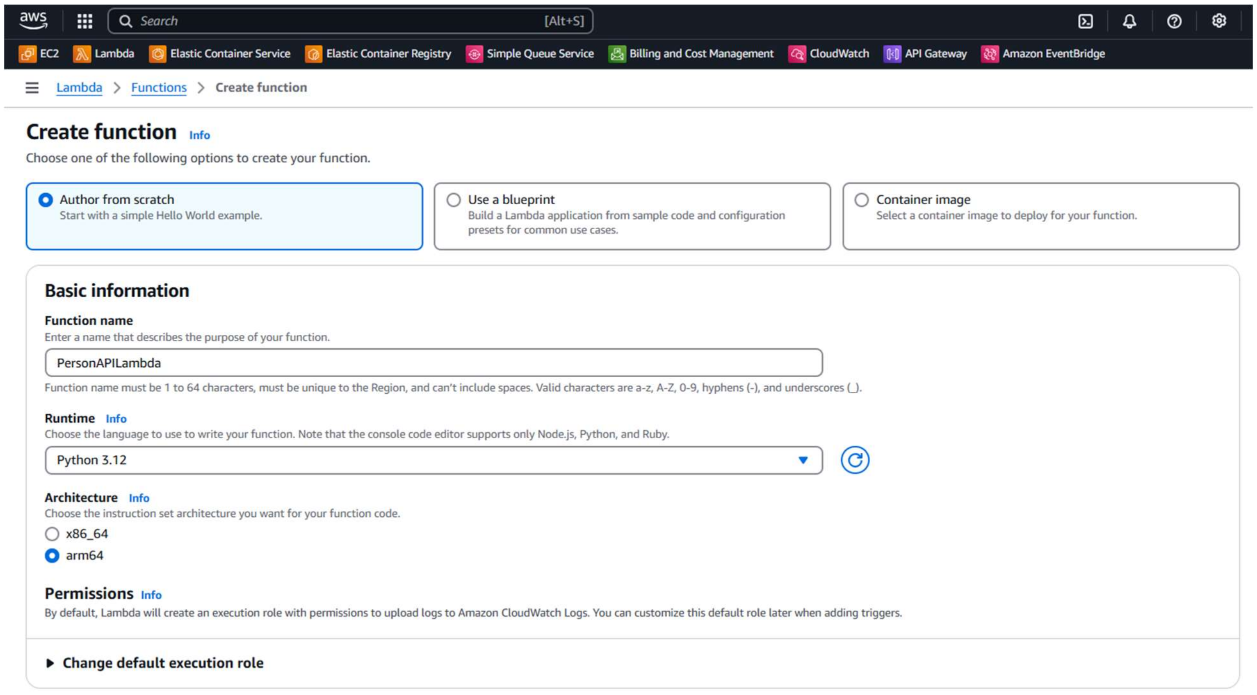
Report on AWS Lambda and HTTP API Gateway Implementation

Objective

This project involves setting up an **AWS Lambda function** and an **HTTP API Gateway** to handle GET requests. The Lambda function extracts the `personId` parameter from the query string and prints it.

Step 1: Creating the AWS Lambda Function

1. Navigate to the **AWS Lambda** console.
2. Click on **Create function**.
3. Select **Author from scratch** and enter the following details:
   - **Function Name:** `PersonAPILambda`
   - **Runtime:** Python 3.12
4. Click **Create function**.


aws
Search [Alt+S]
EC2
Lambda
Elastic Container Service
Elastic Container Registry
Simple Queue Service
Billing and Cost Management
CloudWatch
API Gateway
Amazon EventBridge
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 image — Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
PersonAPILambda
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12

Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64 [✓ arm64]

