation
Delete API
all to your method with the provided input
meters exist for this resource. You can define
athParam} in a resource path.
ngs
g parameters exist for this method. You can
rameters exist for this method. You can add t
Stage Variables
No stage variables exist for this method.
Request Body


aws
Services
Resource Groups
developers/156526581143267...
N. Virginia
Support
Amazon API Gateway
Show all hints
APIs
dev-rust-giftcodes
rust-serverless-example
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Settings
Deploy API
Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.
Deployment stage
[New Stage]
Stage name*
dev
Stage description
Deployment description
Cancel
Deploy
e path parameters by using the
n add them via Method
No header parameters exist for this method. You can add them via Method Request.


serverless
serverless framework
Build apps with radically less overhead and cost
get started free
learn more
Intro
Develop
Deploy
Test
Secure
Monitor

# The template

- https://github.com/softprops/serverless-aws-rust-http
- npm install -g serverless
- npx serverless install \

  --url https://github.com/softprops/serverless-aws-rust-http \

  --name my-new-api

# serverless.yml

```yaml
service: rust-serverless-example
provider:
  name: aws
  runtime: rust
  memorySize: 128

package:
  individually: true

plugins:
  - serverless-rust
```

```yaml
functions:
  hello:
    # handler value syntax is
    #`{cargo-package-name}.{bin-name}`
    # or `{cargo-package-name}`
    handler: hello
    events:
      - http:
          path: '/'
          method: GET
```

