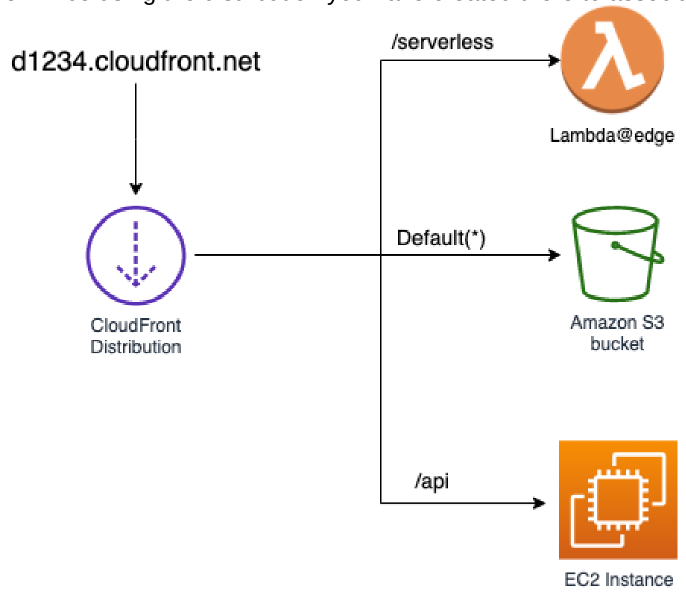
Generate content dynamically with Lambda@Edge

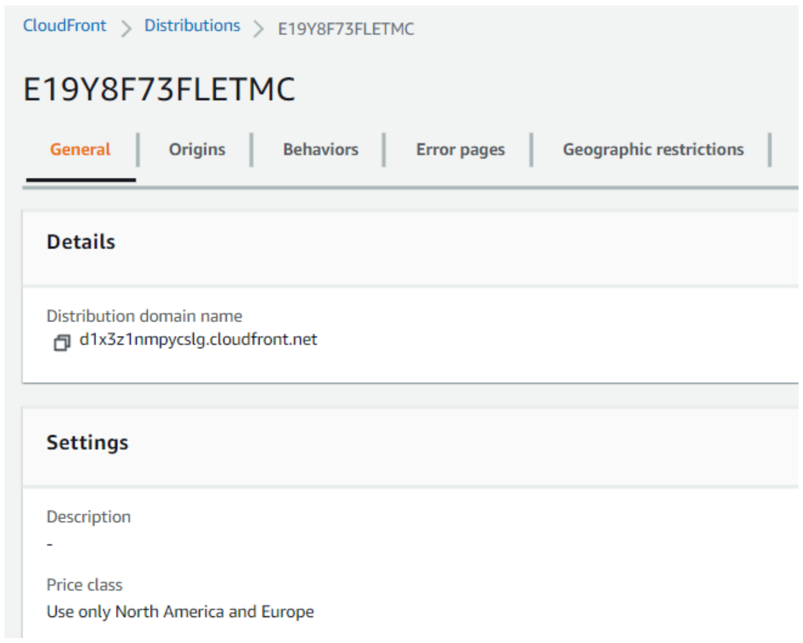
In this lab, you will use Lambda@Edge to dynamically generate the page created in Edge Content Acceleration experiment. Rather than a simple dump of the request headers, we will output the values in an HTML table. You will learn how to create Lamdba@Edge permissions, deploy functional Lambda@Edge code, and test it. Finally, you will be able to view function logs, metrics, and troubleshoot your code.
As a pre-requisite to this lab, you must first complete the "Edge Content Acceleration" experiment, as we will be using the distribution you have created there to associate our



lambda function.

Create a new Cache Behavior
In this section you will create a new cache behavior in the CloudFront distribution that you created in EdgeContentAcceleration Experiment. You will use this cache behavior to associate the Lambda@Edge function that you will be writing.

1. Go to the CloudFront console and select the distribution you created in Edge Content Acceleration Experiment. View our CloudFront Edge Cache Distribution



CloudFront > Distributions > E19Y8F73FLETMC

E19Y8F73FLETMC

Details

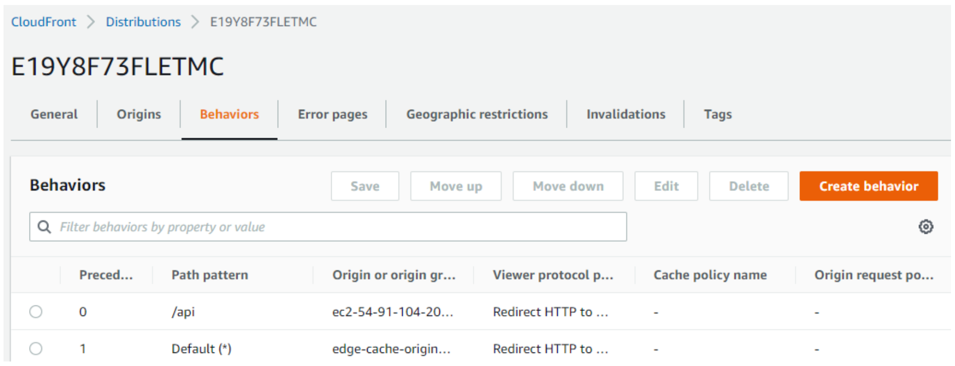
Distribution domain name
d1x3z1nmpycslg.cloudfront.net

Settings

Description
-

Price class
Use only North America and Europe

2. Select the **Behaviors** Tab



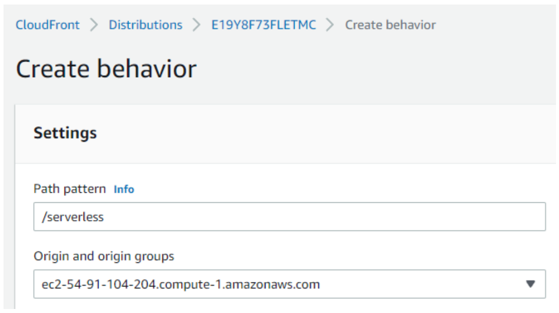
CloudFront > Distributions > E19Y8F73FLETMC

E19Y8F73FLETMC

Behaviors

	Preced...	Path pattern	Origin or origin gr...	Viewer protocol p...	Cache policy name	Origin request po...
○	0	/api	ec2-54-91-104-20...	Redirect HTTP to ...	-	-
○	1	Default (*)	edge-cache-origin...	Redirect HTTP to ...	-	-

3. Select **Create behavior**


CloudFront > Distributions > E19Y8F73FLETMC > Create behavior

Create behavior

Settings

Path pattern Info

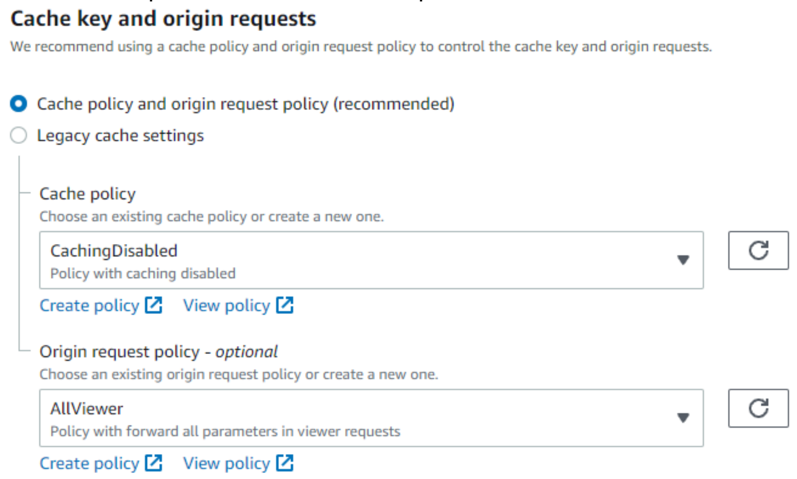
/serverless

Origin and origin groups

ec2-54-91-104-204.compute-1.amazonaws.com

4. Update the following

- Path : /serverless
- Origin: <Select the EC2 origin created in our Edge Content Acceleration and used in our /api behaviour from the drop down>



Cache key and origin requests

We recommend using a cache policy and origin request policy to control the cache key and origin requests.

[✓ Cache policy and origin request policy (recommended)]

Legacy cache settings

Cache policy

Choose an existing cache policy or create a new one.

