Using Global Settings to Configure the Implicit API

In this lesson, you will learn how to modify global settings to configure API Gateway.

WE'LL COVER THE FOLLOWING

With the Lambda Proxy integration, API Gateway just packages all the request properties and forwards them to Lambda. However, if you look at one of the request logs carefully (see figure here), you'll notice some headers that do not look like they are coming from a browser. For example, `CloudFront-Viewer-Country` shows the ISO country code where the request originated. This is because the request did not go directly from a browser to API Gateway. Instead, it first went to the AWS global content distribution system called CloudFront, which put in some additional headers.

API Gateway configurations

SAM can set up two types of API Gateway configuration:

1. edge-optimised
2. regional

Edge-optimised APIs

Edge optimised APIs are served from a specific region, but the clients connect to the nearest AWS presence point. So far, you've been deploying an API to the `us-east-1` region in North Virginia, USA. With edge-optimised APIs, a user from Germany will likely connect to an AWS CloudFront endpoint in Frankfurt first, then the request will pass to the API through the internal AWS links, not through the public internet. AWS has fast links between regions and using the nearest point of presence generally improves connection latency for global users. But this also means that someone trying to access the API from North Virginia will go through an unnecessary intermediary.

Regional APIs

Regional APIs do not set up an intermediary connection using CloudFront. If you deployed a regional API to `us-east-1`, a user from Germany would connect to the AWS endpoint in North Virginia through the public internet. That would likely make global connections worse, but local connections would be slightly faster.

Although CloudFront adds additional cost to each request, removing it may not actually reduce the overall price significantly. Both API Gateway and CloudFront charge for the number of requests and for data transfer. Data transfer through CloudFront is slightly cheaper than using API Gateway, and AWS does not charge for transfer between its own services. With both services in the pipeline, the requests are slightly more expensive, but data transfers are slightly cheaper. In conclusion, don't deploy regional endpoints expecting to save money, but do it if you want to remove an intermediary for connections in a specific territory.

By default, SAM assumes that you want to take advantage of edge-optimised APIs, so your users globally get faster responses if you deploy it to just one region. If you plan to deploy entire stacks in different AWS regions, so you can serve users locally, tell SAM to use regional APIs.

To change the endpoint type of the API that SAM creates automatically, create an `Api` property in the `Globals` section. Then add a sub-property called `EndpointConfiguration` inside, and set it to `REGIONAL` (add lines 4-5 from the following listing to your template).

```
Globals:
  Function:
    Timeout: 3
  Api:
    EndpointConfiguration: REGIONAL
```

Line 7 to Line 11 of code/ch6/template-regional.yaml

Now you'll build, package, and deploy the stack again, then trigger another

Now you'll build, package, and deploy the stack again, then trigger another request and look at the logs. CloudFront headers should no longer appear

because the API is now regional).

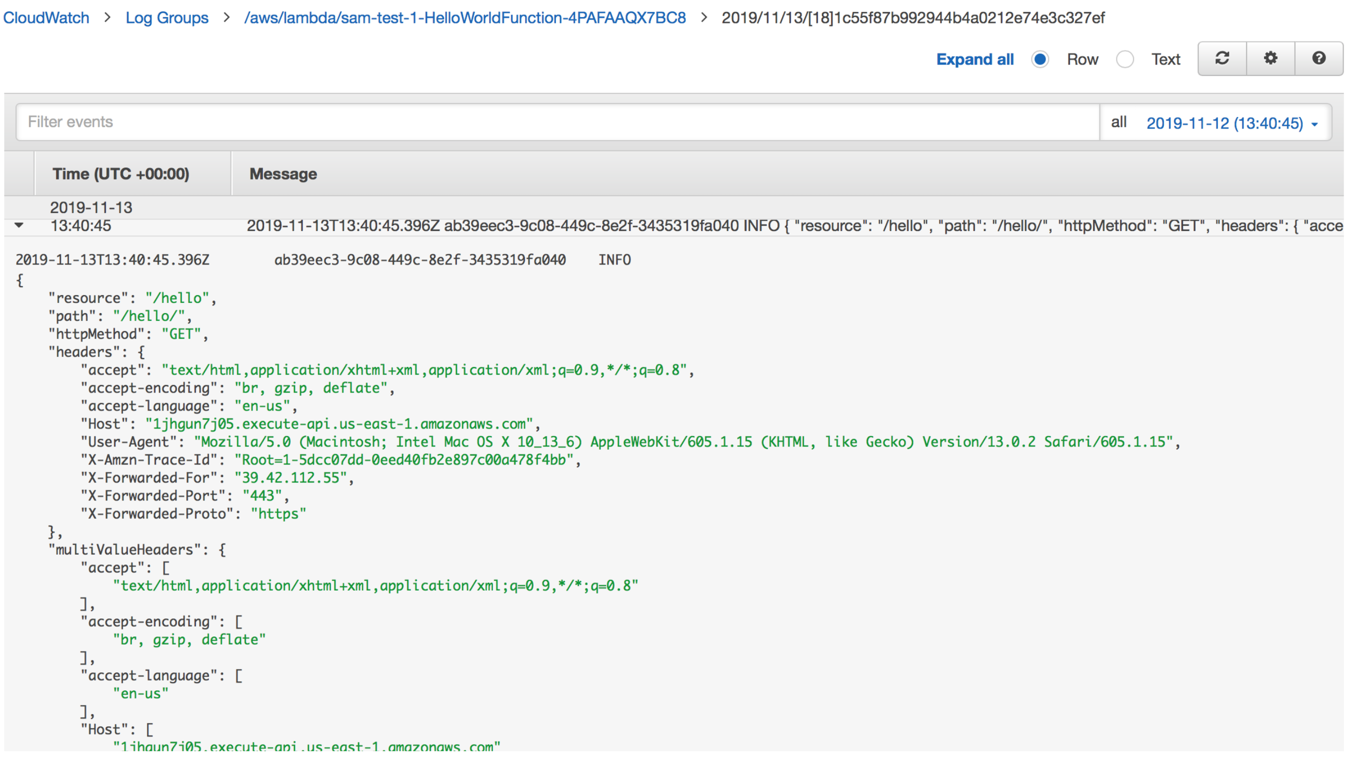
`template-regional.yaml` is included in the widget below and the application is built using that file.

