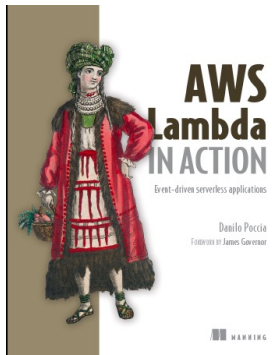


# Function as a Service (FaaS) aka Serverless Computing



AWS Lambda in Action  
Danilo Poccia  
Manning

# Function as a Service (FaaS)

---

## ■ Cloud provider

- Manages infrastructure
- Manages software stack (OS, runtime)
- Handles provisioning
- Availability
- Scalability

## ■ Developer

- Implements application as set of functions
- Functions run when certain events are triggered
  - Web request
  - File upload
  - Alarm
  - Database update

## ■ Examples:

- AWS Lambda, Google Cloud Functions, IBM OpenWhisk, MSFT Azure Functions

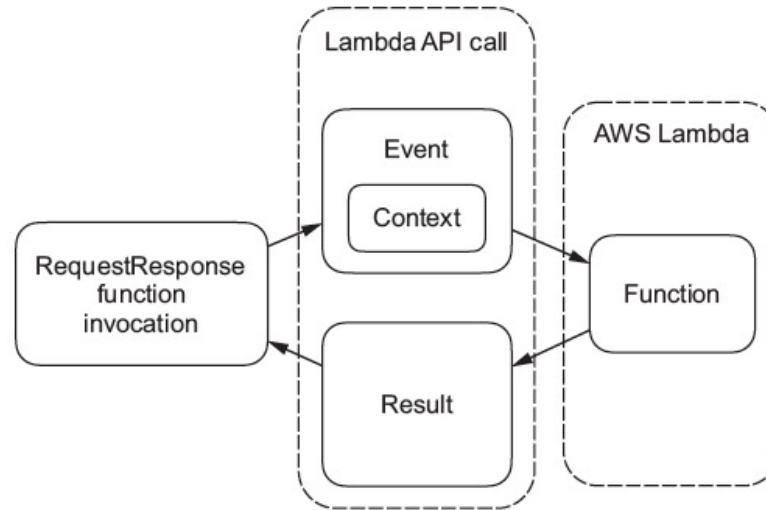
# Lambda Execution Environment

---

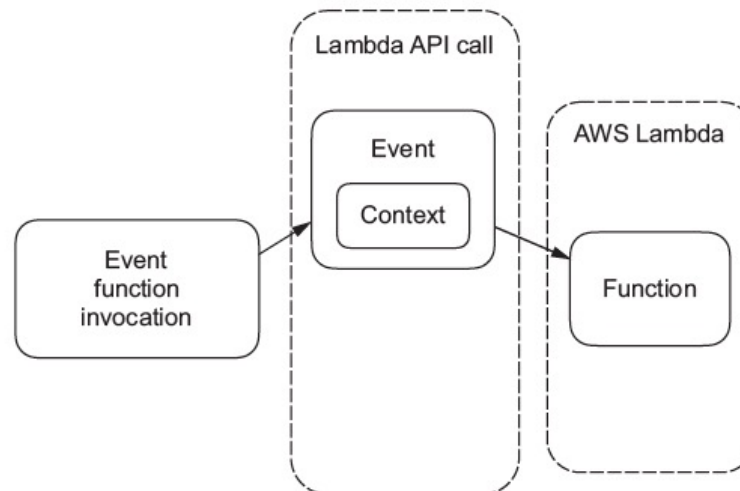
- **Function runs inside an operating system container**
  - Single OS runs multiple containers
  - Isolation ensures that only processes inside container are visible
  - Has its own file system /tmp
    - Runtimes: C#, Node.js, Java, Python
- **A container handles a single function/event at a time**
- **Concurrent execution by container replication**
- **Containers may be reused for subsequent function executions**
- **Containers may be terminated at any time**
  - Stateless
  - Store all persistent state outside of container

# Lambda Invocation Modalities

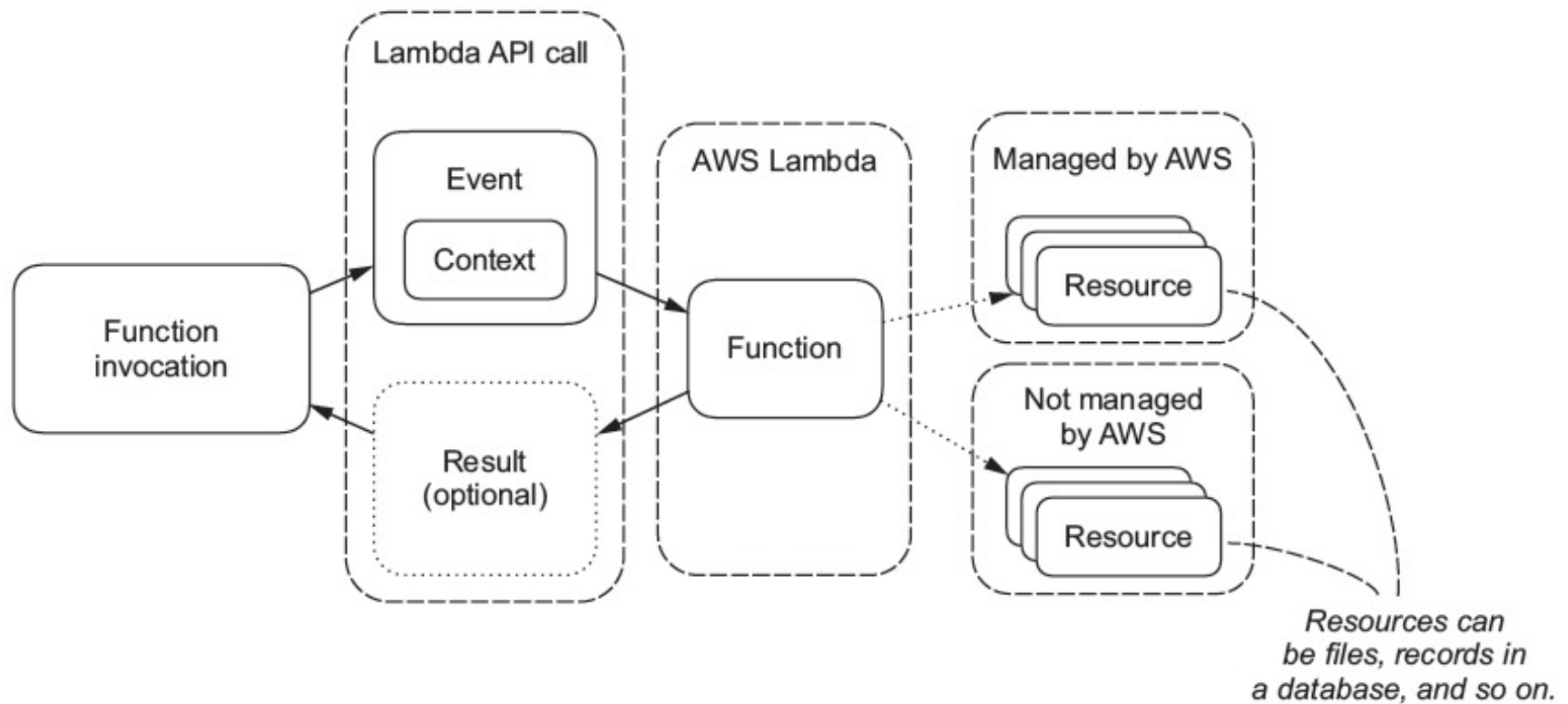
## ■ Synchronous (RequestResponse)



## ■ Asynchronous (Event)



# Lambda Application Architecture



# 1<sup>st</sup> Lambda Function

The screenshot shows the AWS Lambda Management Console in a web browser. The browser's address bar displays the URL `https://console.aws.amazon.com/lambda/home?region=us-`. The console's navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information for 'Eyal de Lara' in the 'N. Virginia' region. The main content area is titled 'COMPUTE' and features a large heading 'AWS Lambda lets you run code without thinking about servers.' Below this, a subheading explains the pay-per-use model: 'You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.'

To the right of the main heading is a 'Get started' section with the text 'Author a Lambda function from scratch, or choose from one of many preconfigured examples.' and a prominent orange button labeled 'Create a function'.