CachingDisabled

Policy with caching disabled

Origin request policy - *optional*

Choose an existing origin request policy or create a new one.

AllViewer

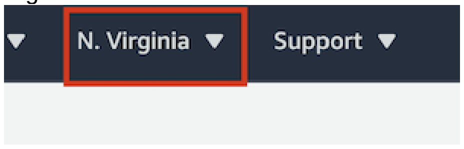
Policy with forward all parameters in viewer requests

- Cache Policy: **CachingDisabled**
- Origin Request Policy: **AllViewer**
- Leave all other setting at their default
- Select **Create behavior**

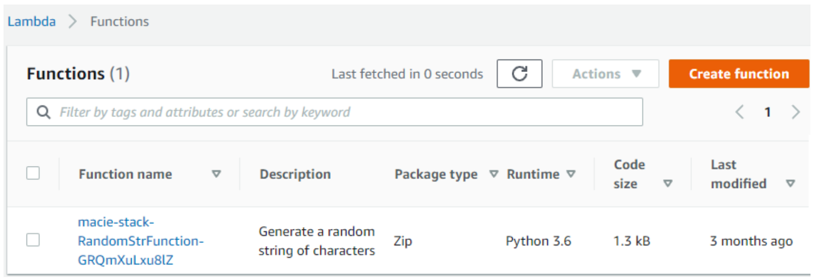
Commented [GN1]:

Create a Lambda@Edge function
In this section we create Lambda@Edge function.

1. Go to the AWS Console and make sure you are in the US-EAST-1 N. Virginia region.

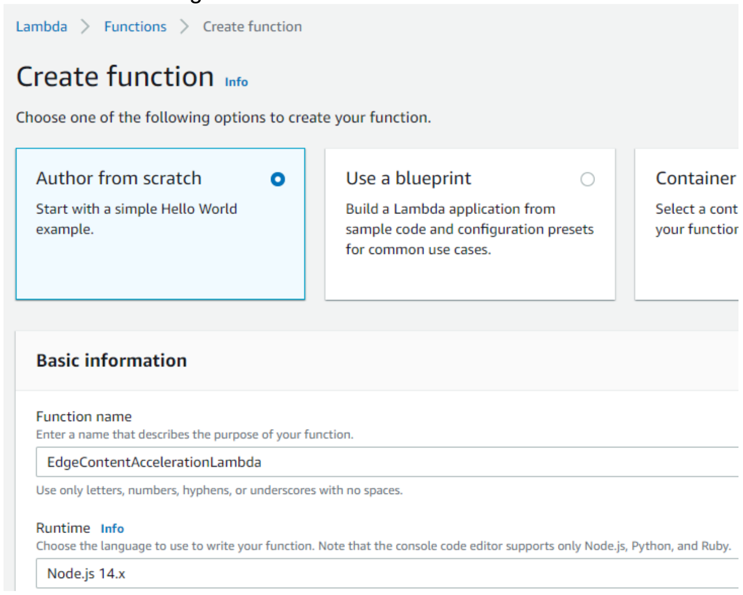


2. Navigate to the Lambda console


Lambda > Functions
Functions (1)
Last fetched in 0 seconds
Actions
Create function
Filter by tags and attributes or search by keyword
1

	Function name	Description	Package type	Runtime	Code size	Last modified
☐	macie-stack-RandomStrFunction-GRQmXuLxu8lZ	Generate a random string of characters	Zip	Python 3.6	1.3 kB	3 months ago

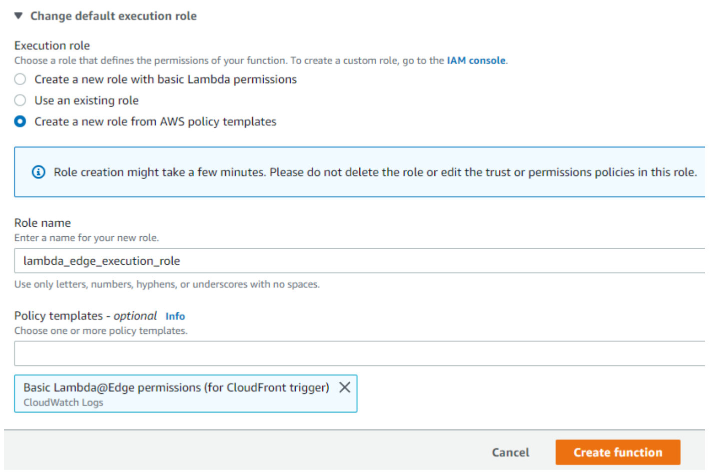
3. Enter the following:


Lambda > Functions > Create function
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.
Container
Select a cont
your functior
Basic information

Function name
Enter a name that describes the purpose of your function.
EdgeContentAccelerationLambda
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 14.x

- Select **Author from scratch**
- Name: **EdgeContentAccelerationLambda**
- Runtime: **Node.js 14.x**


▼ 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

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name

Enter a name for your new role.

lambda_edge_execution_role

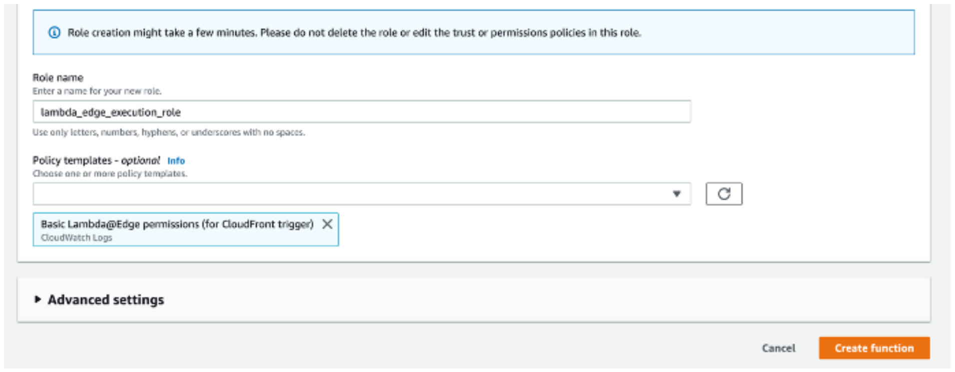
Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - *optional* Info

Choose one or more policy templates.

Basic Lambda@Edge permissions (for CloudFront trigger)
CloudWatch Logs

- Role: **Create new role from AWS policy templates**
- Role name: **lambda_edge_execution_role**
- Policy templates: **Basic Lambda@Edge permissions (for CloudFront trigger)**
- Click **Create function**


Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name

Enter a name for your new role.

lambda_edge_execution_role

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

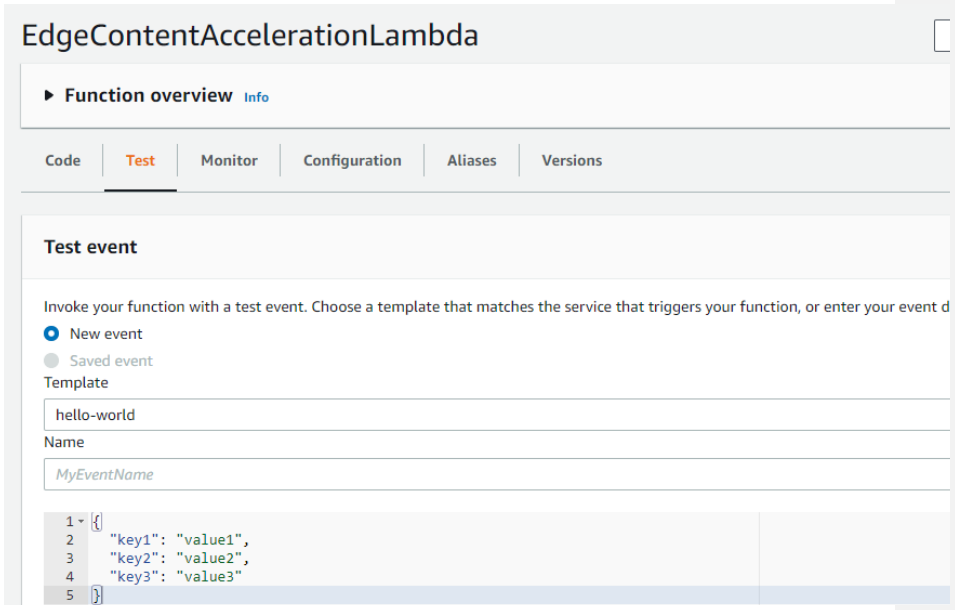
Policy templates - *optional* Info

Choose one or more policy templates.