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

Step 2: Writing the Python Code

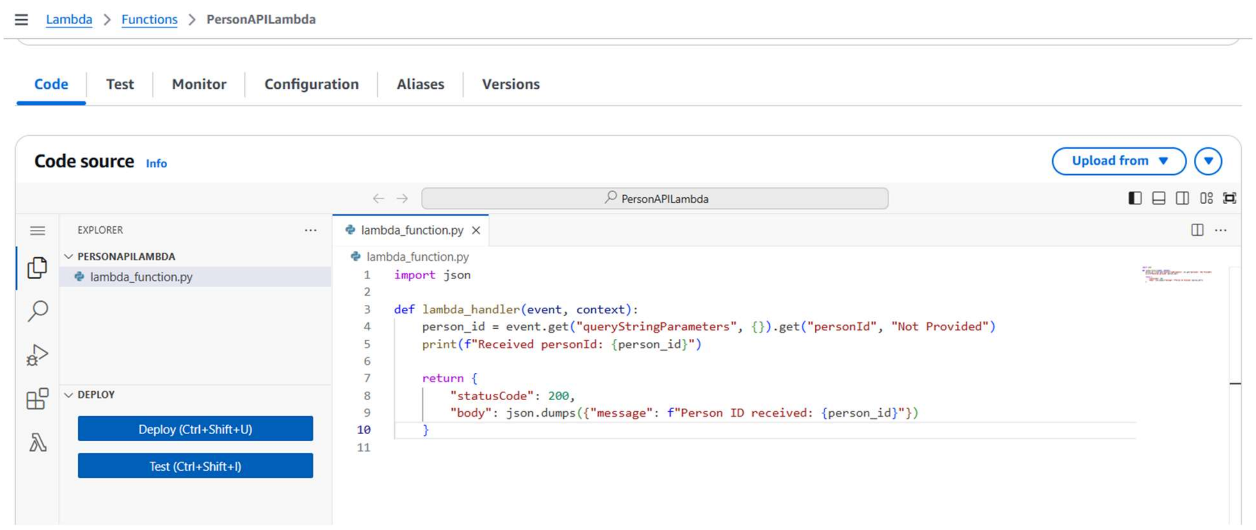
1. In the **Code** tab of `PersonAPILambda`, replace the default code with the following script:

```
import json

def lambda_handler(event, context):
   person_id = event.get("queryStringParameters", {}).get("personId", "Not Provided")
   print(f"Received personId: {person_id}")

   return {
      "statusCode": 200,
      "body": json.dumps({"message": f"Person ID received: {person_id}"})
   }
```

2. Click **Deploy** to save changes.



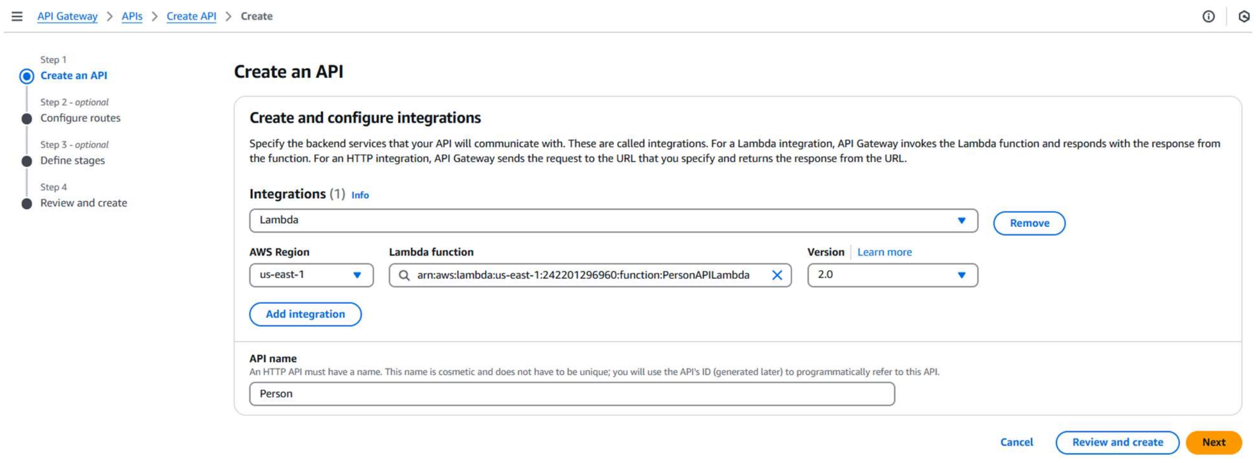
```
import json

def lambda_handler(event, context):
    person_id = event.get("queryStringParameters", {}).get("personId", "Not Provided")
    print(f"Received personId: {person_id}")

    return {
        "statusCode": 200,
        "body": json.dumps({"message": f"Person ID received: {person_id}"})
    }
```