Below the main heading is a 'How it works' section. It includes a 'Run' button and a link 'Next: Lambda responds to events'. A code editor displays the following JavaScript code:

```
1 exports.handler = (event, context, callback) => {  
2   // Succeed with the string "Hello world!"  
3   callback(null, 'Hello world!');  
4 };
```

The footer of the console contains a 'Feedback' link, 'English (US)' language selection, copyright information '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.', and links to 'Privacy Policy' and 'Terms of Use'.

# 1<sup>st</sup> Lambda Function

The screenshot shows the AWS Lambda console interface for creating a new function. The browser address bar shows the URL `console.aws.amazon.com/lambda/home?region=us-east-1#/create/function`. The page title is "Create function" with an "Info" link. Below the title, a message says "Choose one of the following options to create your function." There are three main options, each with a radio button:

- Author from scratch** (selected): "Start with a simple Hello World example." It includes an icon of a code editor and a gear.
- Use a blueprint**: "Build a Lambda application from sample code and configuration presets for common use cases." It includes an icon of a document with a checkmark.
- Browse serverless app repository**: "Deploy a sample Lambda application from the AWS Serverless Application Repository." It includes an icon of a cloud and a document.

Below these options is a section titled "Basic information" containing two form fields:

- Function name**: "Enter a name that describes the purpose of your function." The input field contains the text "helloWorld". Below the field, a note says "Use only letters, numbers, hyphens, or underscores with no spaces."
- Runtime** (with an "Info" link): "Choose the language to use to write your function." The dropdown menu is set to "Python 3.8".

The footer of the console includes links for "Feedback", "English (US)", copyright information "© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.", and links to "Privacy Policy" and "Terms of Use".

# 1<sup>st</sup> Lambda Function

Lambda

console.aws.amazon.com/lambda/home?region=us-east-1#/create/function

aws Services

Eyal de Lara N. Virginia Support

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

helloWorld

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

**Runtime** [Info](#)  
Choose the language to use to write your function.

Python 3.8

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **Change default execution role**

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Create a new role with basic Lambda permissions

☐ Use an existing role

☐ Create a new role from AWS policy templates

[i](#) Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named helloWorld-role-zsteinrm, with permission to upload logs to Amazon CloudWatch Logs.

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



# 1<sup>st</sup> Lambda Function

The screenshot displays the AWS Lambda console interface. At the top, the AWS logo and navigation menu are visible, including 'Services', 'Resource Groups', and user information. A green notification bar at the top states: 'Successfully created the function helloWorld. You can now change its code and configuration. To Invoke your function with a test event, choose "Test".' Below this, the breadcrumb navigation shows 'Lambda > Functions > helloWorld'. The function's ARN is displayed as 'arn:aws:lambda:us-east-1:470901565805:function:helloWorld'. The main section is titled 'helloWorld' and includes tabs for 'Configuration', 'Permissions', and 'Monitoring'. The 'Configuration' tab is active, showing a 'Designer' section with a visual representation of the function. This representation includes a box labeled 'helloWorld' with the Lambda icon, and a 'Layers' section below it showing '(0)' layers. To the left of the designer is a '+ Add trigger' button, and to the right is a '+ Add destination' button. At the bottom of the console, there is a footer with 'Feedback', 'English (US)', and copyright information: '© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

aws Services ▾ Resource Groups ▾

🔔 Eyal de Lara ▾ N. Virginia ▾ Support ▾

☑ Successfully created the function **helloWorld**. You can now change its code and configuration. To Invoke your function with a test event, choose "Test".

Lambda > Functions > helloWorld

ARN - arn:aws:lambda:us-east-1:470901565805:function:helloWorld

**helloWorld** Throttle Qualifiers ▾ Actions ▾ Select a test event ▾ Test Save

Configuration Permissions Monitoring

▼ Designer

helloWorld

Layers (0)

+ Add trigger + Add destination

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

# 1<sup>st</sup> Lambda Function

The screenshot displays the AWS Lambda console interface for a newly created function named "helloWorld". At the top, a green notification bar states: "Successfully created the function helloWorld. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the function name "helloWorld" is prominently displayed. To its right are buttons for "Throttle", "Qualifiers", "Actions", and a "Select a test event" dropdown, followed by a "Test" button. The main section is titled "Function code" and includes a "Deploy" button and an "Actions" dropdown. Below the title bar is a menu with options: File, Edit, Find, View, Go, Tools, Window, Test, and Deploy. On the left, an "Environment" sidebar shows a folder "helloWorld - /" containing a file "lambda\_function.py". The central editor displays the Python code for the lambda handler:

```
1 import json
2
3 def lambda_handler(event, context):
4     print('Received event: ' + json.dumps(event,indent=2))
5     if 'name' in event and len(event['name']) > 0:
6         name = event['name']
7     else:
8         name = "World"
9         greetings = "Hello " + name + '!'
10    print(greetings)
11    return greetings
12
```

At the bottom right of the code editor, it indicates "11:5 Python Spaces: 4". Below the code editor is an "Execution Result" section, which currently shows "No execution results yet". The footer of the console includes a "Feedback" link, "English (US)" language selection, and copyright information: "© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved." along with links to "Privacy Policy" and "Terms of Use".

# 1<sup>st</sup> Lambda Function

The screenshot displays the AWS Lambda Management Console interface. The browser address bar shows the URL `https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/helloWorld?tab=...`. The console header includes the AWS logo, navigation tabs for 'Services' and 'Resource Groups', and user information for 'Eyal de Lara' in the 'N. Virginia' region. The main content area is divided into two columns. The left column features the 'Execution role' section, which explains that it defines permissions and provides a dropdown to 'Choose an existing role'. Below this, an 'Existing role' section shows 'lambda\_basic\_execution' selected. The right column contains the 'Basic settings' section, which includes a 'Memory (MB)' slider set to 128 MB, a 'Timeout' of 0 minutes and 3 seconds, and a 'Description' text area. At the bottom of the console, there are four expandable sections: 'Network', 'Debugging and error handling', and two others partially visible. The footer contains a 'Feedback' button, 'English (US)' language selection, and copyright information for Amazon Web Services, Inc. (© 2008 - 2017), along with links to 'Privacy Policy' and 'Terms of Use'.

**Execution role**

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Choose an existing role ▼

**Existing role**

You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

lambda\_basic\_execution ▼

**Basic settings**

**Memory (MB)** [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 MB

**Timeout** [Info](#)

0 min 3 sec

**Description**

► **Network**

► **Debugging and error handling**

[Feedback](#) [English \(US\)](#) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

# Lambda Function Testing Arguments

The screenshot shows the AWS Lambda Management Console interface. A modal dialog titled "Configure test event" is open, allowing the user to set up a test event for a Lambda function. The dialog includes a description, radio buttons for creating or editing test events, a dropdown for the event template, a text field for the event name, and a code editor for the event payload.

**Configure test event**

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☒ Create new test event  
☐ Edit saved test events

Event template  
Hello World

Event name  
helloWorldEvent

```
1 {  
2   "name": "Eyal"  
3 }
```

The background shows the AWS console navigation bar with the "Lambda" service selected. The left sidebar contains a list of services, and the right sidebar shows the "Logs" section for the selected function.

# Lambda Function Testing

The screenshot displays the AWS Lambda Management Console interface. The browser address bar shows the URL `https://console.aws.amazon.com/lambda/home?region=us-east-1#/function/helloWord`. The console header includes the AWS logo, navigation tabs for 'Services' and 'Resource Groups', and user information for 'Eyal de Lara' in the 'N. Virginia' region. The main content area is titled 'helloWord' and shows the function's ARN: `arn:aws:lambda:us-east-1:470901565805:function:helloWord`. Below the title, there are buttons for 'Throttle', 'Qualifiers', 'Actions', and a dropdown menu currently set to 'helloWorldEvent', along with 'Test' and 'Save' buttons. A green banner indicates a successful execution: 'Execution result: succeeded (logs)'. Under the 'Details' section, it states: 'The section below shows the result returned by your function execution.' followed by a text box containing the output: `"Hello World!"`. A 'Summary' section provides execution metrics in a two-column table:

Code SHA-256	Request ID
D8wUy77Ts8/Mg1nd5uC9+kfKSuJdxvApwHaKhwi6DjA=	5c2a854b-dd2c-11e8-9a1f-138b16ec1863
Duration	Billed duration
0.47 ms	100 ms
Resources configured	Max memory used
128 MB	21 MB

Below the summary, the 'Log output' section explains that the logs correspond to a single row in the CloudWatch log group. The log output itself is shown in a text box:

```
START RequestId: 5c2a854b-dd2c-11e8-9a1f-138b16ec1863 Version: $LATEST
Received event: {
  "other": "Eyal"
}
Hello World!
```

The footer of the console includes a 'Feedback' button, a language selector set to 'English (US)', and copyright information: '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.' along with links to 'Privacy Policy' and 'Terms of Use'.