Basic Lambda@Edge permissions (for CloudFront trigger)
CloudWatch Logs

You have now created a new IAM role that will be used to allow CloudFront to invoke Lambda and write logs to CloudWatch.

4. Configure a new test event that can be used to test your function locally from the Lambda console:


EdgeContentAccelerationLambda

Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event d

[✓ New event] Saved event

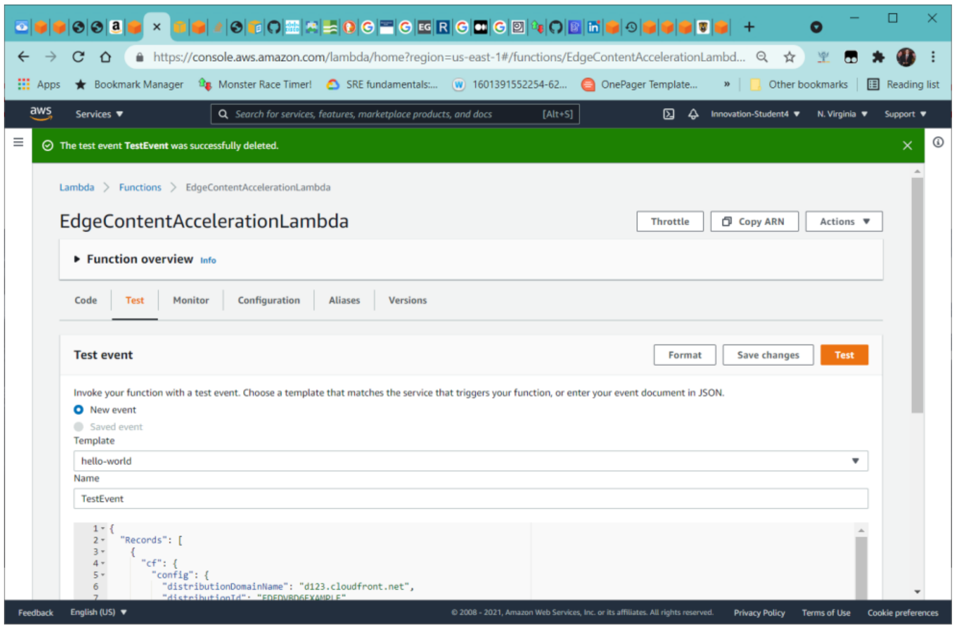
Template: hello-world

Name: MyEventName

```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

- Choose the Test tab.
- Event Name : “TestEvent”
- Replace the Hello World JSON with the following or copy the contents of the **EdgeContentAccelerationServerlessTest.txt** from the GitHub repo:

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d123.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-request",
          "requestId":
"MRVMF7KydIvxMWfJIglgwHQwZsbG2IhRJ07sn9AkKUFSHS9EXAMPLE=="
        },
        "request": {
          "clientIp": "2001:0db8:85a3:0:0:8a2e:0370:7334",
          "querystring": "size=large",
          "uri": "/picture.jpg",
          "method": "GET",
          "headers": {
            "host": [
              {
                "key": "Host",
                "value": "d111111abcdef8.cloudfront.net"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "curl/7.51.0"
              }
            ]
          },
          "origin": {
            "custom": {
              "customHeaders": {
                "my-origin-custom-header": [
                  {
                    "key": "My-Origin-Custom-Header",
                    "value": "Test"
                  }
                ]
              },
              "domainName": "example.com",
              "keepaliveTimeout": 5,
              "path": "/custom_path",
              "port": 443,
              "protocol": "https",
              "readTimeout": 5,
              "sslProtocols": [
                "TLSv1",
                "TLSv1.1"
              ]
            },
            "s3": {
              "authMethod": "origin-access-identity",
              "customHeaders": {
                "my-origin-custom-header": [
                  {
                    "key": "My-Origin-Custom-Header",
                    "value": "Test"
                  }
                ]
              },
              "domainName": "my-bucket.s3.amazonaws.com",
              "path": "/s3_path",
              "region": "us-east-1"
            }
          }
        }
      }
    }
  ]
}
```


The test event **TestEvent** was successfully deleted.

Lambda > Functions > EdgeContentAccelerationLambda

EdgeContentAccelerationLambda

Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

New event

Saved event

Template: hello-world

Name: TestEvent

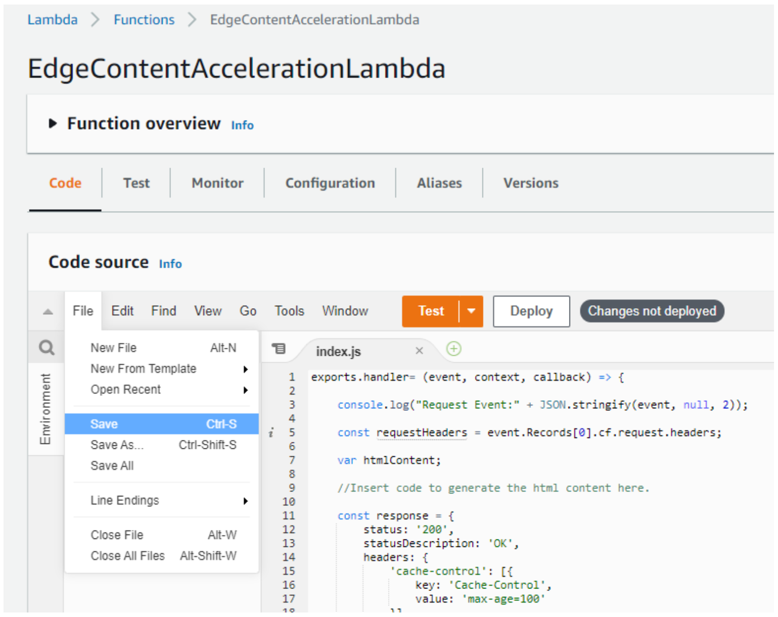
```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d123.cloudfront.net",
```

- Select **Save changes**

5. Now let's write a Functions as a Service (FaaS) AKA Serverless AKA Lambd to generate the html produced in the Edge Content Acceleration experiment.

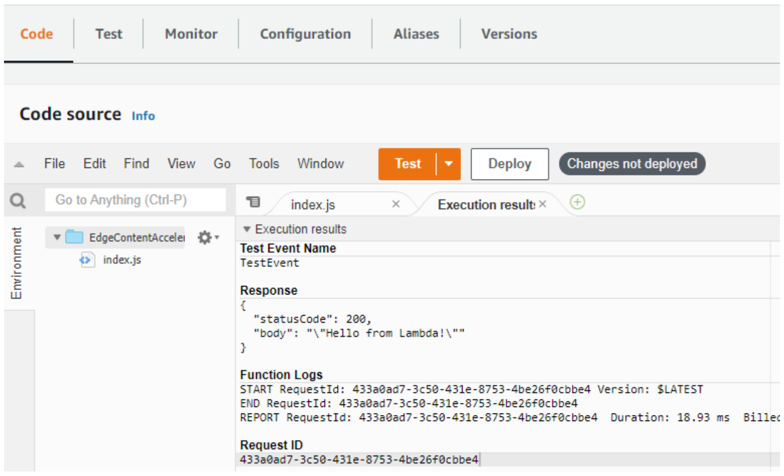
- Copy and paste the code below into the function code window or use the **EdgeContentAccelerationServerless.txt** for the source in the GitHub repo

```
exports.handler= (event, context, callback) => {

    console.log("Request Event:" + JSON.stringify(event, null, 2));

    const requestHeaders = event.Records[0].cf.request.headers;

    var htmlContent;

    //Insert code to generate the html content here.

    const response = {
        status: '200',
        statusDescription: 'OK',
        headers: {
            'cache-control': [{
                key: 'Cache-Control',
                value: 'max-age=100'
            }],
            'content-type': [{
                key: 'Content-Type',
                value: 'text/html'
            }],
            'content-encoding': [{
                key: 'Content-Encoding',
                value: 'UTF-8'
            }],
        },
        body: htmlContent,
    };

    callback(null, response);
}
```



```
exports.handler= (event, context, callback) => {

    console.log("Request Event:" + JSON.stringify(event, null, 2));

    const requestHeaders = event.Records[0].cf.request.headers;

    var htmlContent;

    //Insert code to generate the html content here.

    const response = {
        status: '200',
        statusDescription: 'OK',
        headers: {
            'cache-control': [{
                key: 'Cache-Control',
                value: 'max-age=100'
```

