

# Name - Avinash Mule

## API GATEWAY

### Q. What is an API?

-> An **API** (Application Programming Interface) is a set of rules and protocols that allows two different software programs to communicate and exchange data with each other. It acts as a messenger or a middleman, facilitating interactions without the applications needing to know the internal workings of one another.

**Ex-** A food delivery application access the google maps.

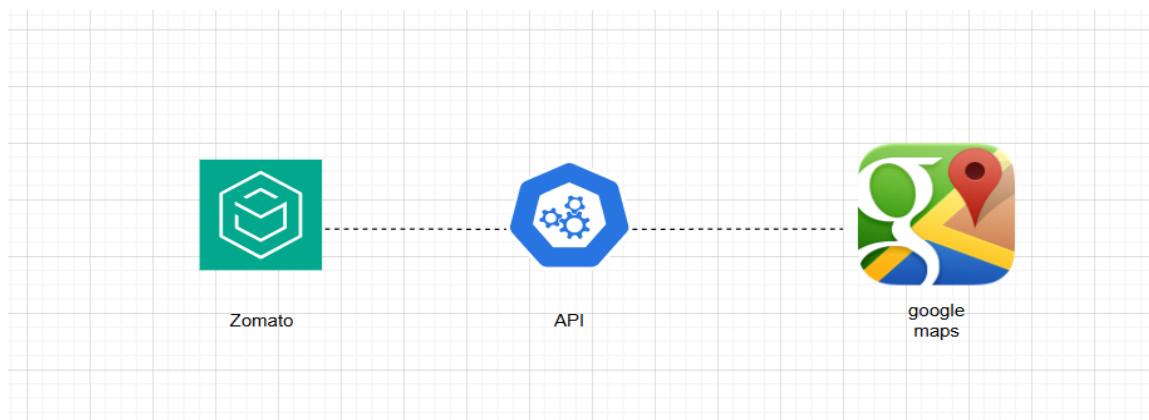


Fig 1 – API diagram of an Application

- Here the food delivery app requests the api to access the data from google maps .
- The API then requests the google maps to access the data and to respond it to the Zomato app.
- API acts as an mediator for authentication , request and respond the data .

## Q. What is API Gateway?

->**Amazon API Gateway** is a fully managed AWS service that acts as a "front door" for applications to access logic and data from backend services, such as AWS Lambda functions, EC2 instances, or any publicly routable web application. It handles the heavy lifting of managing API traffic at any scale, including traffic management, authorization, monitoring, and version management.

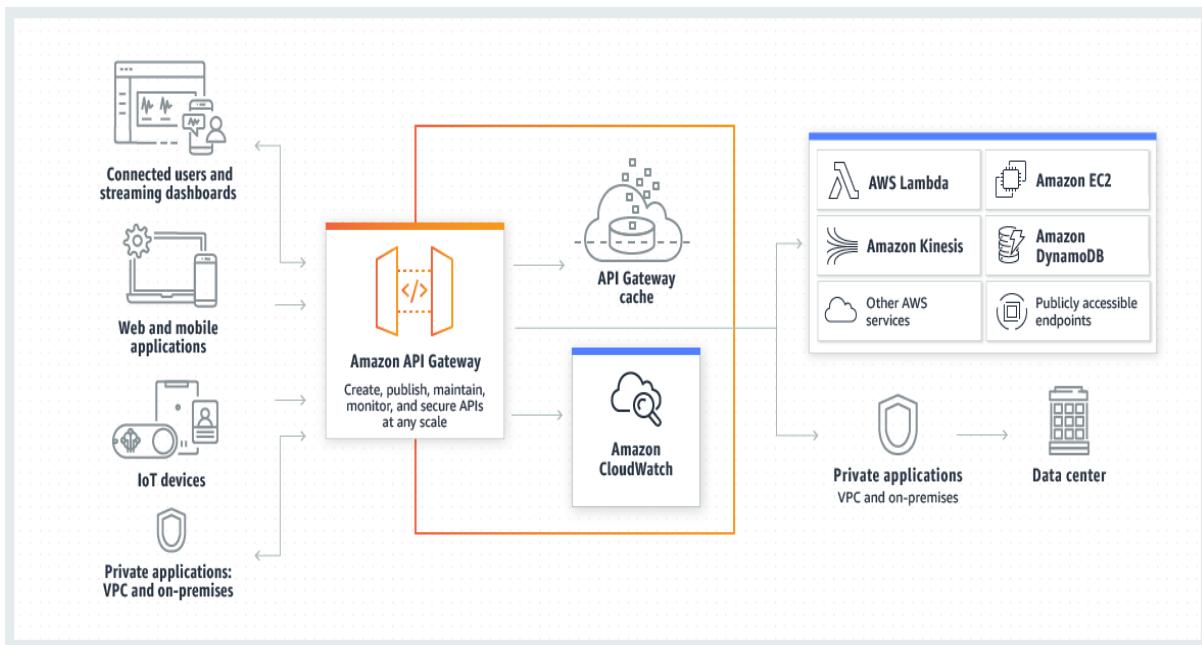


Fig 2 – Diagram of API Gateway

This diagram illustrates how the APIs you build in Amazon API Gateway provide you or your developer customers with an integrated and consistent developer experience for building AWS serverless applications. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls. These tasks include traffic management, authorization and access control, monitoring, and API version management.