# Lambda Command Line Interface (CLI)

---

- Download from <http://aws.amazon.com/cli>

- **Configure credentials**

Create access key on AWS Console

> aws configure

- **Call function**

> aws lambda list-functions

> aws lambda invoke --function-name helloWorld --payload '{}' output.txt

> aws lambda invoke --function-name helloWorld --payload '{"name":"Eyal"}' output.txt

# Reacting to S3 Events

The screenshot shows the AWS Lambda Management Console interface. The browser address bar displays `https://console.aws.amazon.com/lambda/home?region=us-...`. The console header includes the AWS logo, navigation tabs for Services and Resource Groups, and user information for Eyal de Lara in N. Virginia. The main content area is titled 'Create function' and offers three options: 'Author from scratch', 'Blueprints' (selected), and 'AWS Serverless Application Repository'. The 'Blueprints' section features a search bar with the filter 'keyword: python3.6' and a list of six preconfigured function templates. The 's3-get-object-python3' template is highlighted, showing it is an Amazon S3 trigger that retrieves metadata for updated objects.

**Create function**

Author from scratch  
Start with a simple "hello world" example.

Blueprints  
Choose a preconfigured template as a starting point for your Lambda function.

AWS Serverless Application Repository  
Find and deploy serverless applications published by AWS, AWS partners, and other developers.

**Blueprints** info Export

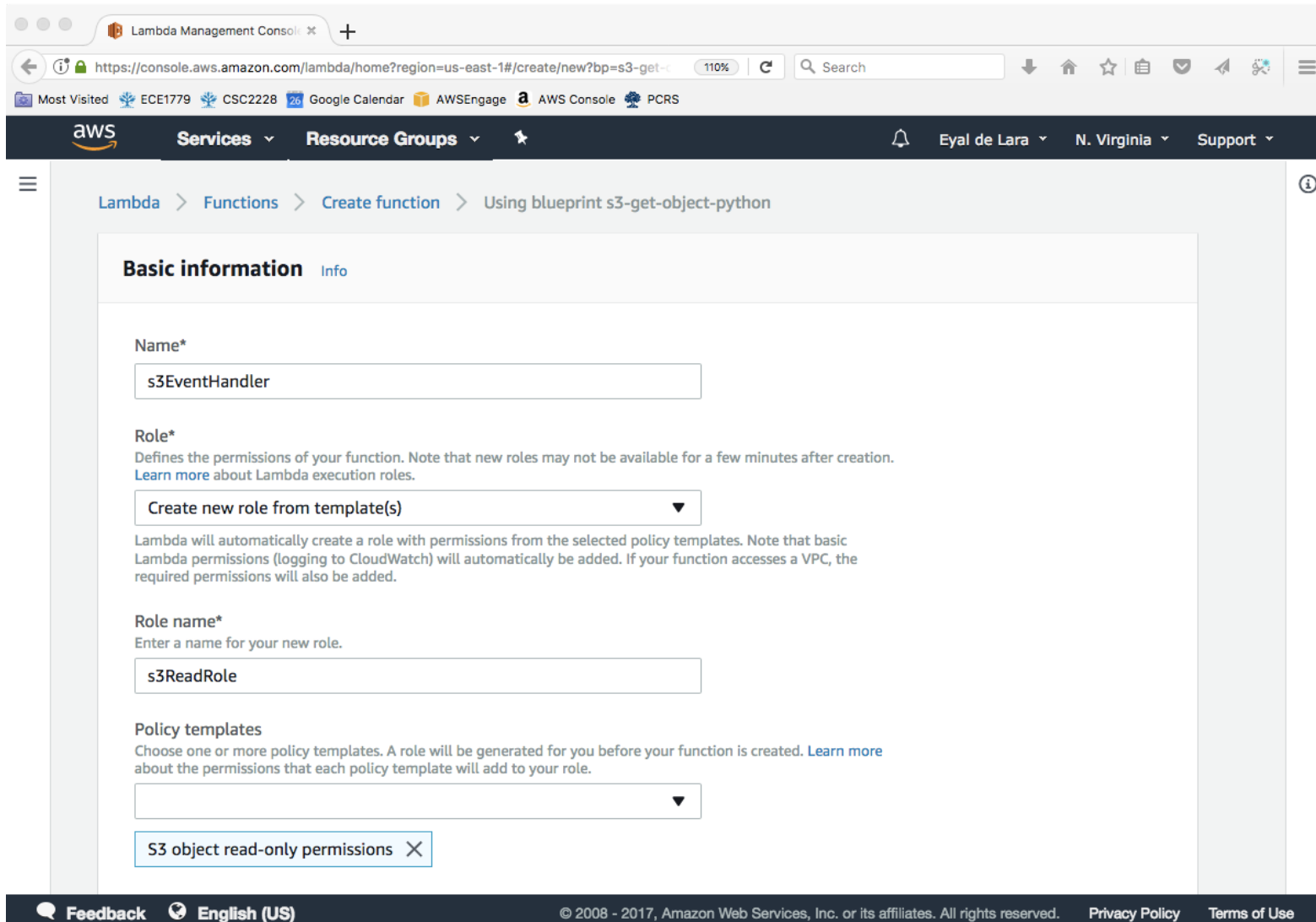
Search: Add filter ?

keyword: python3.6

<b>lambda-canary-python3</b> Performs a periodic check of the given site, erroring out on test failure. python3.6 · cron · testing	<b>microservice-http-endpoint-python3</b> A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway. python3.6 · api-gateway
<b>hello-world-python3</b> A starter AWS Lambda function. python3.6	<b>dynamodb-process-stream-python3</b> An Amazon DynamoDB trigger that logs the updates made to a table. python3.6 · dynamodb
<b>cloudwatch-alarm-to-slack-python3</b> An Amazon SNS trigger that sends CloudWatch alarm notifications to Slack. python3.6 · cloudwatch · slack	<b>s3-get-object-python3</b> An Amazon S3 trigger that retrieves metadata for the object that has been updated. python3.6 · s3

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

# Reacting to S3 Events



Lambda Management Console

https://console.aws.amazon.com/lambda/home?region=us-east-1#/create/new?bp=s3-get-object-python

Services Resource Groups

Eyal de Lara N. Virginia Support

Lambda > Functions > Create function > Using blueprint s3-get-object-python

### Basic information [Info](#)

**Name\***

s3EventHandler

**Role\***

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create new role from template(s)

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

**Role name\***

Enter a name for your new role.