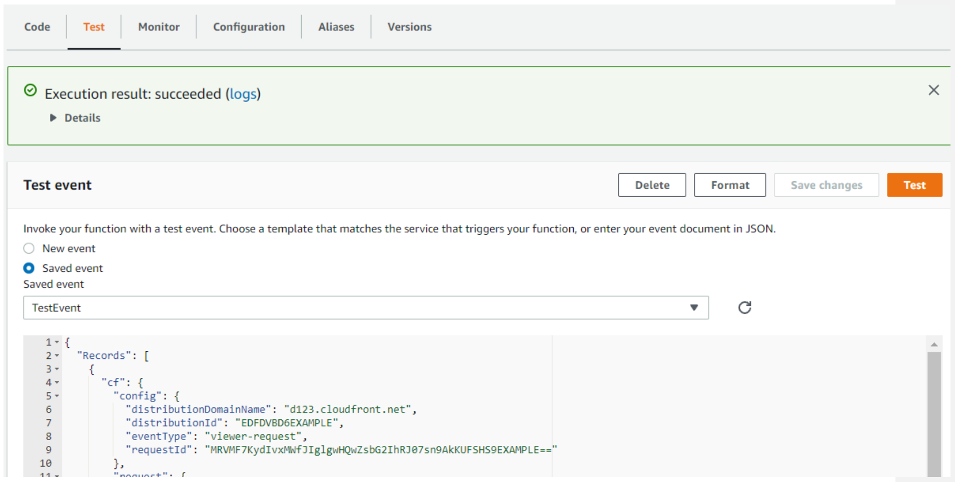
- Save the update



```
▼ Execution results
Test Event Name
TestEvent

Response
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}

Function Logs
START RequestId: 433a0ad7-3c50-431e-8753-4be26f0cbbe4 Version: $LATEST
END RequestId: 433a0ad7-3c50-431e-8753-4be26f0cbbe4
REPORT RequestId: 433a0ad7-3c50-431e-8753-4be26f0cbbe4  Duration: 18.93 ms  Billec

Request ID
433a0ad7-3c50-431e-8753-4be26f0cbbe4
```

- Test the function using the TestEvent configured by selecting **Test** button.

6. Look at the result of the execution. Notice that the output is a JSON representation of the HTTP 200 response that CloudFront will use to respond to the request. Now run our Test from the Test tab in the Lambda console.



Execution result: succeeded (logs)

Test event

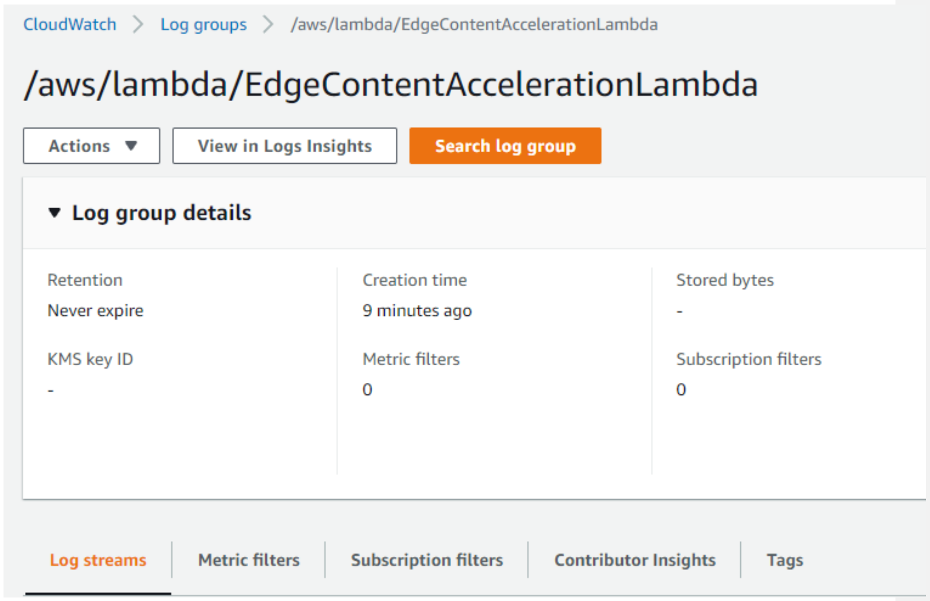
Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

