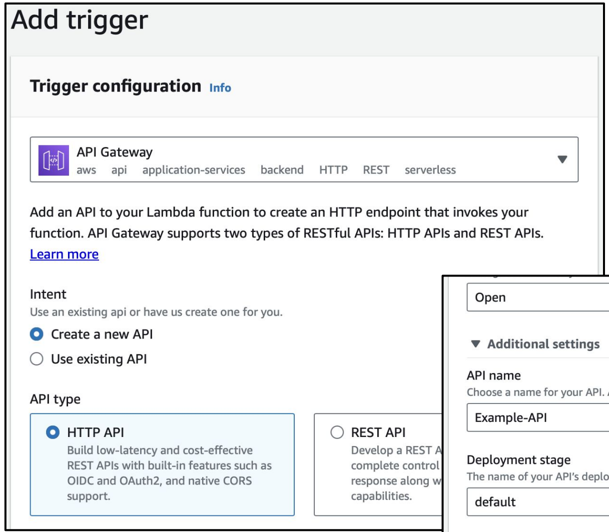
API Gateway & EventBridge (CloudWatch Events)

API Gateway

- Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services.
- In our case we used API Gateway to create web APIs with HTTP endpoints for our Lambda functions.
- There are a few reasons why we used API Gateway to trigger our functions:
    - HTTP-Based Triggers: It allowed us to get HTTP endpoints that can be accessed in our Forage dashboard.
    - CORS Support: Provides Cross-Origin Resource Sharing (CORS) support, which enabled us to access our APIs from different domains and prevent potential security vulnerabilities.
    - Easy Scaling: Amazon API Gateway will automatically scale to handle the amount of traffic our APIs receive. This ensures that our backend Lambda functions are not overwhelmed during traffic spikes.

Process of Adding an API Gateway Trigger



Add trigger

Trigger configuration Info

API Gateway
aws api application-services backend HTTP REST serverless

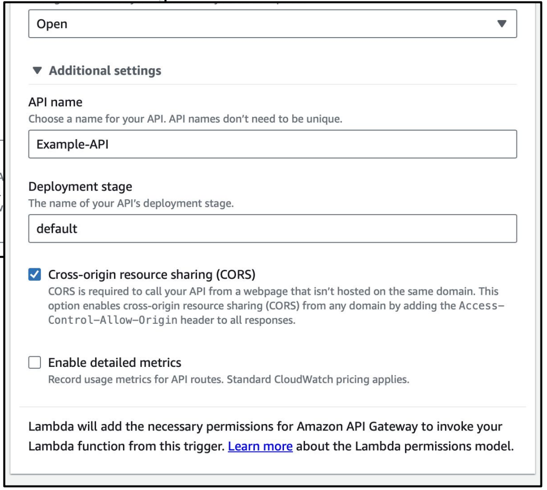
Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs.
Learn more

Intent
Use an existing api or have us create one for you.
[✓ Create a new API]
Use existing API

API type

[✓ HTTP API]
Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

REST API
Develop a REST A
complete control
response along w
capabilities.



Open

Additional settings

API name
Choose a name for your API. API names don't need to be unique.
Example-API

Deployment stage
The name of your API's deployment stage.
default

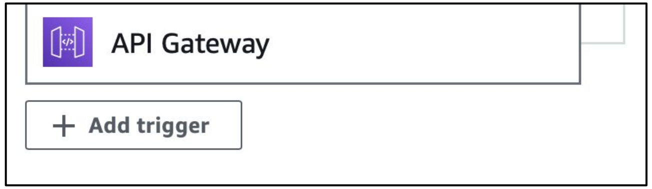
[✓ Cross-origin resource sharing (CORS)]
CORS is required to call your API from a webpage that isn't hosted on the same domain. This option enables cross-origin resource sharing (CORS) from any domain by adding the `Access-Control-Allow-Origin` header to all responses.