s3ReadRole

**Policy templates**

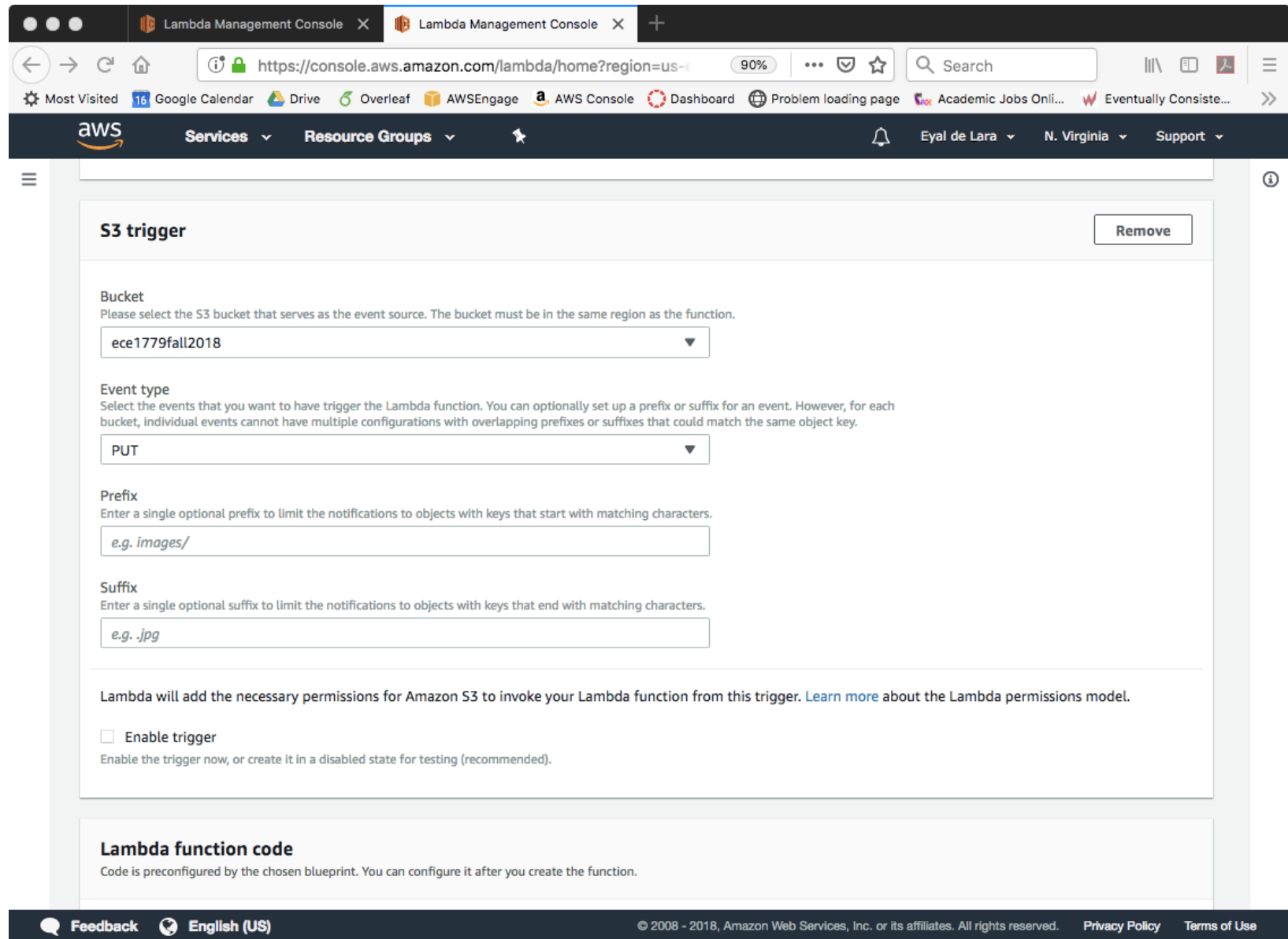
Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

S3 object read-only permissions

Feedback English (US) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



# Reacting to S3 Events

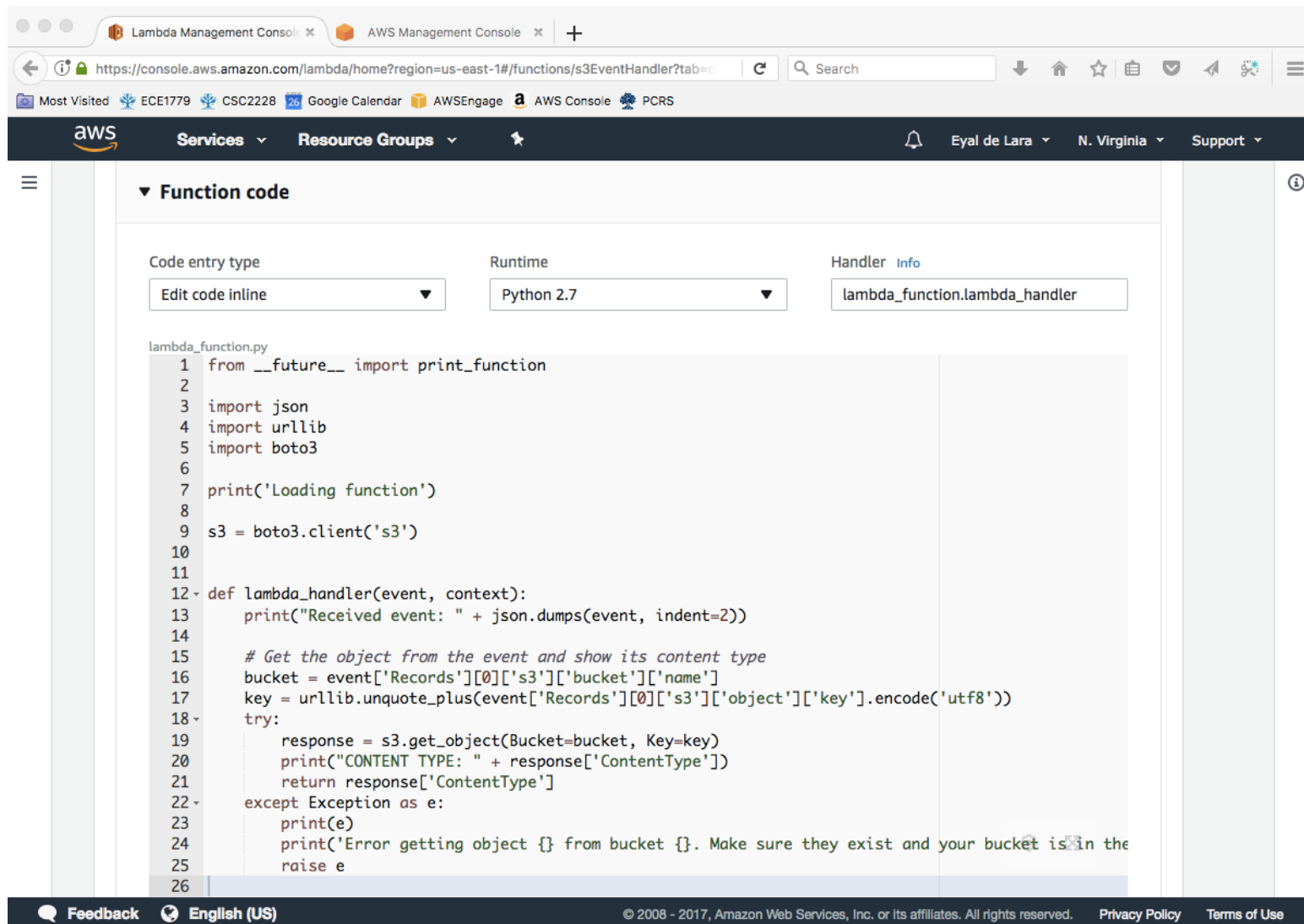


The screenshot shows the AWS Lambda Management Console interface. The browser address bar displays the URL `https://console.aws.amazon.com/lambda/home?region=us-...`. The console header includes the AWS logo, navigation tabs for Services and Resource Groups, and user information for Eyal de Lara in the N. Virginia region. The main content area is titled "S3 trigger" and includes a "Remove" button. The configuration section contains the following fields:

- Bucket:** A dropdown menu showing "ece1779fall2018".
- Event type:** A dropdown menu showing "PUT".
- Prefix:** A text input field containing "e.g. images/".
- Suffix:** A text input field containing "e.g. .jpg".

Below the input fields, a note states: "Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model." At the bottom of the configuration section, there is a checkbox labeled "Enable trigger" with the text "Enable the trigger now, or create it in a disabled state for testing (recommended)." The footer of the console includes a "Feedback" link, the language "English (US)", and copyright information for Amazon Web Services, Inc. (© 2008 - 2018).

# Reacting to S3 Events






The screenshot displays the AWS Lambda Management Console interface. The browser address bar shows the URL `https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/s3EventHandler?tab=c`. The console header includes the AWS logo, navigation tabs for Services and Resource Groups, and user information for Eyal de Lara in the N. Virginia region. The main content area is titled "Function code" and shows the configuration for the `s3EventHandler` function. The "Code entry type" is set to "Edit code inline", the "Runtime" is "Python 2.7", and the "Handler" is `lambda_function.lambda_handler`. The code editor displays the following Python code:

```
lambda_function.py
1 from __future__ import print_function
2
3 import json
4 import urllib
5 import boto3
6
7 print('Loading function')
8
9 s3 = boto3.client('s3')
10
11
12 def lambda_handler(event, context):
13     print("Received event: " + json.dumps(event, indent=2))
14
15     # Get the object from the event and show its content type
16     bucket = event['Records'][0]['s3']['bucket']['name']
17     key = urllib.unquote_plus(event['Records'][0]['s3']['object']['key'].encode('utf8'))
18     try:
19         response = s3.get_object(Bucket=bucket, Key=key)
20         print("CONTENT TYPE: " + response['ContentType'])
21         return response['ContentType']
22     except Exception as e:
23         print(e)
24         print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as your function.')
25         raise e
26
```

The footer of the console includes a Feedback button, language selection (English (US)), and copyright information: © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Links for Privacy Policy and Terms of Use are also present.