Step 3: Creating an HTTP API Gateway

1. Navigate to the **Amazon API Gateway** console.
2. Click **Create API** and select **HTTP API**.
3. Click **Build** and enter the following details:
   - **API Name:** `Person`
   - **Stage Name:** `$default`
4. Click **Next** and choose **Add Integration**.
5. Select **Lambda Function** and choose `PersonAPILambda`.
6. Click **Next** and add a **GET route** with the following details:
   - **Route path:** `/getPerson`
   - **Integration target:** `PersonAPILambda`
7. Click **Create** to finalize the API setup.



Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations (1) Info: Lambda

AWS Region: us-east-1

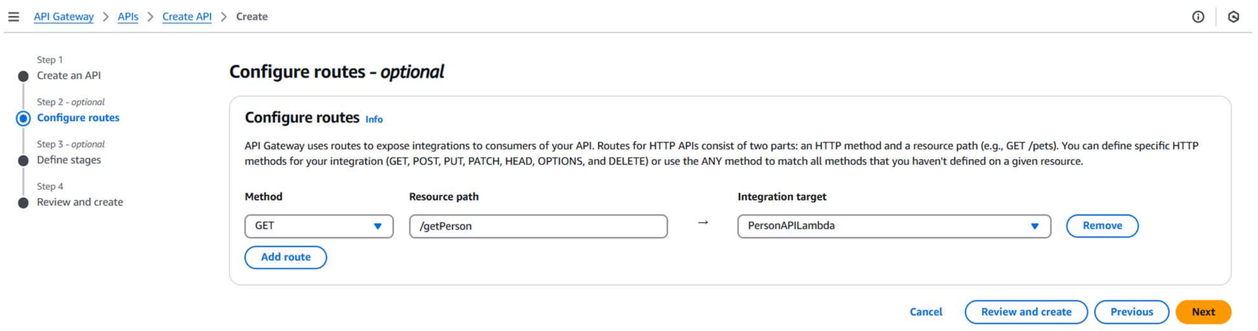
Lambda function: arn:aws:lambda:us-east-1:242201296960:function:PersonAPILambda

Version | Learn more: 2.0

API name

An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

Person

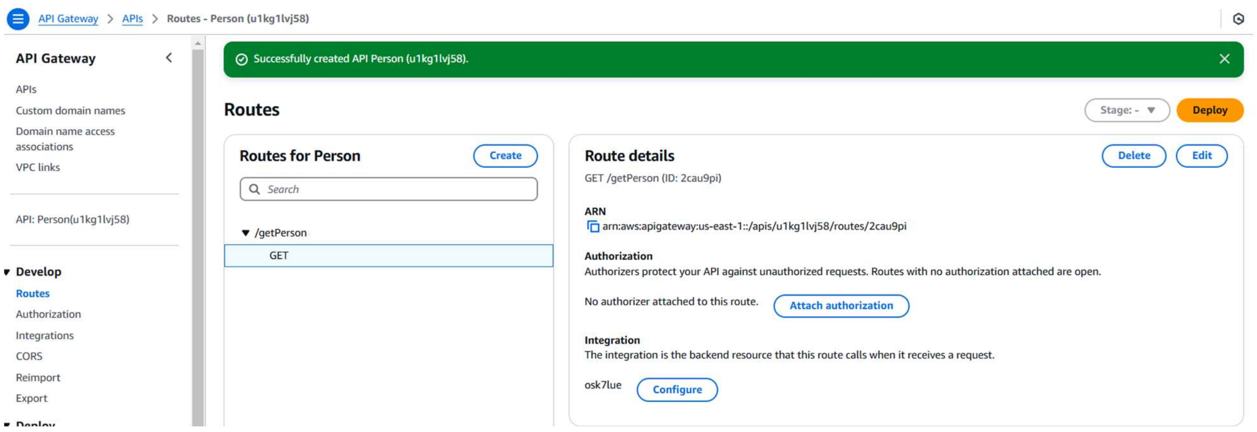


Configure routes - *optional*

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
GET	/getPerson	→	PersonAPILambda



Successfully created API Person (u1kg1lvj58).