( ) New event
(●) Saved event

Saved event: TestEvent

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d123.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-request",
          "requestId": "MRVMF7KydIvxMWfJIglgwHQwZsbG2IhRJ07sn9AkKUFSHS9EXAMPLE=="
        },
```

7. Select the logs hyperlink in the message that should indicate **Execution result: succeeded (logs)**

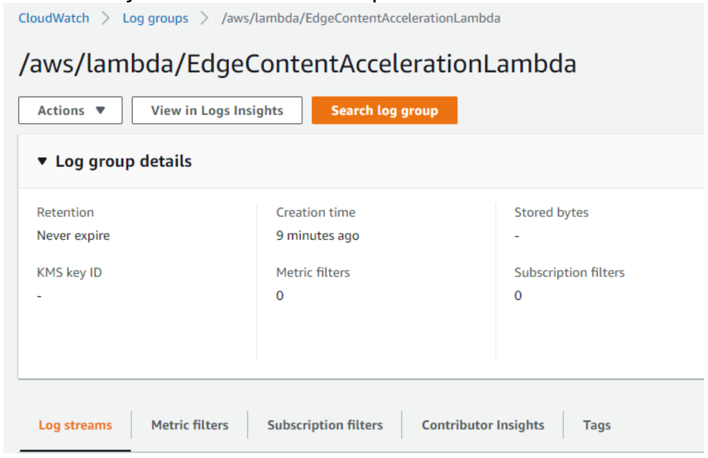

CloudWatch > Log groups > /aws/lambda/EdgeContentAccelerationLambda

/aws/lambda/EdgeContentAccelerationLambda

Log group details

Retention	Creation time	Stored bytes
Never expire	9 minutes ago	-
KMS key ID	Metric filters	Subscription filters
-	0	0

8. Notice that the logs link opens the CloudWatch console. CloudWatch has observability factilities that we'll explore in more detail in a later experiment.

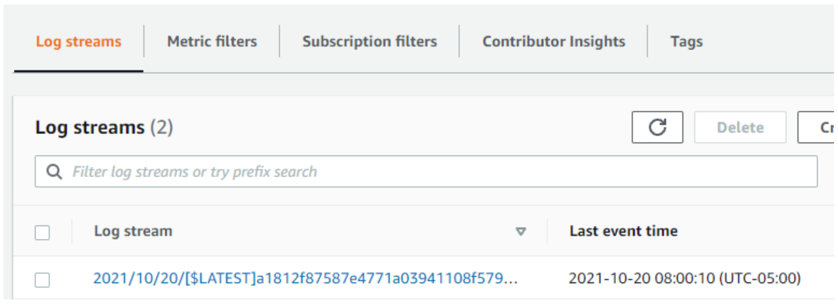

CloudWatch > Log groups > /aws/lambda/EdgeContentAccelerationLambda

/aws/lambda/EdgeContentAccelerationLambda

Log group details

Retention	Creation time	Stored bytes
Never expire	9 minutes ago	-
KMS key ID	Metric filters	Subscription filters
-	0	0

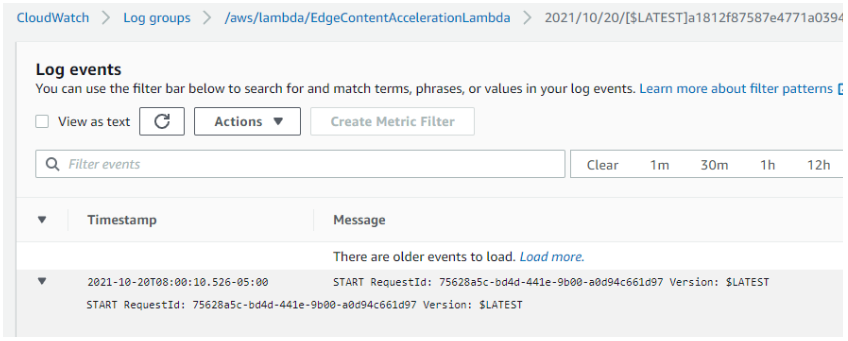
9. Select the last log stream to view the output of the console.log output from our Lambda Serverless FaaS.



Log streams (2)

Log stream	Last event time
2021/10/20/[$LATEST]a1812f87587e4771a03941108f579...	2021-10-20 08:00:10 (UTC-05:00)

10. Notice that we have the output noted from our Lambda that could be used as a monitoring/tracing/logging validation, security audit, or similar observability use cases. If you see the message There are older events to load. *Load more.* Select **Load more** to view the information.



Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Timestamp	Message
	There are older events to load. *Load more.*
2021-10-20T08:00:10.526-05:00	START RequestId: 75628a5c-bd4d-441e-9b00-a0d94c661d97 Version: $LATEST

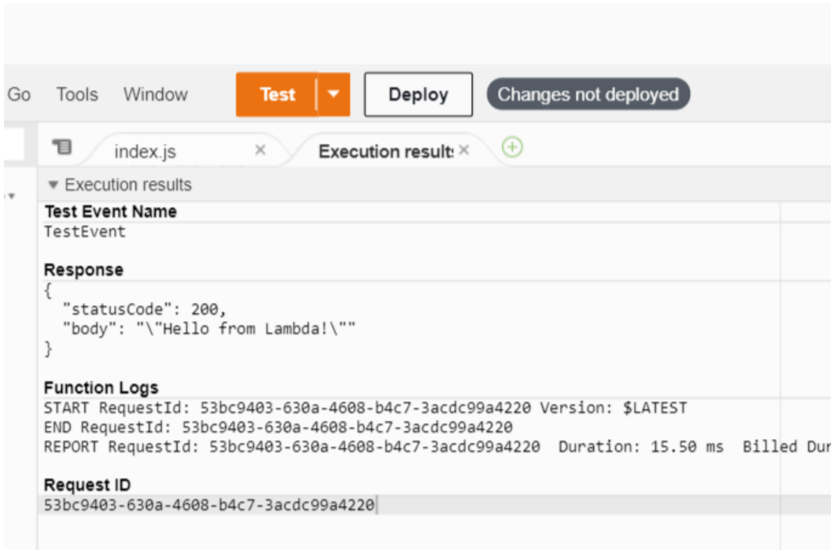
```
START RequestId: 75628a5c-bd4d-441e-9b00-a0d94c661d97 Version: $LATEST
```

11. In this case, the response is still missing the body. Return to the Lambda console browser tab. In the Log output section, notice that the test event that we configured in step 6 logged as the Request Event on the input of the function. This JSON represents attributes of the request received by CloudFront which can be read or modified. In this exercise we will read the headers and return the results in a pretty HTML table.

12. Replace the code in our Lambda with the following needed to generate the html body, or you can copy and paste from **EdgeContentAccelerationServerlessContent.txt** in our GitHub repo. You can use console.log to output to troubleshoot your code if you have issues with the copying and pasting that causes code issues.

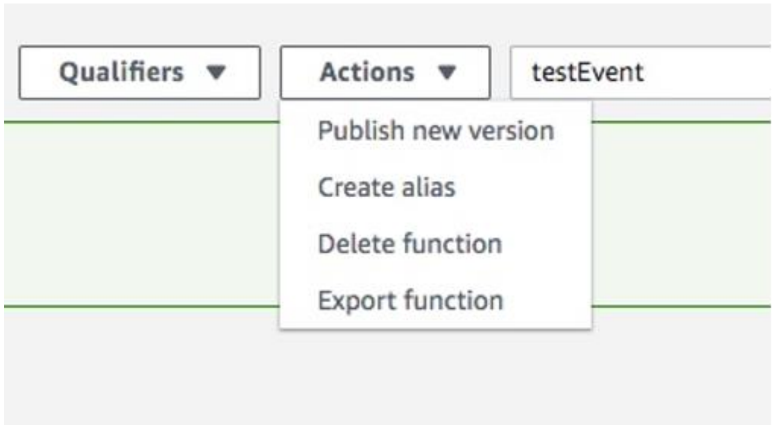
```
exports.handler= (event, context, callback) => {
    const requestHeaders = event.Records[0].cf.request.headers;

    var str = `<table border="1" width="100%">
                    <thead>

<tr><td><h1>Header</h2></td><td><h1>Value</h2></td></tr>
                    </thead>
                    <tbody>`;

    for (var key in requestHeaders) {
      if (requestHeaders.hasOwnProperty(key)) {
        str += "<tr><td>"+key + "</td><td>" + requestHeaders[key][0].value +
"</td></tr>";
      }
    }

    str+= "</tbody></table>";

    var htmlContent = `<html lang="en">
                  <body>
                    <table border="1" width="100%">
                    <thead>
                        <tr><td><h1>Lambda@Edge Lab</h1></td></tr>
                    </thead>
                    <tfoot>
                        <tr><td>Immersion Days - Edge Services - Module
3</td></tr>
                    </tfoot>
                    <tbody>
                        <tr><td>Response sent by API</td></tr>
                    </tbody>
                    <tbody>
                        <tr><td> ` + str + `</td></tr>
                    </tbody>
                    </table>
                  </body>
                </html>`;


    const response = {
        status: '200',
        statusDescription: 'OK',
        headers: {
            'cache-control': [{
                key: 'Cache-Control',
                value: 'max-age=100'
            }],
            'content-type': [{
                key: 'Content-Type',
                value: 'text/html'
            }],
            'content-encoding': [{
                key: 'Content-Encoding',
                value: 'UTF-8'
            }],
        },
        body: htmlContent,
    };

    callback(null, response);
}
```

13. Once you've completed testing of your function, publish and deploy the first version of the function.

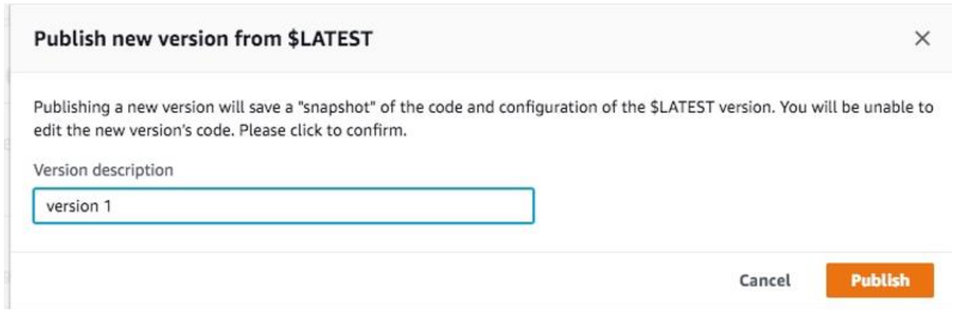


```
Test Event Name
TestEvent

Response
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}

Function Logs
START RequestId: 53bc9403-630a-4608-b4c7-3acdc99a4220 Version: $LATEST
END RequestId: 53bc9403-630a-4608-b4c7-3acdc99a4220
REPORT RequestId: 53bc9403-630a-4608-b4c7-3acdc99a4220  Duration: 15.50 ms  Billed Dur