# Reacting to S3 Events

 Services ▾ Resource Groups ▾ 

 Eyal de Lara ▾ N. Virginia ▾ Support ▾



## s3EventHandler

Throttle Qualifiers ▾ Actions ▾ Select a test event ▾ Test Save 

 Congratulations! Your Lambda function "s3EventHandler" has been successfully created and configured with ece1779winter2020 as a trigger in a disabled state. We recommend testing the function behavior before enabling the trigger. 

Configuration

Permissions

Monitoring

▼ Designer

 S3 (2) + Add trigger

 s3EventHandler

 Layers (0)

+ Add destination

 Feedback

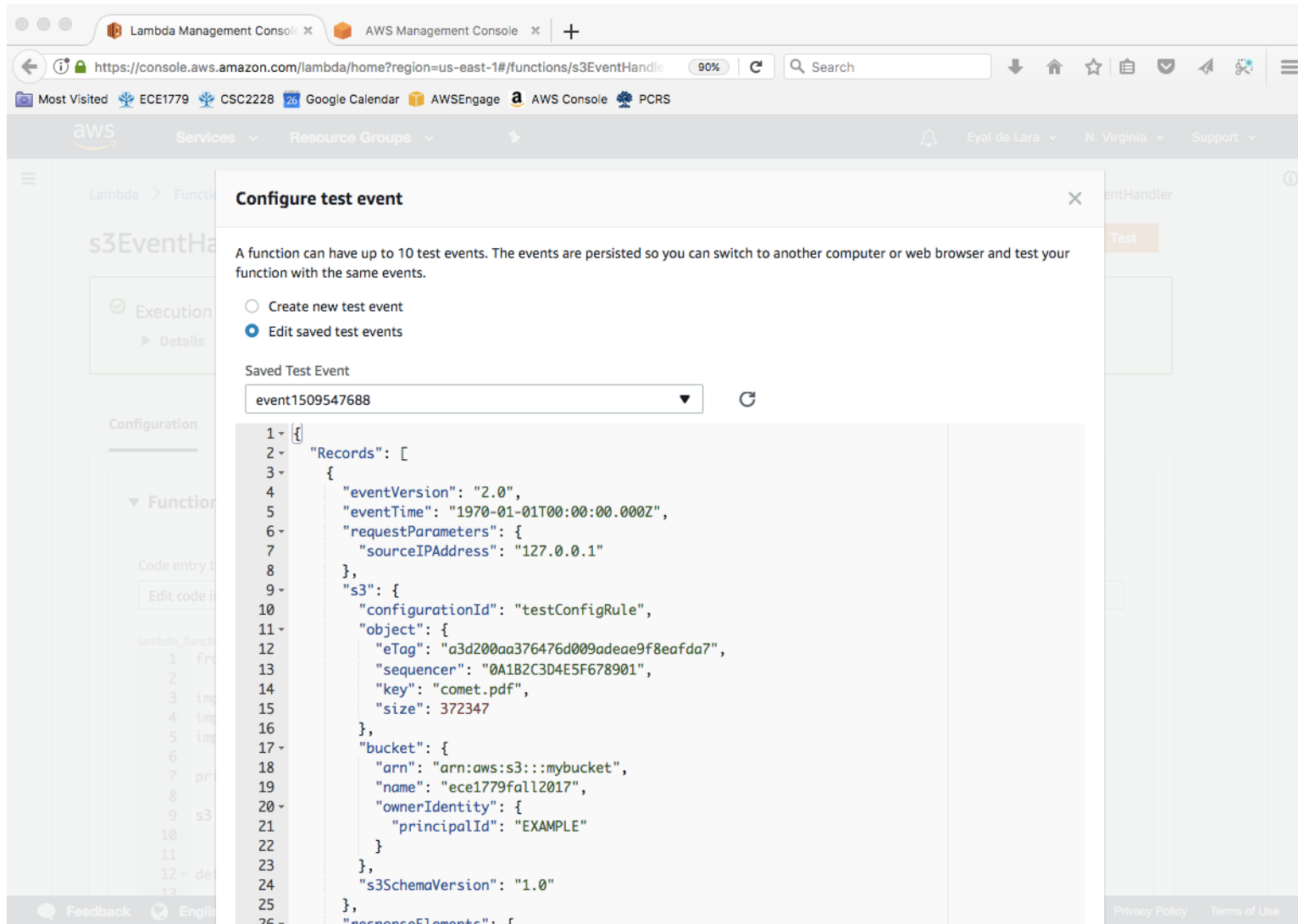
 English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

[Privacy Policy](#)

[Terms of Use](#)

# Reacting to S3 Events



The screenshot shows the AWS Lambda Management Console interface. A modal dialog titled "Configure test event" is open, displaying options to "Create new test event" or "Edit saved test events". The "Edit saved test events" option is selected. Below this, a dropdown menu shows "event1509547688" as the selected saved test event. The main area of the dialog displays a JSON event structure for an S3 event, with line numbers 1 through 26 on the left. The JSON structure is as follows:

```
1 {
2   "Records": [
3     {
4       "eventVersion": "2.0",
5       "eventTime": "1970-01-01T00:00:00.000Z",
6       "requestParameters": {
7         "sourceIPAddress": "127.0.0.1"
8       },
9       "s3": {
10        "configurationId": "testConfigRule",
11        "object": {
12          "eTag": "a3d200aa376476d009adeae9f8eafda7",
13          "sequencer": "0A1B2C3D4E5F678901",
14          "key": "comet.pdf",
15          "size": 372347
16        },
17        "bucket": {
18          "arn": "arn:aws:s3:::mybucket",
19          "name": "ece1779fall2017",
20          "ownerIdentity": {
21            "principalId": "EXAMPLE"
22          }
23        },
24        "s3SchemaVersion": "1.0"
25      },
26      "responseElements": {
```

The background shows the Lambda console for a function named "s3EventHa". The "Configuration" tab is active, and the "Function" section is visible. The bottom of the console shows a "Feedback" button and a language selector set to "English".

# Reacting to S3 Events

The screenshot displays the AWS Lambda Management Console interface. The browser address bar shows the URL `https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/s3EventHandler`. The console header includes the AWS logo, navigation tabs for Services and Resource Groups, and user information for Eyal de Lara in the N. Virginia region. The breadcrumb trail indicates the path: Lambda > Functions > s3EventHandler. The function's ARN is `arn:aws:lambda:us-east-1:470901565805:function:s3EventHandler`.

The main content area shows the execution result for the function `s3EventHandler`. A green checkmark icon indicates a successful execution. The event ID is `event1509547688`, and a `Test` button is available. The execution result is displayed in a dashed box as `"application/pdf"`.

Below the result, a **Summary** section provides details about the function's execution:

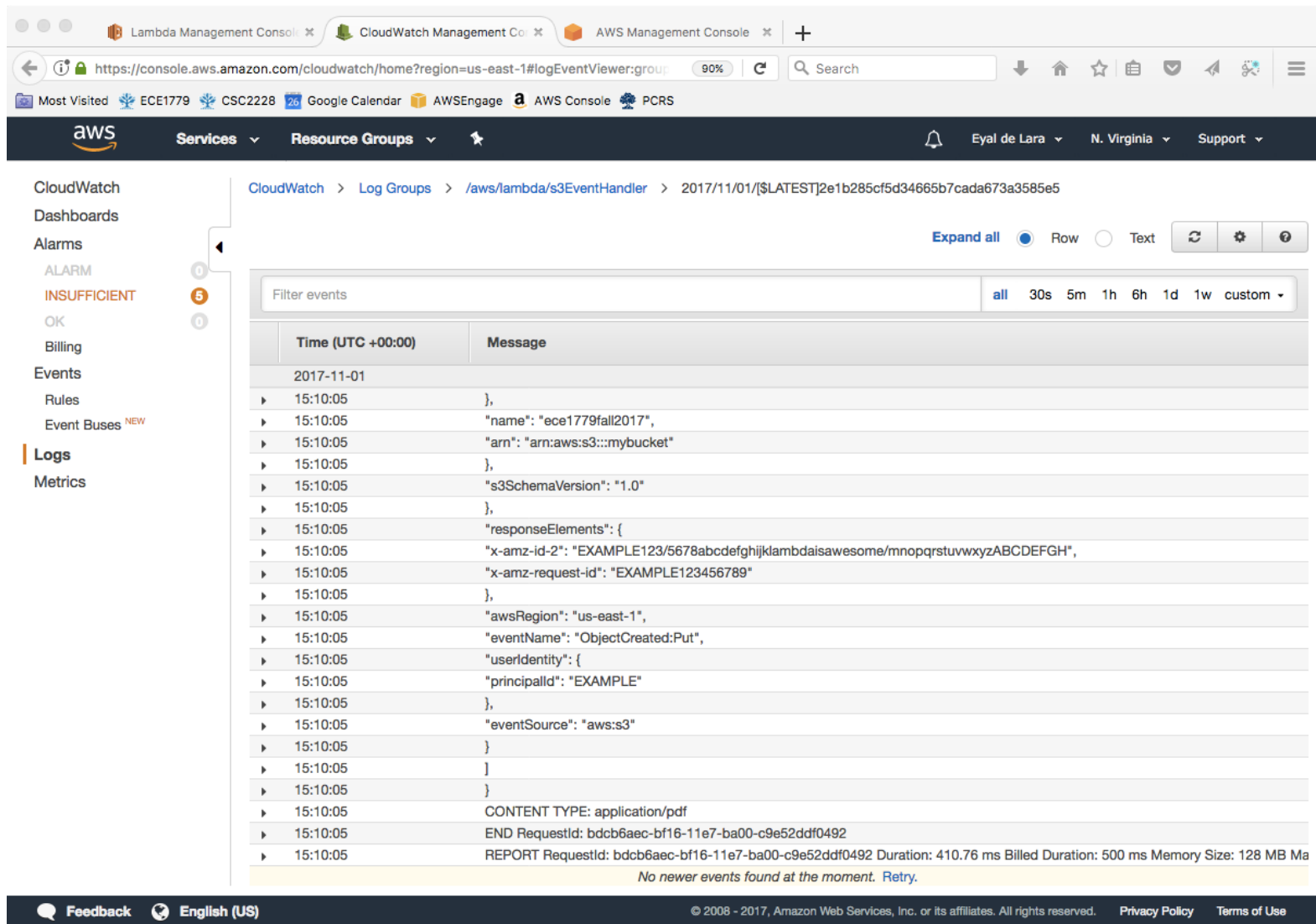
- Code SHA-256:** `posOGV7qe1JfL0Th1BMwRxxVQ0XRrb2pGgl4mcsDrN1U=`
- Request ID:** `bdc6baec-bf16-11e7-ba00-c9e52ddf0492`
- Duration:** 410.76 ms
- Billed duration:** 500 ms
- Resources configured:** 128 MB

A **Log output** section shows the logging calls from the function code, corresponding to a single row in the CloudWatch log group. The log output is as follows:

```
START RequestId: bdc6baec-bf16-11e7-ba00-c9e52ddf0492 Version: $LATEST
Received event: {
  "Records": [
    {
      "eventVersion": "2.0",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "s3": {
        "configurationId": "testConfRule"
      }
    }
  ]
}
```




The footer of the console includes a `Feedback` link, the language `English (US)`, and copyright information: © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Links for `Privacy Policy` and `Terms of Use` are also present.

# Reacting to S3 Events



The screenshot displays the AWS Management Console interface. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information. The left sidebar shows the 'CloudWatch' menu with options like Dashboards, Alarms, and Logs. The main content area is titled 'CloudWatch > Log Groups > /aws/lambda/s3EventHandler > 2017/11/01/[\$LATEST]2e1b285cf5d34665b7cada673a3585e5'. It features a table of log events with columns for 'Time (UTC +00:00)' and 'Message'. The events are filtered for the date 2017-11-01 and show a sequence of log messages for an S3 event handler. The bottom of the console includes a footer with 'Feedback', 'English (US)', and copyright information.

CloudWatch > Log Groups > /aws/lambda/s3EventHandler > 2017/11/01/[\$LATEST]2e1b285cf5d34665b7cada673a3585e5

Expand all ☒ Row ☐ Text   

Filter events  all 30s 5m 1h 6h 1d 1w custom

Time (UTC +00:00)	Message
2017-11-01	
15:10:05	},
15:10:05	"name": "ece1779fall2017",
15:10:05	"arn": "arn:aws:s3:::mybucket"
15:10:05	},
15:10:05	"s3SchemaVersion": "1.0"
15:10:05	},
15:10:05	"responseElements": {
15:10:05	"x-amz-id-2": "EXAMPLE123/5678abcdefghijklmbdaisawesomemnopqrstuvwxyzABCDEFGH",
15:10:05	"x-amz-request-id": "EXAMPLE123456789"
15:10:05	},
15:10:05	"awsRegion": "us-east-1",
15:10:05	"eventName": "ObjectCreated:Put",
15:10:05	"userIdentity": {
15:10:05	"principalId": "EXAMPLE"
15:10:05	},
15:10:05	"eventSource": "aws:s3"
15:10:05	}
15:10:05	}
15:10:05	}
15:10:05	CONTENT TYPE: application/pdf
15:10:05	END RequestId: bdc6baec-bf16-11e7-ba00-c9e52ddf0492
15:10:05	REPORT RequestId: bdc6baec-bf16-11e7-ba00-c9e52ddf0492 Duration: 410.76 ms Billed Duration: 500 ms Memory Size: 128 MB Ma

No newer events found at the moment. [Retry](#).

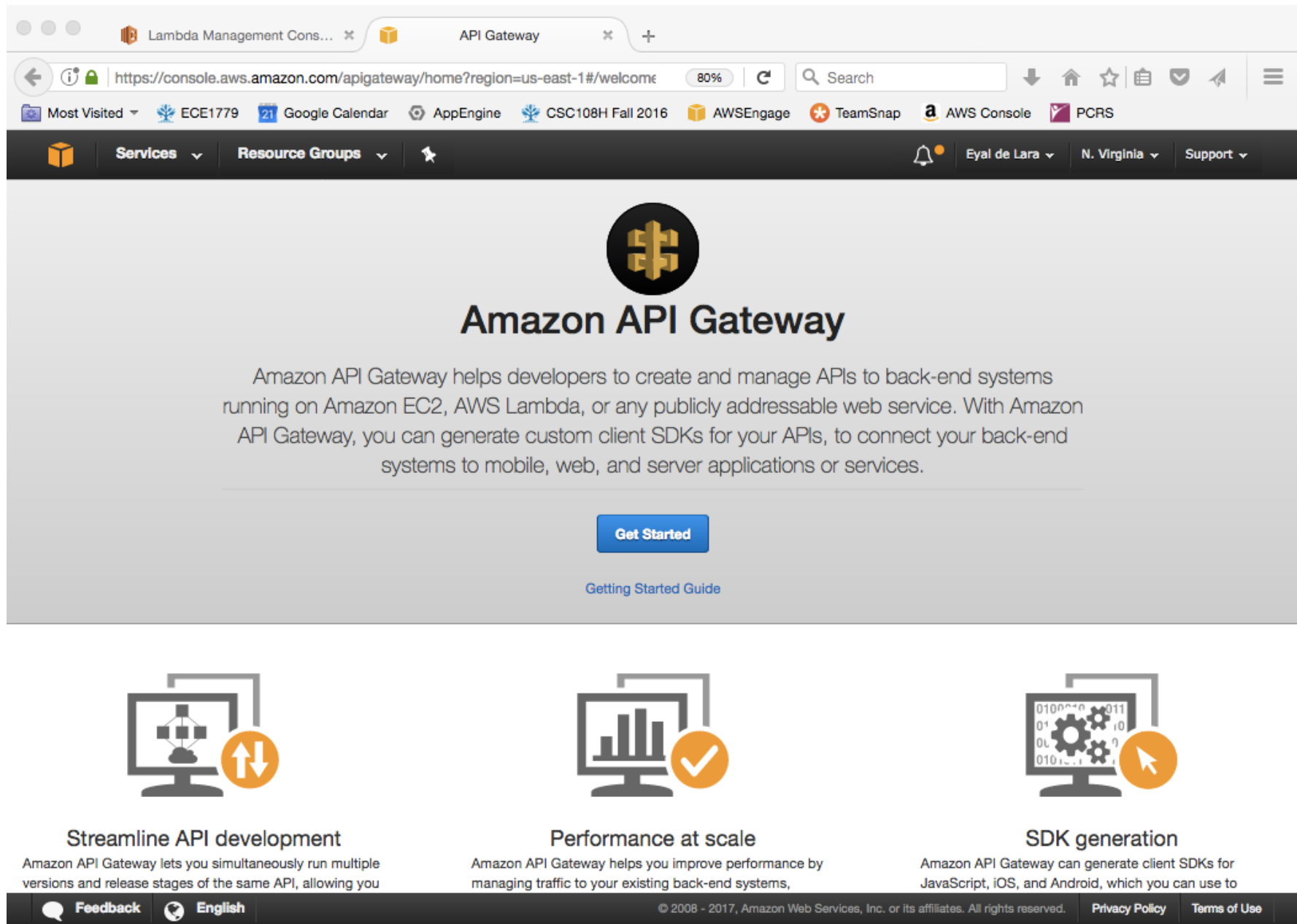
Feedback English (US) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