Routes

Routes for Person

Route details

GET /getPerson (ID: 2cau9pi)

ARN
arn:aws:apigateway:us-east-1::/apis/u1kg1lvj58/routes/2cau9pi

Authorization
Authorizers protect your API against unauthorized requests. Routes with no authorization attached are open.

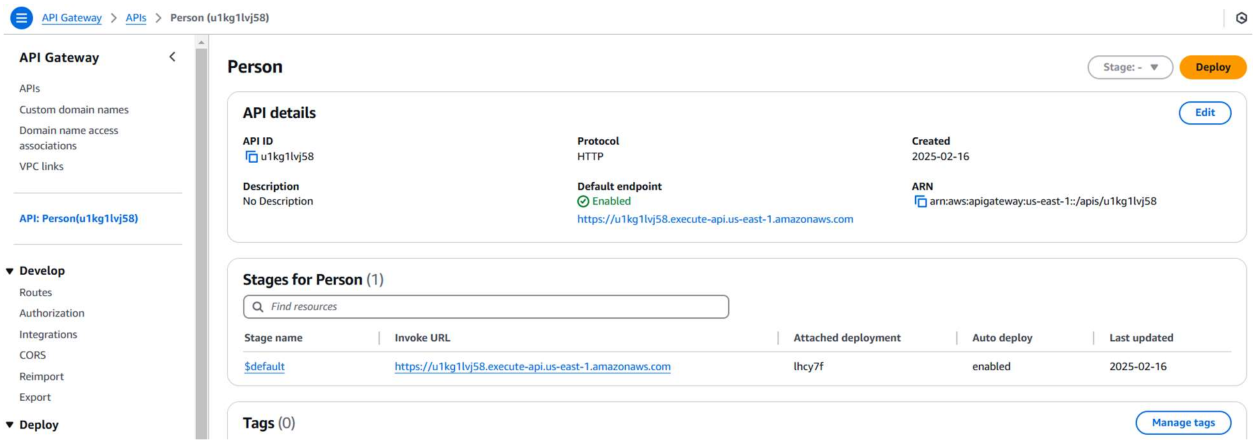
No authorizer attached to this route.

Integration
The integration is the backend resource that this route calls when it receives a request.

osk7lue

Step 4: Deploying and Testing the API

1. Navigate to **API Gateway** and select the **Person** API.
2. Copy the **Invoke URL** from the `$default` stage.
3. Open a browser or use Postman to send a GET request:
   <invoke-url>/getPerson?personId=6
4. Verify that the response contains the extracted `personId`.



Person

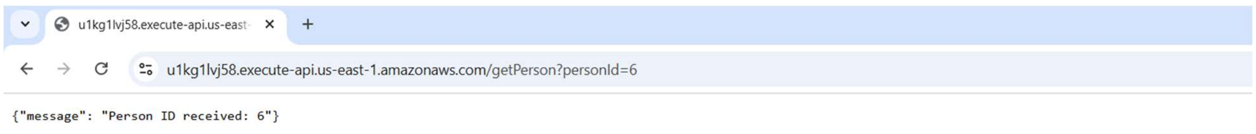
API details

API ID: u1kg1lvj58
Protocol: HTTP
Created: 2025-02-16
Description: No Description
Default endpoint: Enabled
https://u1kg1lvj58.execute-api.us-east-1.amazonaws.com
ARN: arn:aws:apigateway:us-east-1::/apis/u1kg1lvj58

Stages for Person (1)