Request ID
53bc9403-630a-4608-b4c7-3acdc99a4220
```

- Select Deploy to deploy your Lambda
- Select the Actions drop down and select publish new version.



- Specify a version description then click Publish.



Publish new version from $LATEST

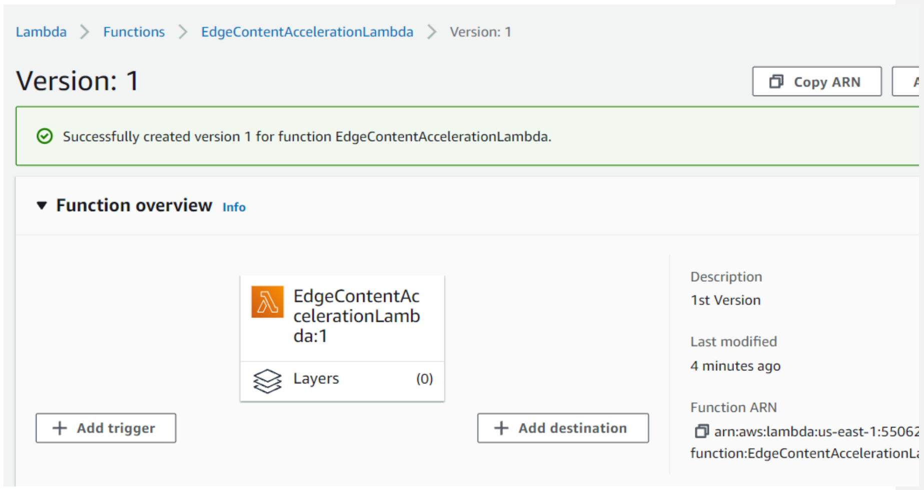
Publishing a new version will save a "snapshot" of the code and configuration of the $LATEST version. You will be unable to edit the new version's code. Please click to confirm.

Version description: version 1

Congratulations you now have a Lambda Function that can be used with CloudFront!

Deploy Lambda@Edge function to CloudFront

1. In the same lambda console, you now have Version 1 of the function deployed.



Lambda > Functions > EdgeContentAccelerationLambda > Version: 1
Version: 1
Copy ARN

Successfully created version 1 for function EdgeContentAccelerationLambda.

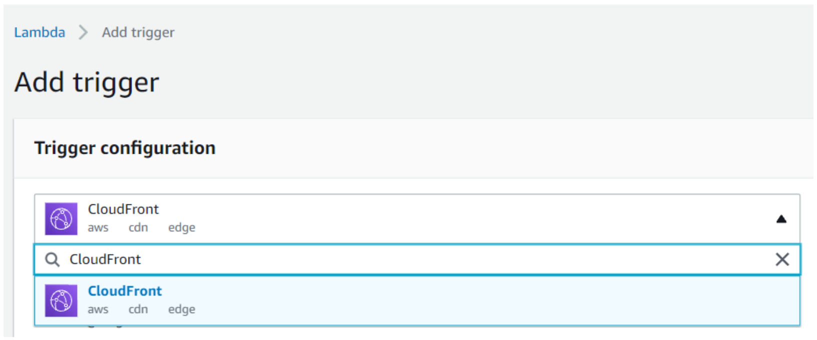
Function overview Info
EdgeContentAccelerationLambda:1
Layers (0)
Add trigger
Add destination

Description: 1st Version

Last modified: 4 minutes ago

Function ARN: arn:aws:lambda:us-east-1:55062 function:EdgeContentAccelerationL

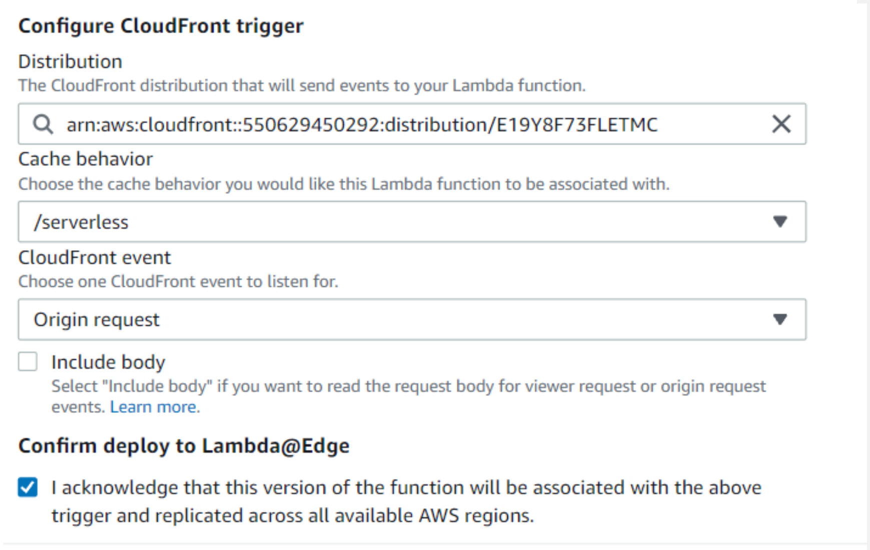
2. Click the Add trigger and select CloudFront.


Lambda > Add trigger

Add trigger

Trigger configuration

3. Update the trigger configuration as noted. This configures the trigger to use the CloudFront Distribution and Cache Behavior created earlier with the following settings:


Configure CloudFront trigger

Distribution
The CloudFront distribution that will send events to your Lambda function.
arn:aws:cloudfront::550629450292:distribution/E19Y8F73FLETMC

Cache behavior
Choose the cache behavior you would like this Lambda function to be associated with.
/serverless

CloudFront event
Choose one CloudFront event to listen for.
Origin request

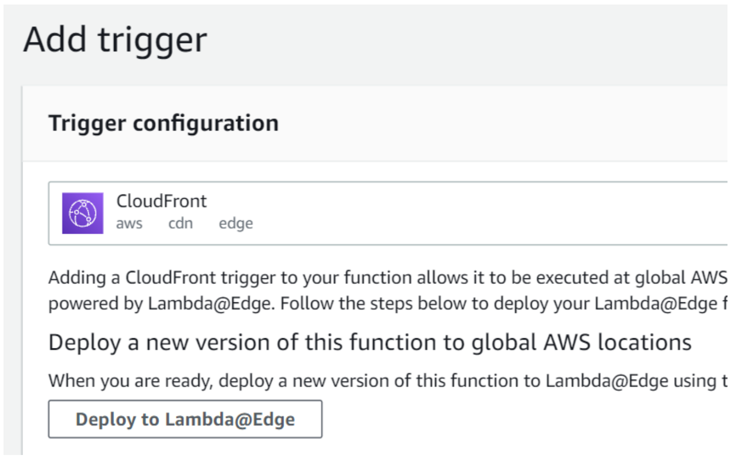
☐ Include body
Select "Include body" if you want to read the request body for viewer request or origin request events. Learn more.

Confirm deploy to Lambda@Edge

☑ I acknowledge that this version of the function will be associated with the above trigger and replicated across all available AWS regions.

- Distribution: <distributionID created from our Edge Content Acceleration should be pre-selected>
- Cache behavior: **/serverless**
- CloudFront event: **Origin request**
- Check the Confirm deploy to Lambda@Edge checkbox to “I acknowledge that on deploy a new version of this function will be published with the above trigger and replicated across all available AWS regions.”
- Select **Add**

Note: Check the triggers to ensure that it was created and associated correctly. If it does not appear. The click Add Trigger again and associate the version we created.



Add trigger

Trigger configuration

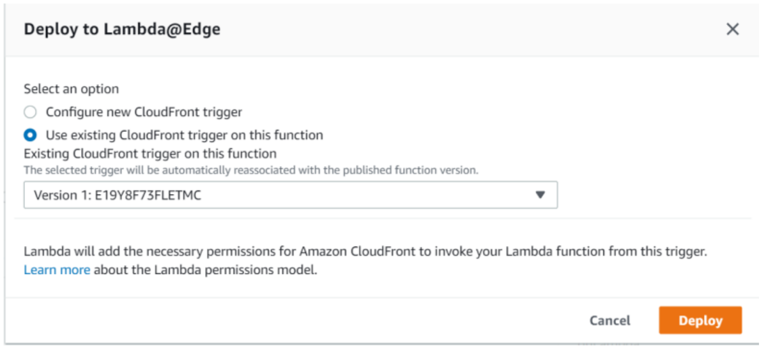
CloudFront
aws cdn edge

Adding a CloudFront trigger to your function allows it to be executed at global AWS
powered by Lambda@Edge. Follow the steps below to deploy your Lambda@Edge f

Deploy a new version of this function to global AWS locations

When you are ready, deploy a new version of this function to Lambda@Edge using t

4. Select **Deploy to Lambda@Edge**



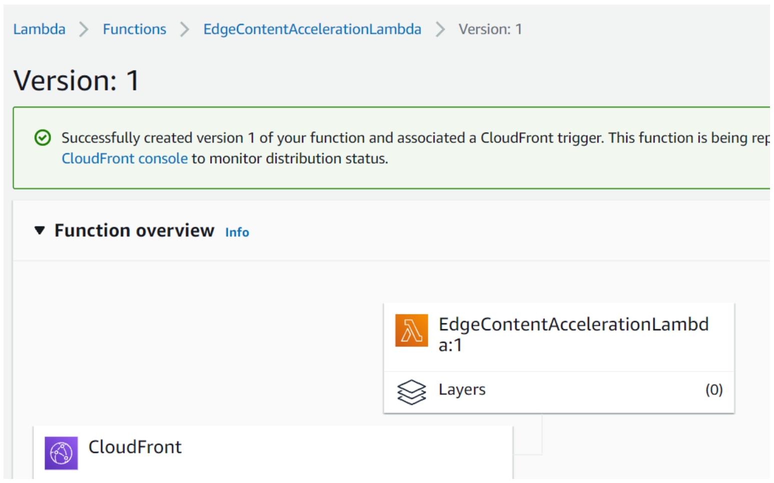
Deploy to Lambda@Edge

Select an option
( ) Configure new CloudFront trigger
(●) Use existing CloudFront trigger on this function

Existing CloudFront trigger on this function
The selected trigger will be automatically reassociated with the published function version.
Version 1: E19Y8F73FLETMC

Lambda will add the necessary permissions for Amazon CloudFront to invoke your Lambda function from this trigger.
Learn more about the Lambda permissions model.

5. Select to Use existing with the trigger we previously created and select **Deploy**.


