**Building a CRUD API**

[HTTP API Lambda Function DynamoDB]

**Project Description**

In this project I will build a simple CRUD (Create, read, update, delete) API. Starting with creating a DynamoDB table using the DynamoDB console, creating a Lambda function in the AWS Lambda console. I will configure an HTTP API using the API Gateway console and last, after launching an AWS Cloud9 IDE I will use it to test my API!

**Architecture – Overview**

A diagram of a diagram

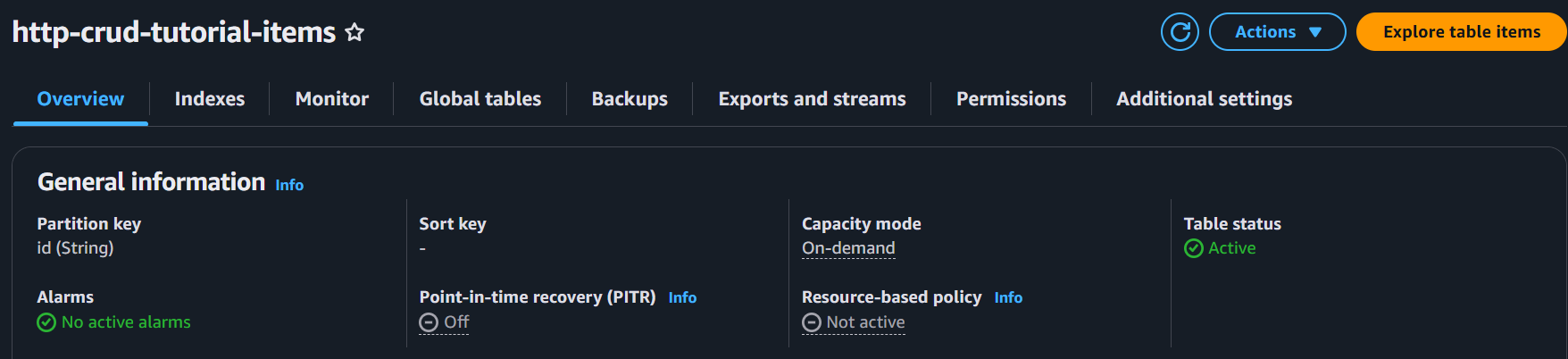
Description automatically generated

**Workflow explanation**

When I invoke my HTTP API, API Gateway routes the request to my Lambda function. The Lambda function interacts with DynamoDB and returns a response to the API Gateway. The API Gateway then returns a response to you.

**Create a DynamoDB table**

 I will use an Amazon DynamoDB table to store data for my API. Each item has a unique ID.



**Create a lambda function**

 I will create a Lambda function for the backend of my API.

The Lambda function **c**reates, **r**eads, **u**pdates, and **d**eletes (**CRUD**) items from DynamoDB.

A screenshot of a computer

Description automatically generated

**Note:** The function uses [events from API Gateway](https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-develop-integrations-lambda.html#http-api-develop-integrations-lambda.proxy-format) to determine how to interact with DynamoDB.





The core logic uses event.routeKey in the switch statement to determine which DynamoDB operation to perform. Then within each case: Delete, Put, Get.