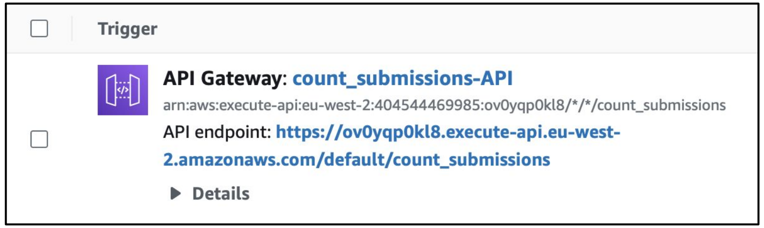
Enable detailed metrics
Record usage metrics for API routes. Standard CloudWatch pricing applies.

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

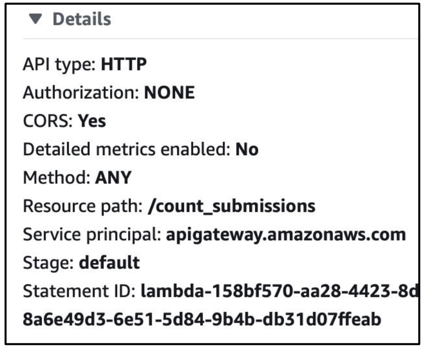
- After completing the code for each of our Lambda functions and testing these, we added a trigger.
- A trigger is a service or resource that invokes your function.
- We configured the trigger to be an API Gateway and chose the HTTP API type in order to create our HTTP endpoints.
- In the additional settings, we chose CORS as this is required in order for us to call our API from Forage, given that it is not hosted on the same domain.

Process of using API Gateway




Trigger

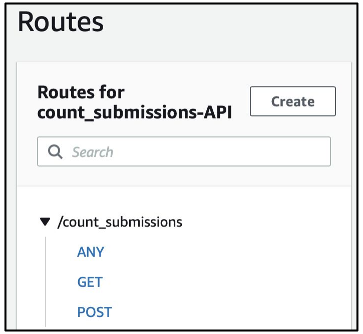
API Gateway: count_submissions-API
arn:aws:execute-api:eu-west-2:404544469985:ov0yqp0kl8/*/*/count_submissions
API endpoint: https://ov0yqp0kl8.execute-api.eu-west-2.amazonaws.com/default/count_submissions
▶ Details


▼ Details

API type: **HTTP**
Authorization: **NONE**
CORS: **Yes**
Detailed metrics enabled: **No**
Method: **ANY**
Resource path: **/count_submissions**
Service principal: **apigateway.amazonaws.com**
Stage: **default**
Statement ID: **lambda-158bf570-aa28-4423-8d**
8a6e49d3-6e51-5d84-9b4b-db31d07ffeab

- Once the API endpoints were successfully created, this could be accessed by checking our trigger list and clicking the API endpoint link.
- If you press the details toggle, it breaks down the details of the particular trigger.
- All of our API triggers had an API type HTTP and the HTTP request chosen was ANY.
- The resource path refers to our lambda function, so an ANY request is being sent to that specific function.

Process of using API Gateway

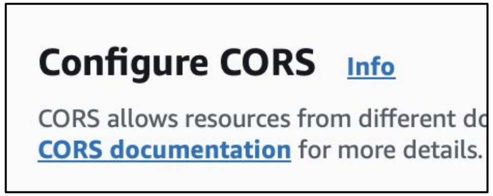


Routes

Routes for count_submissions-API

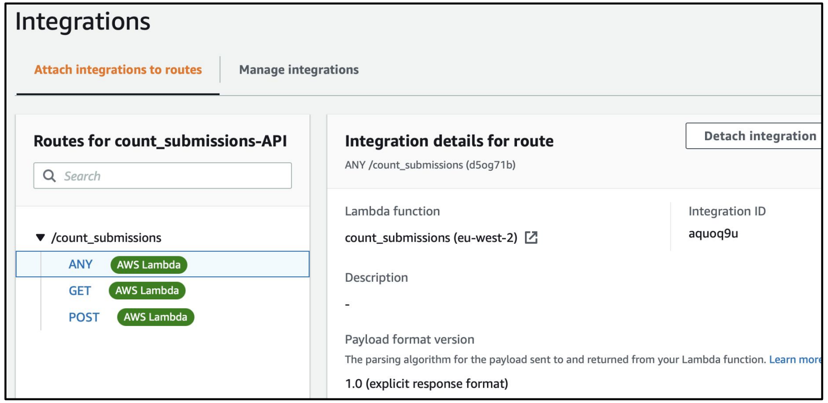
/count_submissions

- ANY
- GET
- POST



Configure CORS Info

CORS allows resources from different do
CORS documentation for more details.