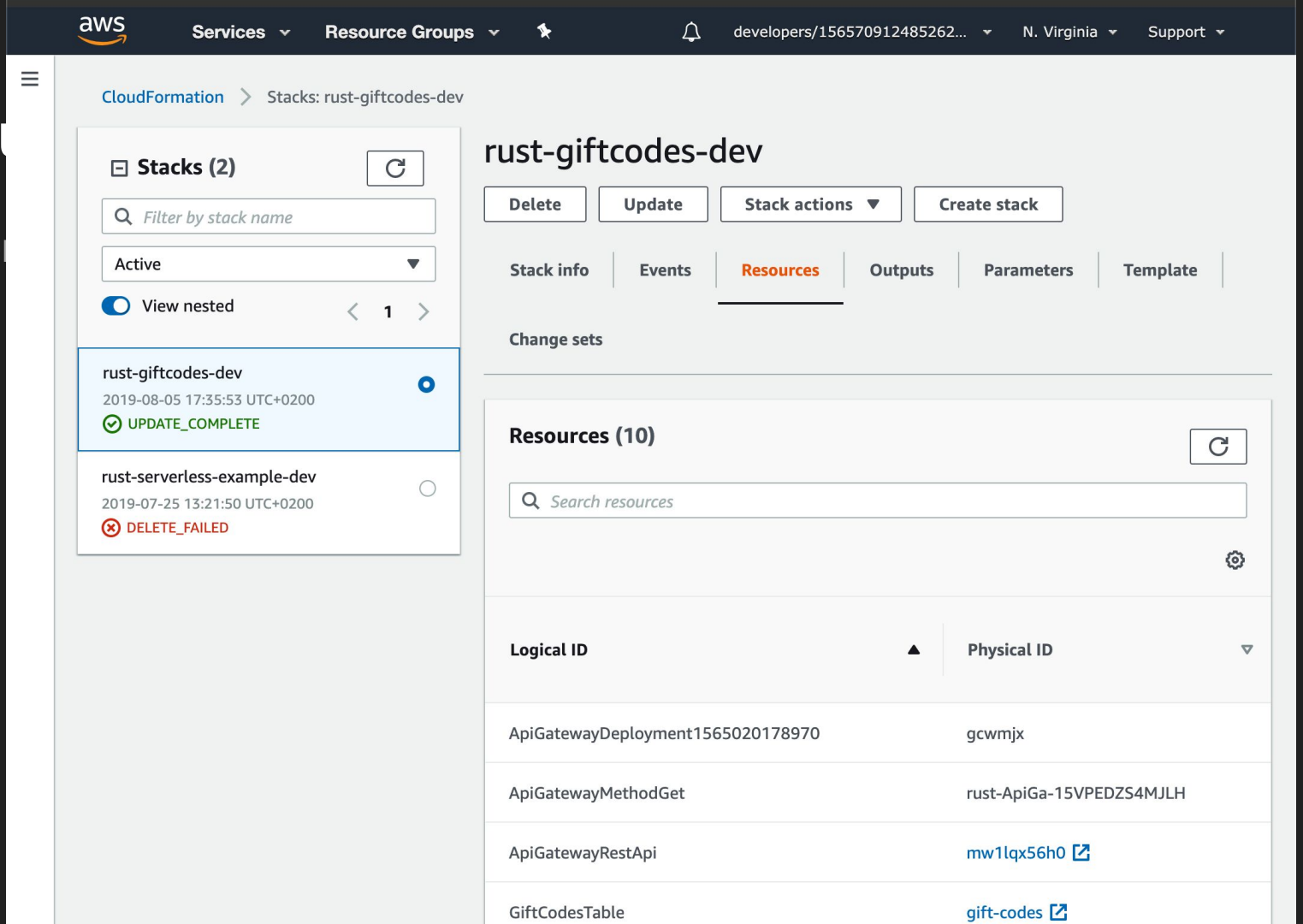
# serverless deployment commands

- Set up your AWS credential: https://serverless.com/framework/docs/providers/aws/guide/credentials/

- npm install
- npx serverless deploy

Clou

scree


aws
Services
Resource Groups
developers/156570912485262...
N. Virginia
Support
CloudFormation
Stacks: rust-giftcodes-dev
Stacks (2)
Filter by stack name
Active
View nested
1
rust-giftcodes-dev
2019-08-05 17:35:53 UTC+0200
UPDATE_COMPLETE
rust-serverless-example-dev
2019-07-25 13:21:50 UTC+0200
DELETE_FAILED
rust-giftcodes-dev
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Resources (10)
Search resources
Logical ID
Physical ID
ApiGatewayDeployment1565020178970
gcwmjx
ApiGatewayMethodGet
rust-ApiGa-15VPEDZS4MJLH
ApiGatewayRestApi
mw1lqx56h0
GiftCodesTable
gift-codes