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
$default	https://u1kg1lvj58.execute-api.us-east-1.amazonaws.com	lhcy7f	enabled	2025-02-16

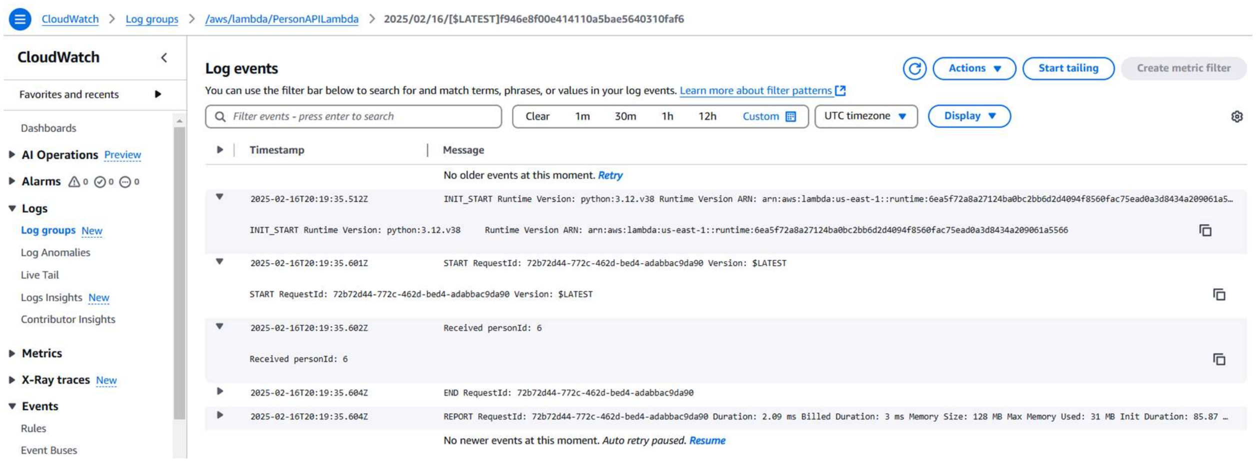
Tags (0)


u1kg1lvj58.execute-api.us-east-
u1kg1lvj58.execute-api.us-east-1.amazonaws.com/getPerson?personId=6

```
{"message": "Person ID received: 6"}
```

Step 5: Observing Logs and Metrics

1. Navigate to **AWS CloudWatch** and open **Logs**.
2. Find the log group for `PersonAPILambda`.
3. Open the latest log stream and verify that the function logs `Received personId: 5`.



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
                           No older events at this moment. Retry
2025-02-16T20:19:35.512Z   INIT_START Runtime Version: python:3.12.v38 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:6ea5f72a8a27124ba0bc2bb6d2d4094f8560fac75ead0a3d8434a209061a5…
INIT_START Runtime Version: python:3.12.v38    Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:6ea5f72a8a27124ba0bc2bb6d2d4094f8560fac75ead0a3d8434a209061a5566
2025-02-16T20:19:35.601Z   START RequestId: 72b72d44-772c-462d-bed4-adabbac9da90 Version: $LATEST
START RequestId: 72b72d44-772c-462d-bed4-adabbac9da90 Version: $LATEST
2025-02-16T20:19:35.602Z   Received personId: 6
Received personId: 6
2025-02-16T20:19:35.604Z   END RequestId: 72b72d44-772c-462d-bed4-adabbac9da90
2025-02-16T20:19:35.604Z   REPORT RequestId: 72b72d44-772c-462d-bed4-adabbac9da90 Duration: 2.09 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 31 MB Init Duration: 85.87 …
                           No newer events at this moment. Auto retry paused. Resume
```

Conclusion

The Lambda function **PersonAPILambda** was successfully set up and integrated with an **API Gateway (Person API)**. It correctly extracts the `personId` from GET requests and logs the value. The API is accessible using the HTTP route `/getPerson?personId=<id>`.

Next Steps

- Add input validation to handle missing or invalid `personId` values.
- Implement error handling for edge cases.
- Enable authentication and authorization if needed.