## Function as Web API

---

- Amazon API Gateway can be integrated with Lambda
- Turn http request into Lambda events
- Maps each Lambda Event into a URL endpoint

# Function as Web API



The screenshot shows the Amazon API Gateway console interface. At the top, there's a browser window with the URL `https://console.aws.amazon.com/apigateway/home?region=us-east-1#/welcome`. Below the browser window is a navigation bar with 'Services' and 'Resource Groups' dropdowns, and user information for 'Eyal de Lara' in 'N. Virginia'. The main content area features the Amazon API Gateway logo and a welcome message: 'Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services.' A 'Get Started' button and a 'Getting Started Guide' link are provided. Below this, three key features are highlighted with icons and text:

- Streamline API development**: Amazon API Gateway lets you simultaneously run multiple versions and release stages of the same API, allowing you
- Performance at scale**: Amazon API Gateway helps you improve performance by managing traffic to your existing back-end systems,
- SDK generation**: Amazon API Gateway can generate client SDKs for JavaScript, iOS, and Android, which you can use to

The footer contains a 'Feedback' button, a language selector set to 'English', and copyright information: '© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.' Links for 'Privacy Policy' and 'Terms of Use' are also present.



# Function as Web API

The screenshot displays the AWS API Gateway console interface. The browser's address bar shows the URL: `console.aws.amazon.com/apigateway/main/precreate?integration=u842mlm&region=us-east-1&routes=nhfz859...`. The console header includes the AWS logo, a 'Services' dropdown, a search bar with the placeholder text 'Search for services, features, marketplace products, and docs', and user information for 'Eyal de Lara' in the 'N. Virginia' region. A left-hand navigation pane is titled 'API Gateway' and contains links for 'APIs', 'Custom domain names', and 'VPC links'. The main content area is titled 'Choose an API type' and presents four options:

- HTTP API**: Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support. Works with the following: Lambda, HTTP backends. Buttons: Import, Build.
- WebSocket API**: Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards. Works with the following: Lambda, HTTP, AWS Services. Button: Build.
- REST API**: Develop a REST API where you gain complete control over the request and response along with API management capabilities. Works with the following: Lambda, HTTP, AWS Services. Buttons: Import, Build.
- REST API Private**: Create a REST API that is only accessible from within a VPC. Works with the following: Lambda, HTTP, AWS Services. Buttons: Import, Build.

The footer of the console contains links for 'Feedback', 'English (US)', and copyright information: '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.' It also includes links for 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

# Function as Web API

The screenshot shows the AWS API Gateway console in a web browser. The browser's address bar displays the URL: `console.aws.amazon.com/apigateway/main/create?integration=u842mlm&region=us-east-1&routes=nhfz859&st...`. The console interface includes a top navigation bar with the AWS logo, a search bar, and user information. On the left, a sidebar lists navigation options: 'API Gateway' (selected), 'APIs', 'Custom domain names', and 'VPC links'. The main content area is titled 'Configure routes' and shows a four-step process: 'Step 1: Create an API', 'Step 2: Configure routes' (current step), 'Step 3: Define stages', and 'Step 4: Review and create'. The 'Configure routes' section contains an 'Info' box explaining that routes consist of an HTTP method and a resource path, and lists supported methods (GET, POST, PUT, PATCH, HEAD, OPTIONS, DELETE, and ANY). Below this, a configuration table is shown with one row: Method 'ANY', Resource path '/helloworld', and Integration target 'helloworld'. An 'Add route' button is at the bottom left of the table. At the bottom right of the console, there are 'Cancel', 'Previous', and 'Next' buttons.

API Gateway

Services

Search for services, features, marketplace products, and docs [Option+S]

Eyal de Lara N. Virginia Support

API Gateway

APIs

Custom domain names

VPC links

Step 1  
[Create an API](#)

Step 2  
**Configure routes**

Step 3  
[Define stages](#)

Step 4  
[Review and create](#)

## Configure routes

**Configure routes** [Info](#)

API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method	Resource path	Integration target	
ANY	/helloworld	helloworld	<a href="#">Remove</a>

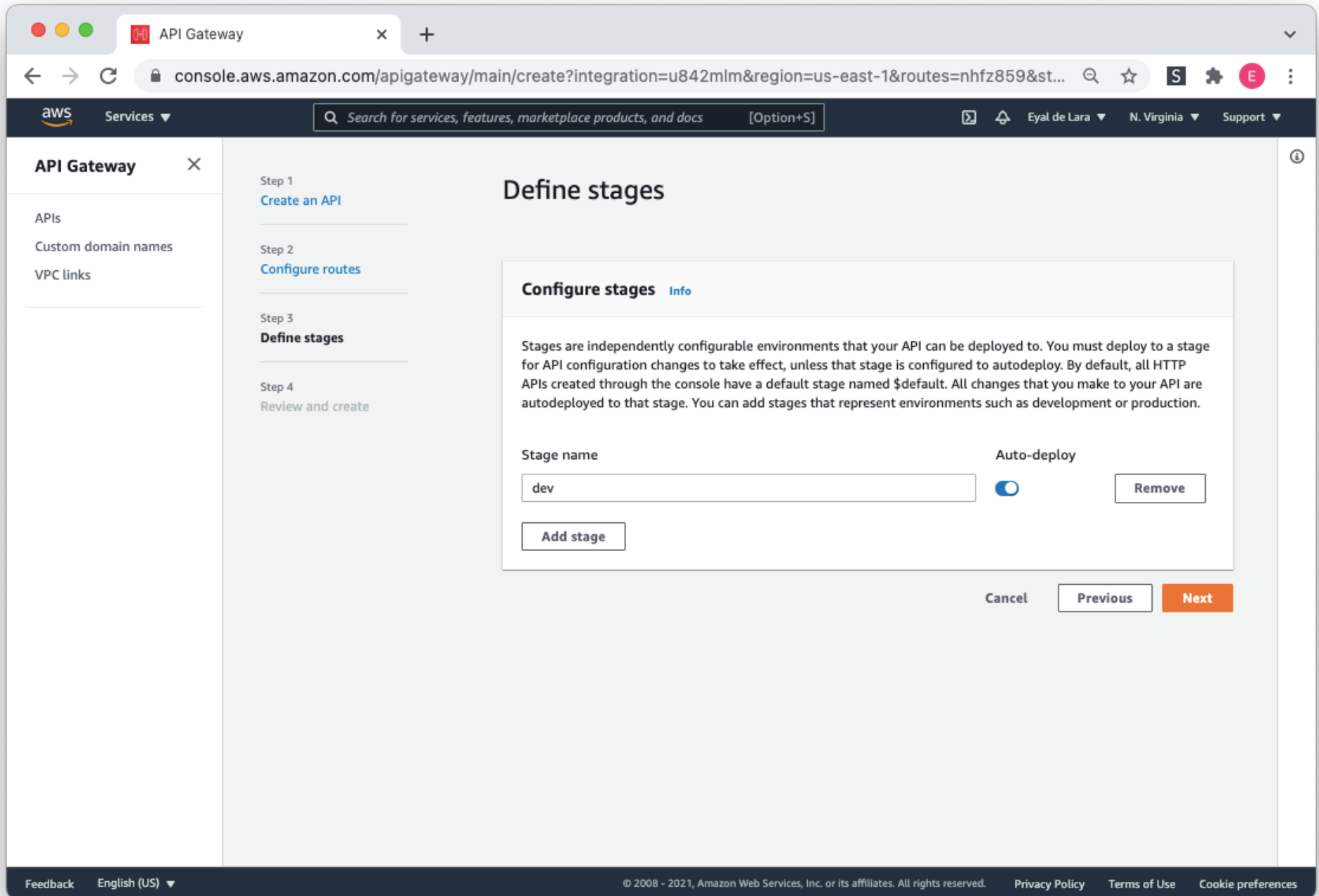
[Add route](#)