# Hello world with lambda_http

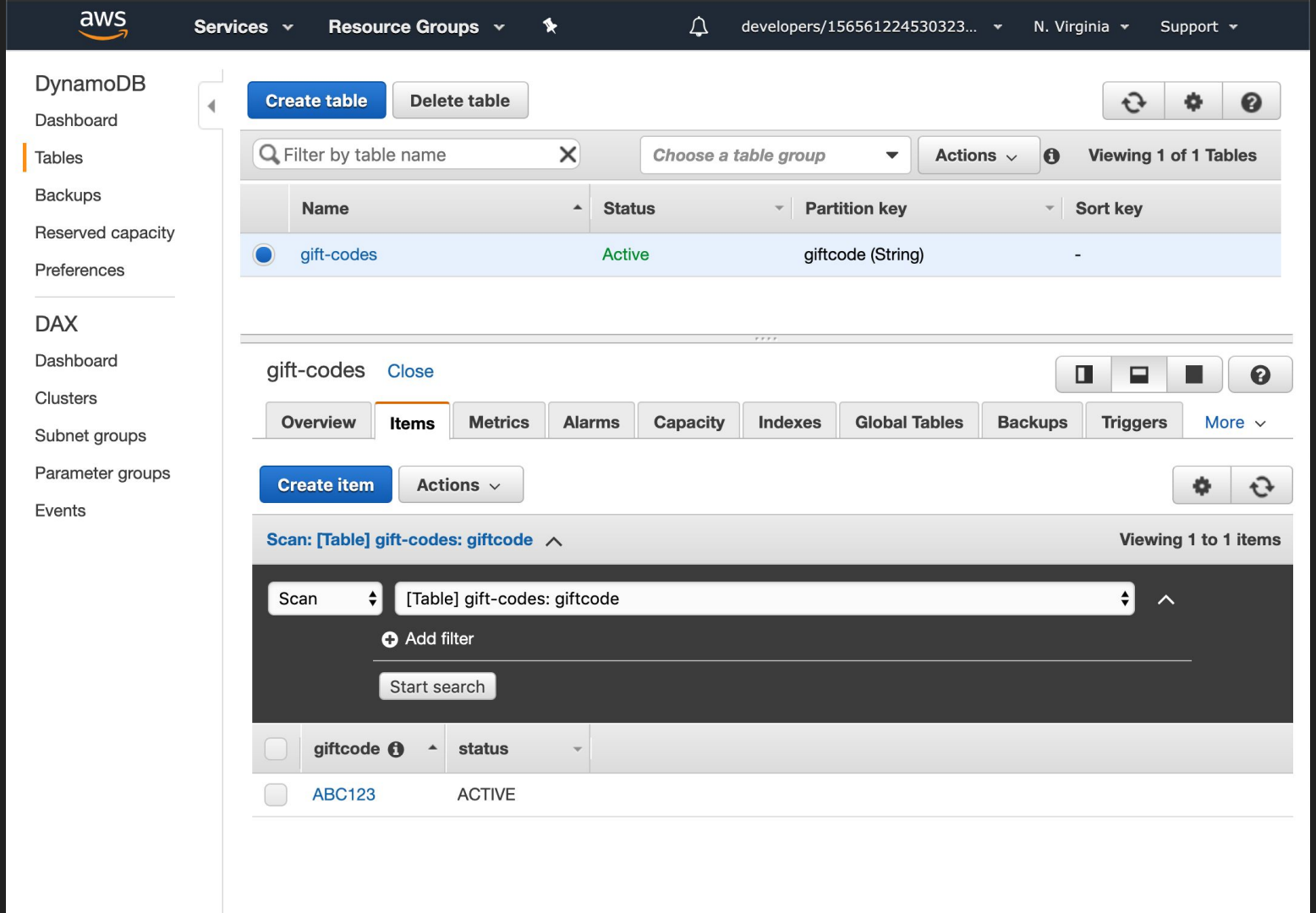
```rust
use lambda_http::{lambda, IntoResponse, Request};
use lambda_runtime::{error::HandlerError, Context};
use serde_json::json;

fn main() {
    lambda!(handler)
}

fn handler(
    _: Request,
    _: Context,
) -> Result<impl IntoResponse, HandlerError> {
    // creating an application/json response
    Ok(json!({  // `serde_json::Values` impl `IntoResponse` by default
        "message": "Go Serverless v1.0! Your function executed successfully!"
    }))
}
```

# Creating a Database

```
# serverless.yml
resources:
  Resources:
    NewResource:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: gift-codes
        AttributeDefinitions:
          - AttributeName: giftcode
            AttributeType: S
        KeySchema:
          - AttributeName: giftcode
            KeyType: HASH
        ProvisionedThroughput:
          ReadCapacityUnits: 1
          WriteCapacityUnits: 1
```


aws
Services
Resource Groups
developers/156561224530323...
N. Virginia
Support
DynamoDB
Dashboard
Tables
Backups
Reserved capacity
Preferences
DAX
Dashboard
Clusters
Subnet groups
Parameter groups
Events
Create table
Delete table
Filter by table name
Choose a table group
Actions
Viewing 1 of 1 Tables
Name
Status
Partition key
Sort key
gift-codes
Active
giftcode (String)
-
gift-codes
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Global Tables
Backups
Triggers
More
Create item
Actions
Scan: [Table] gift-codes: giftcode
Viewing 1 to 1 items
Scan
[Table] gift-codes: giftcode
Add filter
Start search
giftcode
status
ABC123
ACTIVE

# IAM permissions

```
# serverless.yml
provider:
  name: aws

  iamRoleStatements:
      - Effect: "Allow"
        Action:
            - "dynamodb:GetItem"
        Resource:
            - "*" # DANGER!
```

- with S

README.md

# Rusoto

| | |
|---|---|
| Linux / OS X | Azure Pipelines succeeded |
| Windows | build passing |
| Ceph and Minio support | BUILD STATUS PASSED |

API docs | crates.io v0.40.0 | license MIT | Total lines 949.5K

**Rusoto is an AWS SDK for Rust**

---

You may be looking for:

- An overview of Rusoto
- AWS services supported by Rusoto
- API documentation
- Getting help with Rusoto