Environment Variables

Key:	Value:
AWS_ACCESS_KEY_ID	Not Specified...
AWS_SECRET_ACCE...	Not Specified...
BUCKET_NAME	Not Specified...
AWS_REGION	Not Specified...

```
{
  "body": "{\"message\": \"hello world\"}",
  "resource": "/{proxy+}",
  "path": "/path/to/resource",
  "httpMethod": "POST",
  "isBase64Encoded": false,
  "queryStringParameters": {
    "foo": "bar"
  },
  "pathParameters": {
    "proxy": "/path/to/resource"
  },
  "stageVariables": {
    "baz": "qux"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate, sdch",
    "Accept-Language": "en-US,en;q=0.8",
    "Cache-Control": "max-age=0",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Desktop-Viewer": "true",
    "CloudFront-Is-Mobile-Viewer": "false",
    "CloudFront-Is-SmartTV-Viewer": "false",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Viewer-Country": "US",
    "Host": "1234567890.execute-api.us-east-1.amazonaws.com",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Custom User Agent String",
    "Via": "1.1 08f323deadbeefa7af34d5feb414ce27.cloudfront.net (CloudFront)",
    "X-Amz-Cf-Id": "cDehVQoZnx43VYQb9j2-nvCh-9z396Uhbp027Y2JvkCPNLmGJHqlaA==",
    "X-Forwarded-For": "127.0.0.1, 127.0.0.2",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"
  },
  "requestContext": {
    "accountId": "123456789012",
    "resourceId": "123456",
    "stage": "prod",
    "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
    "requestTime": "09/Apr/2015:12:34:56 +0000",
    "requestTimeEpoch": 1428582896000,

    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "accessKey": null,
      "sourceIp": "127.0.0.1",
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,
      "userAgent": "Custom User Agent String",
      "user": null
    },
    "path": "/prod/path/to/resource",
    "resourcePath": "/{proxy+}",
    "httpMethod": "POST",
    "apiId": "1234567890",
    "protocol": "HTTP/1.1"
  }
}
```


CloudWatch > Log Groups > /aws/lambda/sam-test-1-HelloWorldFunction-4PAFAAQX7BC8 > 2019/11/13/[18]1c55f87b992944b4a0212e74e3c327ef

Expand all [✓ Row] Text

Filter events — all 2019-11-12 (13:40:45)

Time (UTC +00:00)	Message
2019-11-13 13:40:45	2019-11-13T13:40:45.396Z ab39eec3-9c08-449c-8e2f-3435319fa040 INFO { "resource": "/hello", "path": "/hello/", "httpMethod": "GET", "headers": { "acce

```
2019-11-13T13:40:45.396Z        ab39eec3-9c08-449c-8e2f-3435319fa040    INFO
{
    "resource": "/hello",
    "path": "/hello/",
    "httpMethod": "GET",
    "headers": {
        "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
        "accept-encoding": "br, gzip, deflate",
        "accept-language": "en-us",
        "Host": "1jhgun7j05.execute-api.us-east-1.amazonaws.com",
        "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.2 Safari/605.1.15",
        "X-Amzn-Trace-Id": "Root=1-5dcc07dd-0eed40fb2e897c00a478f4bb",
        "X-Forwarded-For": "39.42.112.55",
        "X-Forwarded-Port": "443",
        "X-Forwarded-Proto": "https"
    },
    "multiValueHeaders": {
        "accept": [
            "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
        ],
        "accept-encoding": [
            "br, gzip, deflate"
        ],
        "accept-language": [
            "en-us"
        ],
        "Host": [
            "1jhgun7j05.execute-api.us-east-1.amazonaws.com"
```

Regional APIs do not pass through the content distribution network, so there are no more CloudFront headers.

For a full list of properties, you can configure this way, check out the Globals Section documentation page for AWS SAM.

Building a different template

Notice that the title of the previous listing has a different template file name (`template-regional.yaml`). If you are following the tutorial, just modify your main template file with the proposed change. If you want to

use the source code package from https://runningserverless.com, you'll find several templates in the directory for this chapter – one with the regional configuration and one without. So far, you've always used `sam build` to prepare the default `template.yaml` file for packaging. You can use the `-t <FILE NAME>` option to tell SAM to use a specific template file. To build directly from the regional template example, you should use the following command:

```
sam build -t template-regional.yaml
```

The commands to package and deploy stay the same as before.

Hopefully you are clear about everything you have seen so far. See you in the next lesson!