Integrations

Routes for count_submissions-API

/count_submissions

- ANY AWS Lambda
- GET AWS Lambda
- POST AWS Lambda

Integration details for route

ANY /count_submissions (d5og71b)

Lambda function

count_submissions (eu-west-2)

Integration ID

aquoq9u

Description

-

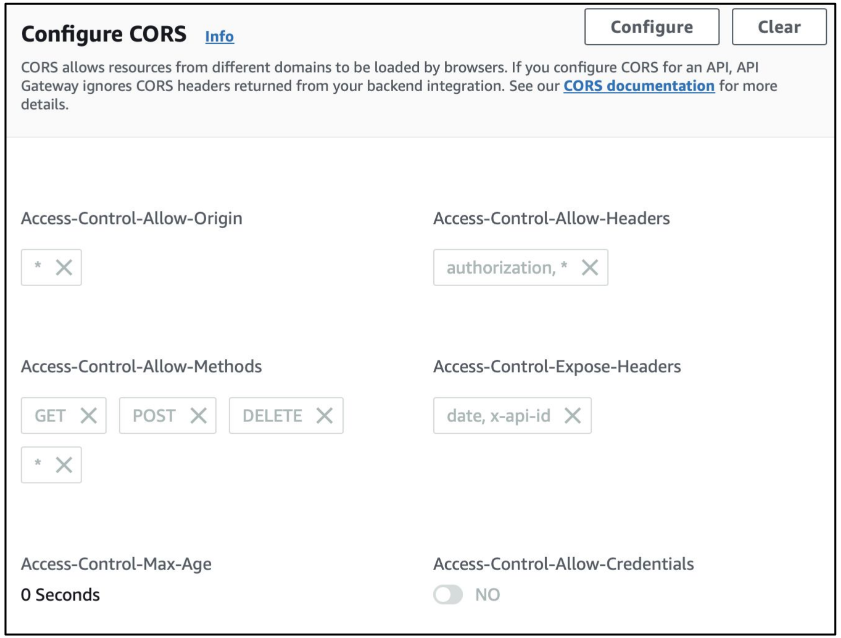
Payload format version

The parsing algorithm for the payload sent to and returned from your Lambda function. Learn more

1.0 (explicit response format)

- The next step was to add routes and integrations to our API.
- Routes direct incoming API requests to backend resources. Routes consist of two parts: an HTTP method and a resource path—for example, GET /count_submissions.
- You can define specific HTTP methods for your route. Or, you can use the ANY method to match all methods that you haven't defined for a resource.
- As shown alongside our existing ANY method, we added a GET and POST route and integrated these to the specific Lambda function.
- We also had to configure CORS.

Cross-Origin Resource Sharing (CORS)



Configure CORS Info

CORS allows resources from different domains to be loaded by browsers. If you configure CORS for an API, API Gateway ignores CORS headers returned from your backend integration. See our CORS documentation for more details.

Access-Control-Allow-Origin: *

Access-Control-Allow-Headers: authorization, *

Access-Control-Allow-Methods: GET, POST, DELETE, *

Access-Control-Expose-Headers: date, x-api-id

Access-Control-Max-Age: 0 Seconds

Access-Control-Allow-Credentials: NO

- CORS is a security feature implemented by web browsers to control how web pages from one domain (origin) can request and access resources from another domain (origin).
- Our Forage dashboard is hosted on herokuapp.com and we wanted to make API requests to our API endpoint that is hosted on AWS. This wasn't possible because browsers implement a security policy called the 'same-origin policy', which prevents web pages from making requests to a different origin than the one that served the web page. So we implemented CORS.
- CORS allows us to relax the same-origin policy and define which origins are allowed to access resources on our API Gateway.