## Installation

# Reading from Database

```
use rusoto_core::{Region};
use rusoto_dynamodb::{AttributeValue, DynamoDb, DynamoDbClient, GetItemInput};

let client = DynamoDbClient::new(Region::UsEast1);

// prepare the query: get_item_input

match client.get_item(get_item_input).sync() {
    Ok(output) => { // return 200 OK response }
    Err(error) => { // return 500 Internal Server Error response }
}
```

# Reading the query parameters

```
fn handler(req: Request, _: Context) -> Result<impl IntoResponse, HandlerError> {

    let response = match req.query_string_parameters().get("giftcode") {
        Some(giftcode) => {
            // Search the giftcode in DynamoDB and return the response
        }
        None => { // Return 400 Bad Request }
    };
    Ok(response)
}
```

# Preparing the Query

```
let mut key: HashMap<String, AttributeValue> = HashMap::new();
key.insert(
    "giftcode".to_string(),
    AttributeValue {
        s: Some(giftcode.to_string()),
        ..Default::default()
    }
);

let get_item_input: GetItemInput = GetItemInput {
    table_name: "gift-codes".to_string(),
    key: key,
    ..Default::default()
};
```

# 200 OK Response

```
match client.get_item(get_item_input).sync() {
    Ok(output) => {
        match output.item {
            Some(item) => {
                json!({
                    "giftcode": item.get("giftcode").unwrap().s,
                    "status": item.get("status").unwrap().s,
                }).into_response()
            },
            None => { // 404 Not Found }
        }
    }
    Err(error) => { // 500 Internal Server Error }
}
```

```
{
    "giftcode": "ABC123"
    "status": "ACTIVE"
}
```

# 404 Not found Response

```
match client.get_item(get_item_input).sync() {
    Ok(output) => {
        match output.item {
            Some(item) => { // 200 OK }
            None => {
                Response::builder()
                    .status(StatusCode::NOT_FOUND)
                    .body("Gift code not found".into())
                    .expect("Failed to render response")
            }
        }
    }
    Err(error) => { // 500 Internal Server Error }
}
```

# 500 Internal Server Error Response

```
match client.get_item(get_item_input).sync() {
    Ok(output) => {
        match output.item {
            Some(item) => { // 200 OK }
            None => { // 404 Not Found }
        }
    }
    Err(error) => {
        Response::builder()
            .status(StatusCode::INTERNAL_SERVER_ERROR)
            .body(format!("{:?}", error).into())
            .expect("Failed to render response")
    }
}
```

# Testing with curl

```
% curl -X GET -v 'https://mw1lqx56h0.execute-api.us-east-1.amazonaws.com/dev/?giftcode=ABC123'

< HTTP/2 200
< content-type: application/json
< content-length: 39
< date: Mon, 12 Aug 2019 13:34:56 GMT

{"giftcode":"ABC123","status":"ACTIVE"}
```

# Testing with curl

```
% curl -X GET -v 'https://mw1lqx56h0.execute-api.us-east-1.amazonaws.com/dev/?giftcode=NOTEXIST'

< HTTP/2 404
< content-type: application/json
< content-length: 19
< date: Mon, 12 Aug 2019 13:38:43 GMT

Gift code not found
```

# Join DAZN!

We are hiring!

**Relocation to Amsterdam!**


Ever wanted to live and work in one of the worlds fastest growing tech hubs?

DAZN is looking to bring people to our newest development centre in the Netherlands to build the world's largest sports streaming platform. DAZN is currently live in Germany, USA, Japan, Switzerland, Canada, Austria, Spain, Brazil and Italy with millions of concurrent users!

We are hosting a Friday Meetup and 90-minute weekend interview slots for Backend Engineers to join DAZN. Sponsorship and relocation assistance to Europe is on offer!