Lambda > Functions > EdgeContentAccelerationLambda > Version: 1

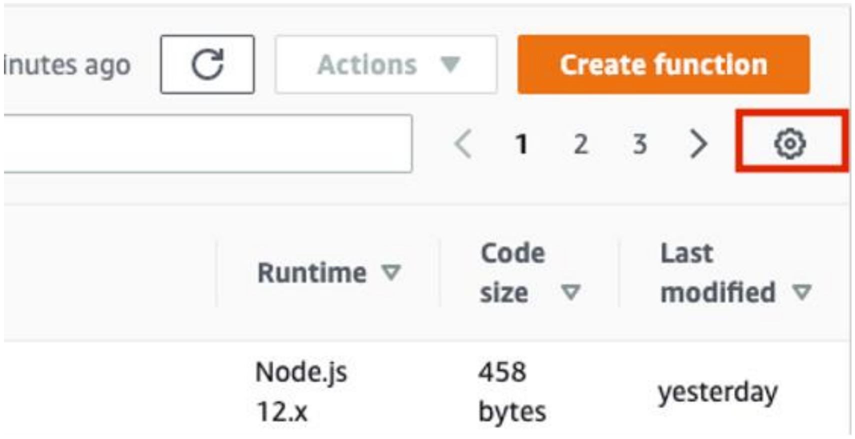
Version: 1

Successfully created version 1 of your function and associated a CloudFront trigger. This function is being rep
CloudFront console to monitor distribution status.

It will take approximately 5 minutes for a full deployment to your CloudFront distribution. In some cases, you may be able to begin testing in less than 5 minutes, depending on your location.

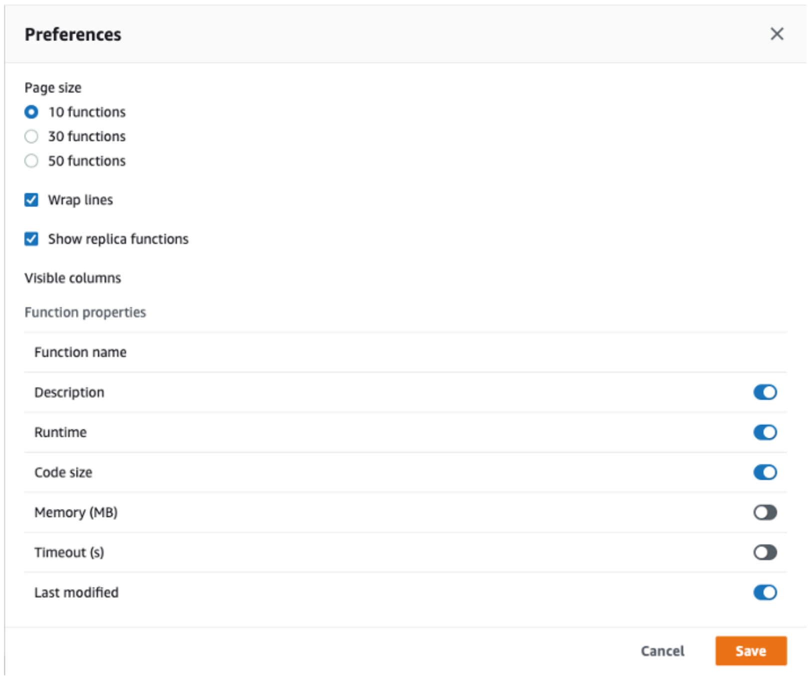
To view the replica functions:

- Return to the Lambda main console view to list your functions
- Select the gear icon on the top right to configure your preferences



Runtime	Code size	Last modified
Node.js 12.x	458 bytes	yesterday

6. Select the check box for “Show replica functions” and click Save.



Preferences

Page size

- (●) 10 functions
- ( ) 30 functions
- ( ) 50 functions

[✓] Wrap lines

[✓] Show replica functions

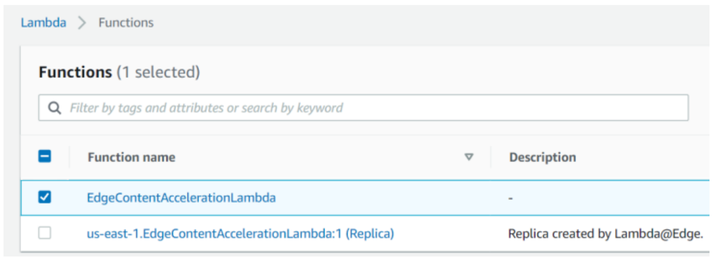
Visible columns

Function properties

Function name	
Description	on
Runtime	on
Code size	on
Memory (MB)	off
Timeout (s)	off
Last modified	on

Cancel Save

7. Now search for your function in the function list and you will find a Replica function in us-east-1 for the function you created. When you switch to other AWS regions, you will find that there is a replica function in all of the regions where CloudFront has a Regional Edge Cache. These are the functions that will be invoked when your CloudFront distribution executes Lambda@Edge.



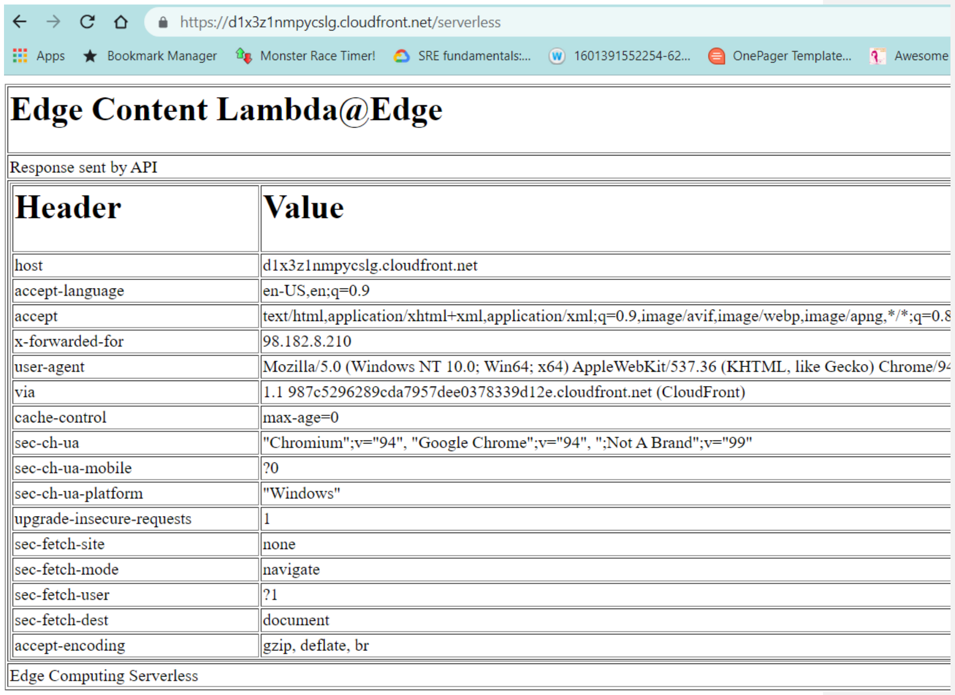
Lambda > Functions

Functions (1 selected)

Filter by tags and attributes or search by keyword

	Function name	Description
✓	EdgeContentAccelerationLambda	-
	us-east-1.EdgeContentAccelerationLambda:1 (Replica)	Replica created by Lambda@Edge.

8. Once your CloudFront distribution has completed deploying, test your CloudFront distribution by going to the Distribution on the browser with the serverless path. You've successfully deployed your Lambda@Edge function to dynamically generate a response.



Edge Content Lambda@Edge

Response sent by API

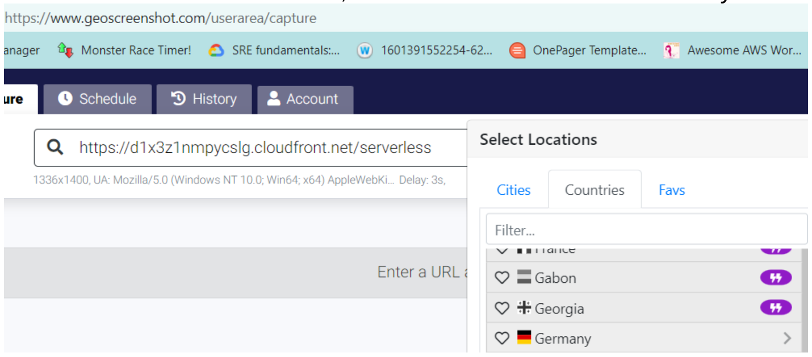
Header	Value
host	d1x3z1nmpycslg.cloudfront.net
accept-language	en-US,en;q=0.9
accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
x-forwarded-for	98.182.8.210
user-agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94
via	1.1 987c5296289cda7957dee0378339d12e.cloudfront.net (CloudFront)
cache-control	max-age=0
sec-ch-ua	"Chromium";v="94", "Google Chrome";v="94", ";Not A Brand";v="99"
sec-ch-ua-mobile	?0
sec-ch-ua-platform	"Windows"
upgrade-insecure-requests	1
sec-fetch-site	none
sec-fetch-mode	navigate
sec-fetch-user	?1
sec-fetch-dest	document
accept-encoding	gzip, deflate, br

Edge Computing Serverless

Metrics and Logs
In this section, you will learn where to view Lambda@Edge metrics and CloudWatch logs for your function.

1. To generate traffic from different geographies, use a free website such as https://www.geoscreenshot.com to send a request to your distribution to hit the serverless path (i.e. https://d123.cloudfront.net/serverless) from different locations. Submit the request several times to generate traffic to CloudFront from different regions. Notice that many of the countries are

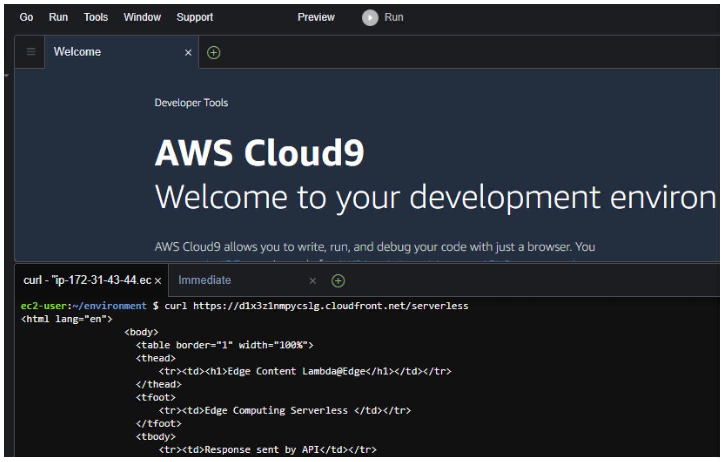
not available in the free version, so choose one that is like Germany.



https://d1x3z1nmpycslg.cloudfront.net/serverless

1336x1400, UA: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKi... Delay: 3s,

2. Alternatively you can do what we did previously which is spin up a Cloud9 instance in a different region like Ireland and use curl.



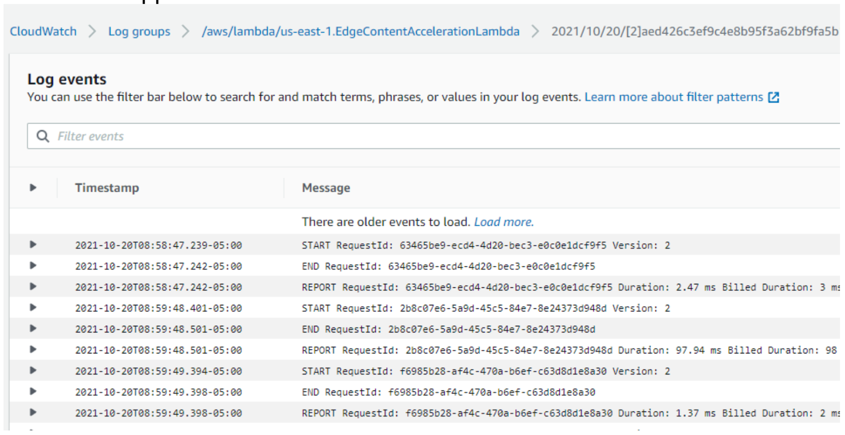
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You