**Create an HTTP API & routes**

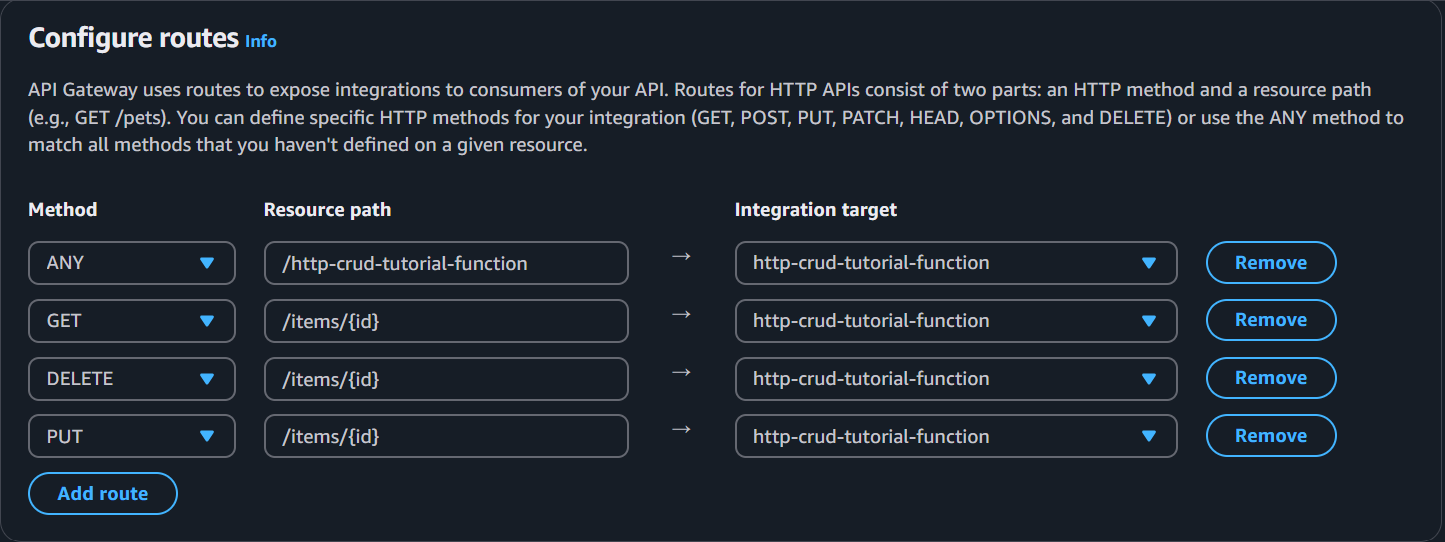
HTTP API provides an HTTP endpoint for my Lambda function

HTTP APIs can be used to send requests to AWS Lambda functions or to any publicly routable HTTP endpoint.

 Routes send incoming API requests to backend resources [GET, PUT & DELETE]

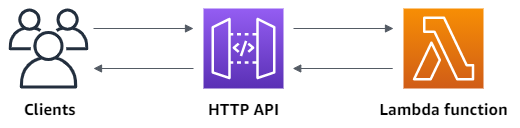
A diagram of a diagram

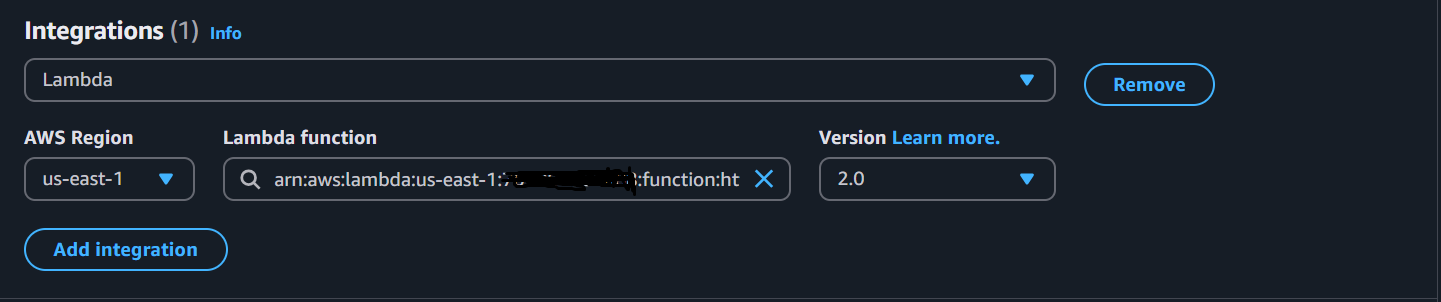
Description automatically generated

****

**Create an integration**

Integrations connect routes to backend resources



****

**Note:** Monoliths work well for the simplest serverless applications that perform single-purpose functions.

As those applications evolve into workflows or develop new features, it becomes important to refactor the code into smaller services.