[Cancel](#) [Previous](#) [Next](#)

Feedback English (US)

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

# Function as Web API



# Function as Web API

The screenshot shows the AWS API Gateway console in a web browser. The browser's address bar displays the URL: `console.aws.amazon.com/apigateway/main/create?integration=u842mlm&region=us-east-1&routes=nhfz859&st...`. The console interface includes a top navigation bar with the AWS logo, a search bar, and user information. On the left, a sidebar lists 'API Gateway' and its sub-items: 'APIs', 'Custom domain names', and 'VPC links'. The main content area is titled 'Review and create' and shows a progress bar with four steps: 'Step 1: Create an API', 'Step 2: Configure routes', 'Step 3: Define stages', and 'Step 4: Review and create' (which is the current step). The 'Review and create' section contains three panels: 'API name and integrations' (with API name 'helloworldapi' and integration 'helloworld (Lambda)'), 'Routes' (with a single route 'ANY /helloworld -> helloworld (Lambda)'), and 'Stages' (with a single stage 'dev (Auto-deploy: enabled)'). Each panel has an 'Edit' button. At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Create'.

API Gateway

Services

Search for services, features, marketplace products, and docs [Option+S]

Eyal de Lara N. Virginia Support

API Gateway

APIs

Custom domain names

VPC links

Step 1  
[Create an API](#)

Step 2  
[Configure routes](#)

Step 3  
[Define stages](#)

Step 4  
**Review and create**

## Review and create

### API name and integrations [Edit](#)

API name  
helloworldapi

Integrations  
helloworld (Lambda)

### Routes [Edit](#)

Routes  
ANY /helloworld → helloworld (Lambda)

### Stages [Edit](#)

Stages  
dev (Auto-deploy: enabled)

Cancel Previous **Create**

Feedback English (US)

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

# Function as Web API

The screenshot shows the AWS API Gateway console interface. At the top, a green banner indicates 'Successfully created API helloworldapi (kvbizjlh3c)'. The left sidebar contains navigation links for 'APIs', 'Custom domain names', 'VPC links', and a 'Develop' section with links for 'Routes', 'Authorization', 'Integrations', 'CORS', 'Reimport', and 'Export'. Below these are 'Deploy' (Stages), 'Protect' (Throttling), and 'Monitor' (Metrics, Logging) sections. The main content area displays the details for the 'helloworldapi'.

**API details**

API ID	Protocol	Created
kvbizjlh3c	HTTP	2021-11-04
Description	Default endpoint	
No Description	Enabled	

**Stages for helloworldapi**

Find resources

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
dev	<a href="https://kvbizjlh3c.execute-api.us-east-1.amazonaws.com/dev">https://kvbizjlh3c.execute-api.us-east-1.amazonaws.com/dev</a>	na6dhj	enabled	2021-11-04

**Tags (0)**

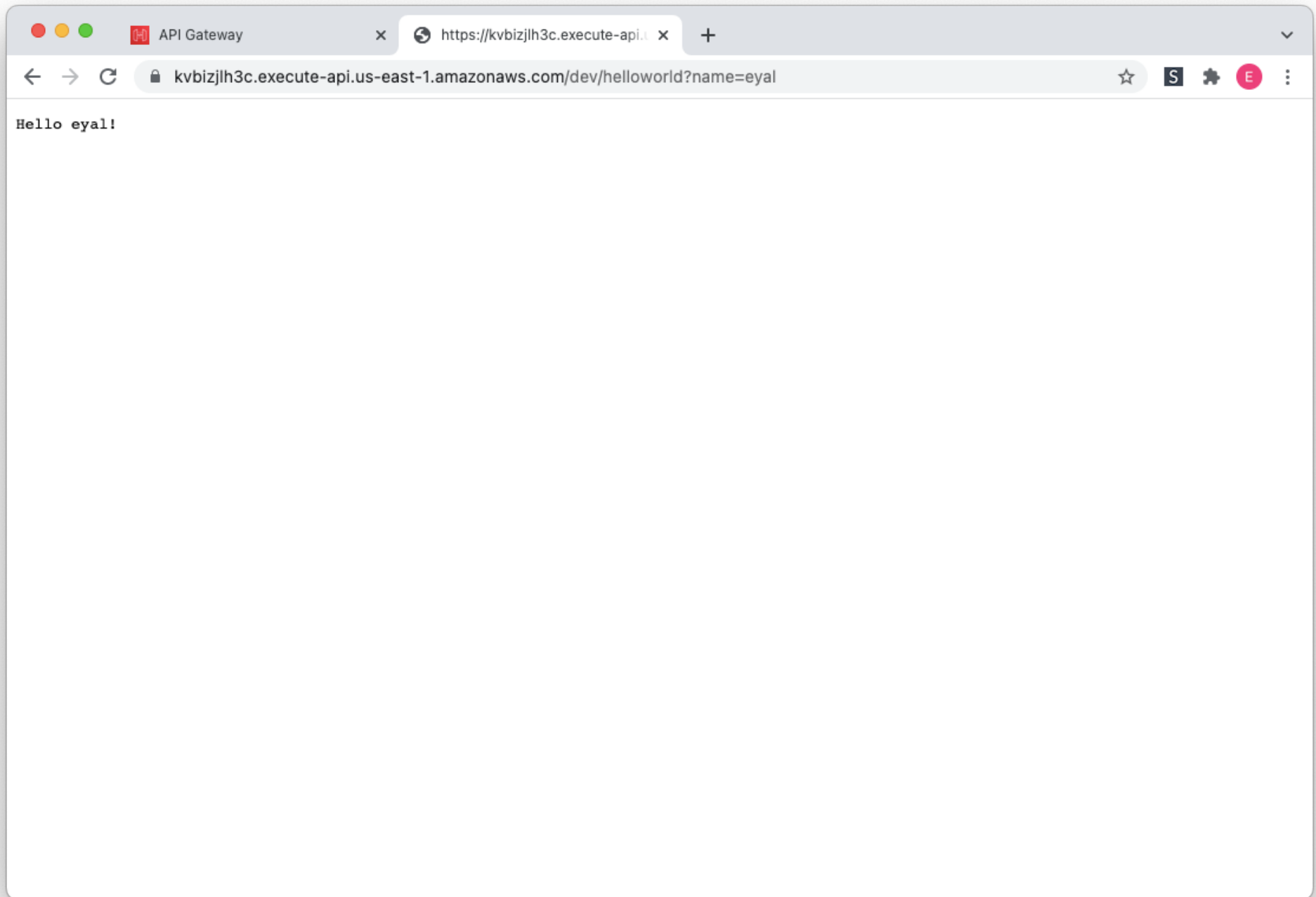
Find resources

Key	Value
-----	-------

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

# Function as Web API

---



# Zappa

---

- Deploy Python WSGI applications on AWS Lambda + API Gateway.
- <https://github.com/Miserlou/Zappa>
- Support limited to Python 3.6, 3.7, 3.8
- Steps:
  - Install and configure AWS CLI
  - Create flask directory structure
  - Create a python virtual environment:
    - > `virtualenv -p python3.8 venv`
    - > `source venv/bin/activate`
  - Install flask
    - > `pip install flask`
  - Install zappa
    - > `pip install zappa`
    - > `zappa init`

# Zappa

---

- zappa\_settings.json

```
{
  "dev": {
    "project_name": "flask_zappa",
    "keep_warm": false,
    "debug": true,
    "log_level": "DEBUG",
    "s3_bucket": "ece1779fall2017",
    "app_function": "app.webapp",
    "http_methods": ["GET", "POST"],
    "parameter_depth": 1,
    "timeout_seconds": 300,
    "memory_size": 128,
    "use_precompiled_packages": true
  }
}
```

- Deploy application to AWS Lambda
  - > zappa deploy dev