```
ec2-user:~/environment $ curl https://d1x3z1nmpycslg.cloudfront.net/serverless
<html lang="en">
                  <body>
                    <table border="1" width="100%">
                    <thead>
                        <tr><td><h1>Edge Content Lambda@Edge</h1></td></tr>
                    </thead>
                    <tfoot>
                        <tr><td>Edge Computing Serverless </td></tr>
                    </tfoot>
                    <tbody>
                        <tr><td>Response sent by API</td></tr>
```

3. Go to the AWS Lambda console and select an AWS region near one of the geographies that you selected to submit the request from from step 1.

4. Ensure that you have your preferences configured to “show replica functions” as configured in the previous sections. Then select the replica function for that region and view the “Monitoring” tab for that function.

5. To view logs from your function execution, select “View logs in CloudWatch”. Here you can view any logs that are generated from your function invocations. Note - any log lines generated from your code will also appear here.



CloudWatch > Log groups > /aws/lambda/us-east-1.EdgeContentAccelerationLambda > 2021/10/20/[2]aed426c3ef9c4e8b95f3a62bf9fa5b

Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Filter events

```
Timestamp                        Message
                                 There are older events to load. Load more.
2021-10-20T08:58:47.239-05:00    START RequestId: 63465be9-ecd4-4d20-bec3-e0c0e1dcf9f5 Version: 2
2021-10-20T08:58:47.242-05:00    END RequestId: 63465be9-ecd4-4d20-bec3-e0c0e1dcf9f5
2021-10-20T08:58:47.242-05:00    REPORT RequestId: 63465be9-ecd4-4d20-bec3-e0c0e1dcf9f5 Duration: 2.47 ms Billed Duration: 3 ms
2021-10-20T08:59:48.401-05:00    START RequestId: 2b8c07e6-5a9d-45c5-84e7-8e24373d948d Version: 2
2021-10-20T08:59:48.501-05:00    END RequestId: 2b8c07e6-5a9d-45c5-84e7-8e24373d948d
2021-10-20T08:59:48.501-05:00    REPORT RequestId: 2b8c07e6-5a9d-45c5-84e7-8e24373d948d Duration: 97.94 ms Billed Duration: 98
2021-10-20T08:59:49.394-05:00    START RequestId: f6985b28-af4c-470a-b6ef-c63d8d1e8a30 Version: 2
2021-10-20T08:59:49.398-05:00    END RequestId: f6985b28-af4c-470a-b6ef-c63d8d1e8a30
2021-10-20T08:59:49.398-05:00    REPORT RequestId: f6985b28-af4c-470a-b6ef-c63d8d1e8a30 Duration: 1.37 ms Billed Duration: 2 ms
```

NOTE: Metrics and Logs are REGIONAL. To view the appropriate logs, switch between different AWS regions to view metrics and logs for function invocations in each region. In our case the accounts we use for these experiments are region constrained so we’ll see details in North American Regions.