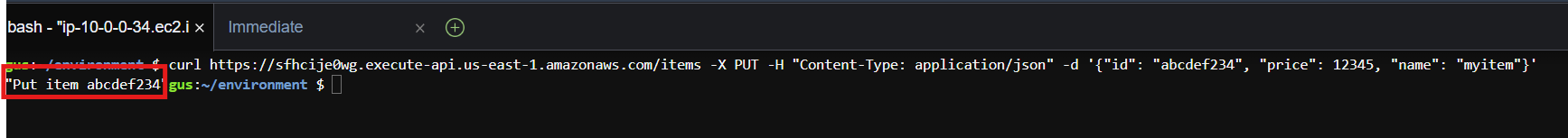
**My HTTP API**

A screenshot of a computer

Description automatically generated

**Testing API AWS with Cloud9 IDE**

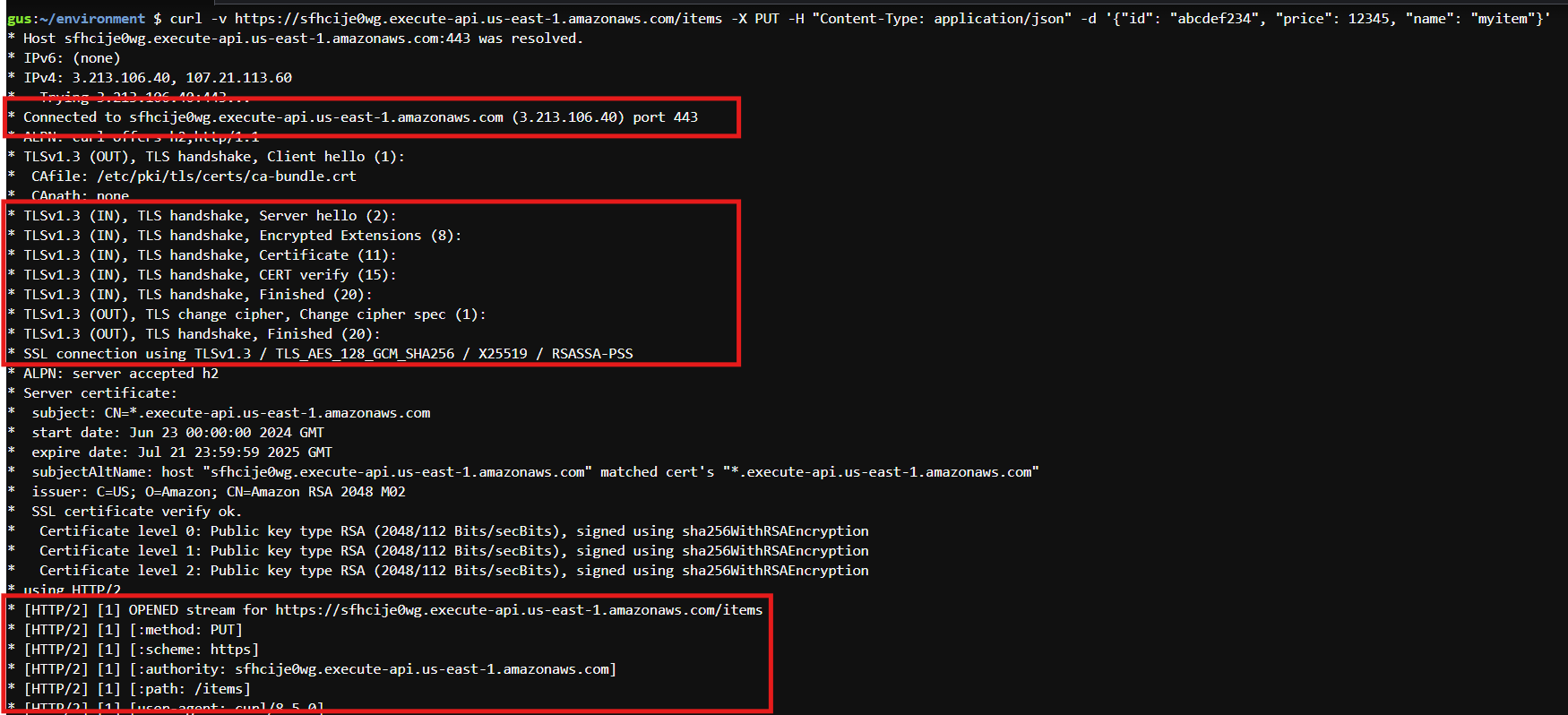
Item update.



A black rectangle with white dots

Description automatically generated

**FYI**



**Done!**