**Where:** Taipei, Taiwan

**When:** Meetup - Friday 20th September / Interviews - Saturday 21st and Sunday 22nd September.

**You'll need:** To pass a short pre-assessment

**A taste of our Tech Stack:** JavaScript, Node, React, AWS, MobX, Docker, Microfrontend Architecture, Serverless

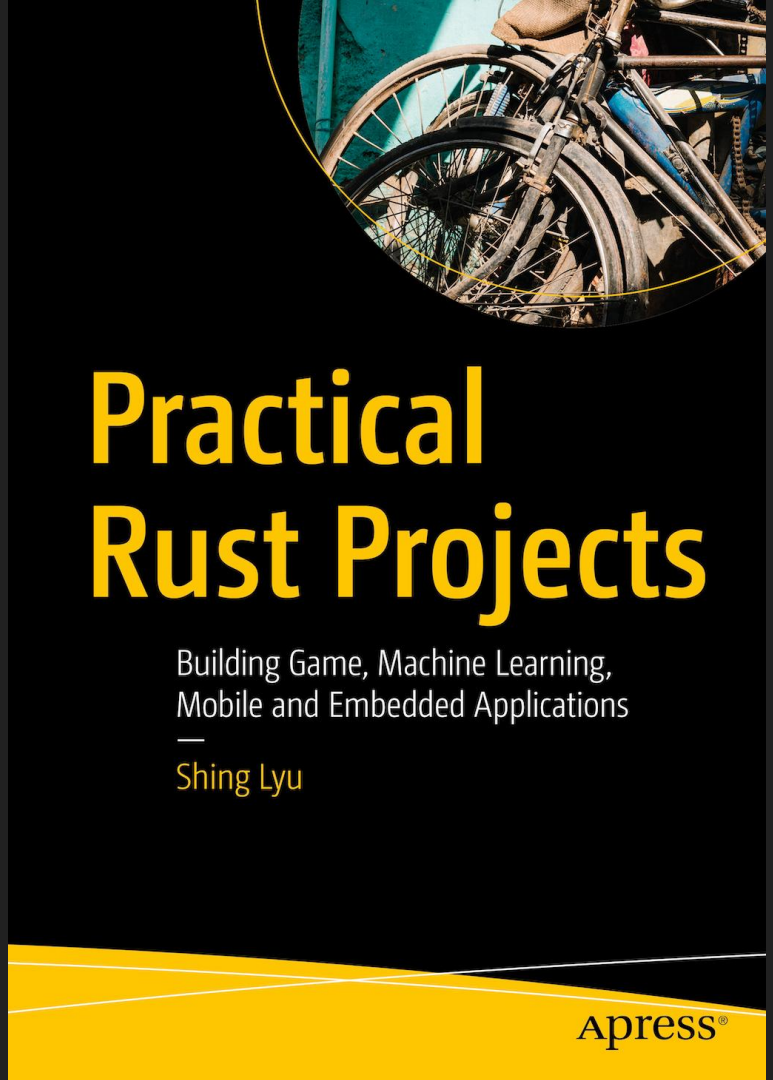
**On offer:** Technical presentations from our engineers, food / drink, SWAG and the chance to be moved across the world to Amsterdam

Please find out more at engineering.dazn.com/hack-taiwan
Email us: taiwantoamsterdam@dazn.com

# Get it now!

- Available on
  - Udemy
  - Safari Books
  - Packt

Coming soon!
(Early 2020)



# Thank you

# Backup

# DAZN HACKS 臺灣

## GET SPONSORSHIP TO MOVE TO AMSTERDAM

Ever wanted to live and work in one of the worlds fastest growing tech hubs?

DAZN is looking to bring people to our newest development centre in the Netherlands to build the world's largest sports streaming platform. DAZN is currently live in Germany, USA, Japan, Switzerland, Canada, Austria, Spain, Brazil and Italy with millions of concurrent users!

We are hosting a Friday Meetup and 90-minute weekend interview slots for Backend Engineers to join DAZN. Sponsorship and relocation assistance to Europe is on offer!

**Where:** Taipei, Taiwan

**When:** Meetup - Friday 20th September / Interviews - Saturday 21st and Sunday 22nd September.

**You'll need:** To pass a short pre-assessment

**A taste of our Tech Stack:** JavaScript, Node, React, AWS, MobX, Docker, Microfrontend Architecture, Serverless

**On offer:** Technical presentations from our engineers, food / drink, SWAG and the chance to be moved across the world to Amsterdam

Please find out more at engineering.dazn.com/hack-taiwan
Email us: taiwantoamsterdam@dazn.com