CORS (extra for previous slide)

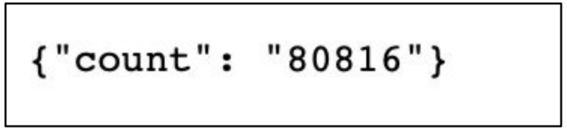
Why cors?

- Without cors, our dashboard wouldn’t be able to make requests to our API gateway, and the browser would block those requests due to security restrictions.

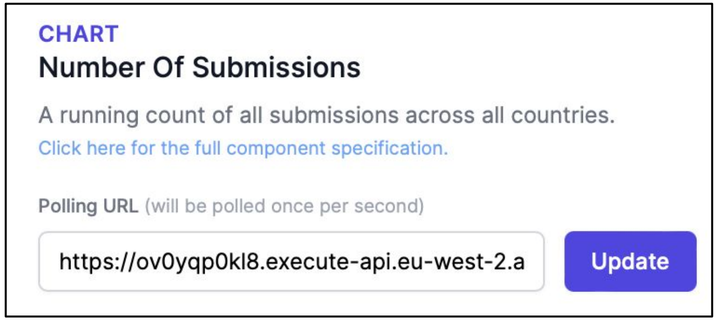
Setting up CORS in API Gateway:

- To enable CORS for each of our API endpoints, we needed to configure the necessary headers.
    - For the Access-Control-Allow-Origin header, we put * which means it will allow all origins.
    - For the Access-Control-Allow-Header header, we put ‘authorization, *’ which means it will allow the authorization header as well as any other header.
    - For the Access-Control-Allow-Methods header, we put some HTTP methods that are allowed to access the endpoint, * means all methods will be allowed.
    - For the Access-Control-Expose-Headers header, we put ‘date,x-api-id’ which means it will expose the headers Date and x-api-id.
- Overall, CORS is essential to enable communication between the different domains we have, whilst also maintaining a level of security.

HTTP endpoint example



```
{"count": "80816"}
```



CHART

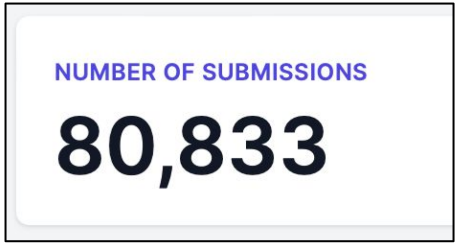
Number Of Submissions

A running count of all submissions across all countries.

Click here for the full component specification.

Polling URL (will be polled once per second)

https://ov0yqp0kl8.execute-api.eu-west-2.a



NUMBER OF SUBMISSIONS

80,833

- Here is the API endpoint for one of our charts: https://ov0yqp0kl8.execute-api.eu-west-2.amazonaws.com/default/count_submissions
- When an HTTP request is made to the API Gateway endpoint, it triggers the configured Lambda function. The Lambda function then executes and performs the desired operations.
- Opening this link in our browser will give the result in the JSON format shown above.
- When this URL is inputted into the polling chart and updated, we get the number of submissions as a number, and the data is up-to-the-second.

EventBridge (CloudWatch Events)

- EventBridge is a serverless service that uses events to connect application components together. It allows you to create a central hub for routing and processing events from various sources.

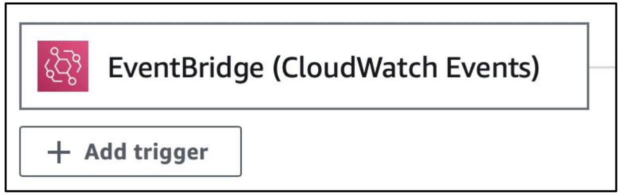
How does it work?

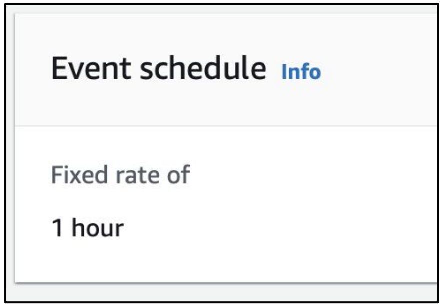
- It uses the concept of events and rules. Events represent changes or notifications that occur within AWS services or applications. Rules define the conditions under which events should trigger specific actions, so in our case invoking the Lambda function.

Using EventBridge to trigger a Lambda function:

- To schedule Lambda function to run every hour using EventBridge, you need to set up an event rule that specifies the schedule and target the Lambda function.

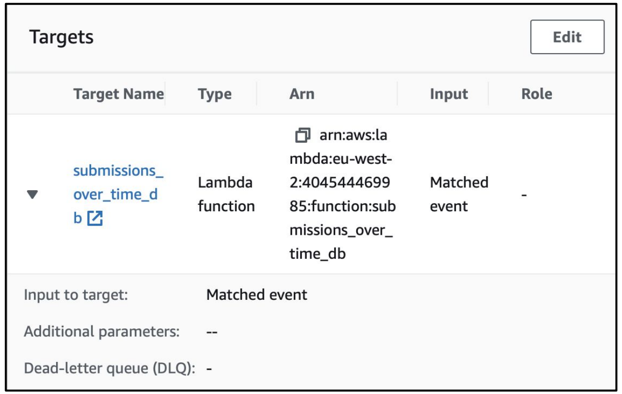
Process of using EventBridge


EventBridge (CloudWatch Events)


Event schedule

Fixed rate of

1 hour

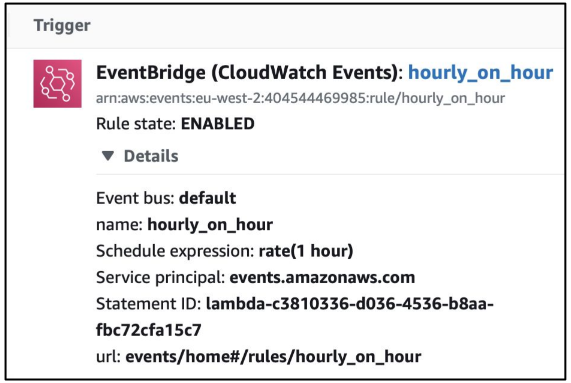

Targets

Target Name	Type	Arn	Input	Role
submissions_over_time_db	Lambda function	arn:aws:lambda:eu-west-2:404544469985:function:submissions_over_time_db	Matched event	-

Input to target: Matched event

Additional parameters: --

Dead-letter queue (DLQ): -


Trigger

EventBridge (CloudWatch Events): hourly_on_hour
arn:aws:events:eu-west-2:404544469985:rule/hourly_on_hour
Rule state: **ENABLED**

▼ Details

Event bus: **default**
name: **hourly_on_hour**
Schedule expression: **rate(1 hour)**
Service principal: **events.amazonaws.com**
Statement ID: **lambda-c3810336-d036-4536-b8aa-fbc72cfa15c7**
url: **events/home#/rules/hourly_on_hour**

- Once our Lambda function was created and tested, we added an EventBridge trigger to execute the code every hour.
- We chose a schedule for our event rule, which was “rate(1 hour)”, indicating that the trigger will execute the code in our Lambda function every hour.
- We specified our ‘Submissions over time’ Lambda function as the target.

Advantages of Using EventBridge

- The main reason we used EventBridge was because we needed a Lambda function that would be triggered periodically and send an hour submission count to one of our databases.
- There are a few reasons why EvenBridge is good for scheduled execution:
    - Serverless: We don’t need to manage the infrastructure at all, EventBridge and Lambda handle all of that.
    - Precise Scheduling: It is easy to set-up schedules using cron expressions “rate(1 hour)”.
    - Event-Driven Architecture: It is easy to integrate other AWS services and custom applications with EventBridge.
- Overall, by using EventBridge to trigger a Lambda function on a schedule, you can efficiently automate tasks that need to run periodically, such as data processing.