## **Core Functionality and Benefits –**

- **Security & Access Control:** It offers multiple security controls, including AWS IAM policies, Amazon Cognito user pools, and Lambda authorizers, to authenticate and authorize access to APIs. It also integrates with AWS WAF to protect against common web exploits.
- **Traffic Management & Resiliency:** API Gateway automatically handles traffic spikes and can be configured with throttling limits (requests per second) and burst limits to protect backend systems from being overwhelmed.
- **Performance & Latency:** It can reduce latency by caching API responses, minimizing the number of calls to the backend. It also leverages Amazon CloudFront's global edge locations for edge-optimized endpoints, improving connection times for geographically diverse clients.
- **Monitoring & Operations:** The service integrates with Amazon CloudWatch for monitoring API calls, latency, and error rates, and with AWS X-Ray for end-to-end tracing.

## **Types of APIs Supported –**

- **REST APIs:** Offer comprehensive API management features, including usage plans and API keys, and support synchronous request-response interactions using standard HTTP methods (GET, POST, PUT, DELETE).

- **HTTP APIs:** A lower-cost, lower-latency alternative optimized for simple proxy functionality to AWS Lambda or HTTP backends. They are ideal for most serverless workloads but have fewer built-in management features compared to REST APIs.
- **WebSocket APIs:** Enable persistent, full-duplex (two-way) communication between clients and servers, making them suitable for real-time applications such as chat apps, streaming dashboards, and online games.

- **Who uses API Gateway?**

- There are two kinds of developers who use API Gateway: API developers and app developers.
- An API developer creates and deploys an API to enable the required functionality in API Gateway. The API developer must be a user in the AWS account that owns the API.
- An app developer builds a functioning application to call AWS services by invoking a WebSocket or REST API created by an API developer in API Gateway.

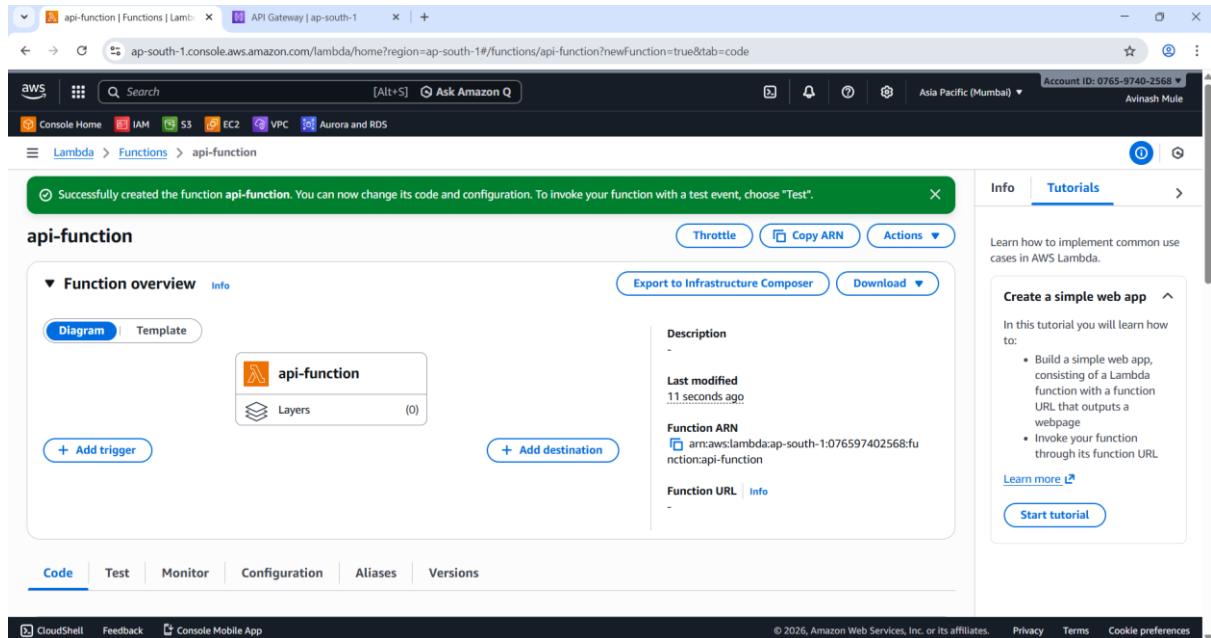
## API Gateway use cases

API gateways serve as a single-entry point for clients, abstracting backend complexities and centralizing common functionalities. They are most used in **microservices**

**architectures, for security enforcement, and to improve API performance and management.**

## Q. How to create an API Gateway?

1. to create api gateway first we need to create a backend service , lets go with lambda service.



- A lambda function is created named api-function.
- In the lambda function a simple python code is deployed and our api-gateway will access the code and return the data.

```

import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello I am Avinash Mule !')
    }

```

## 2. Now we create API Gateway , search api gateway in aws console.

**API details**

**API name**  
An HTTP API must have a name. The name is a non-unique value you use to identify and organize your APIs. To programmatically refer to this API, use the API ID that API Gateway generates for you.

**IP address type** [Info](#)  
Select the type of IP addresses that can invoke the default endpoint for your API. You don't need to redeploy your API for the update to take effect.  
 **IPv4**  
Includes only IPv4 addresses.  
 **Dualstack**  
Includes IPv4 and IPv6 addresses.

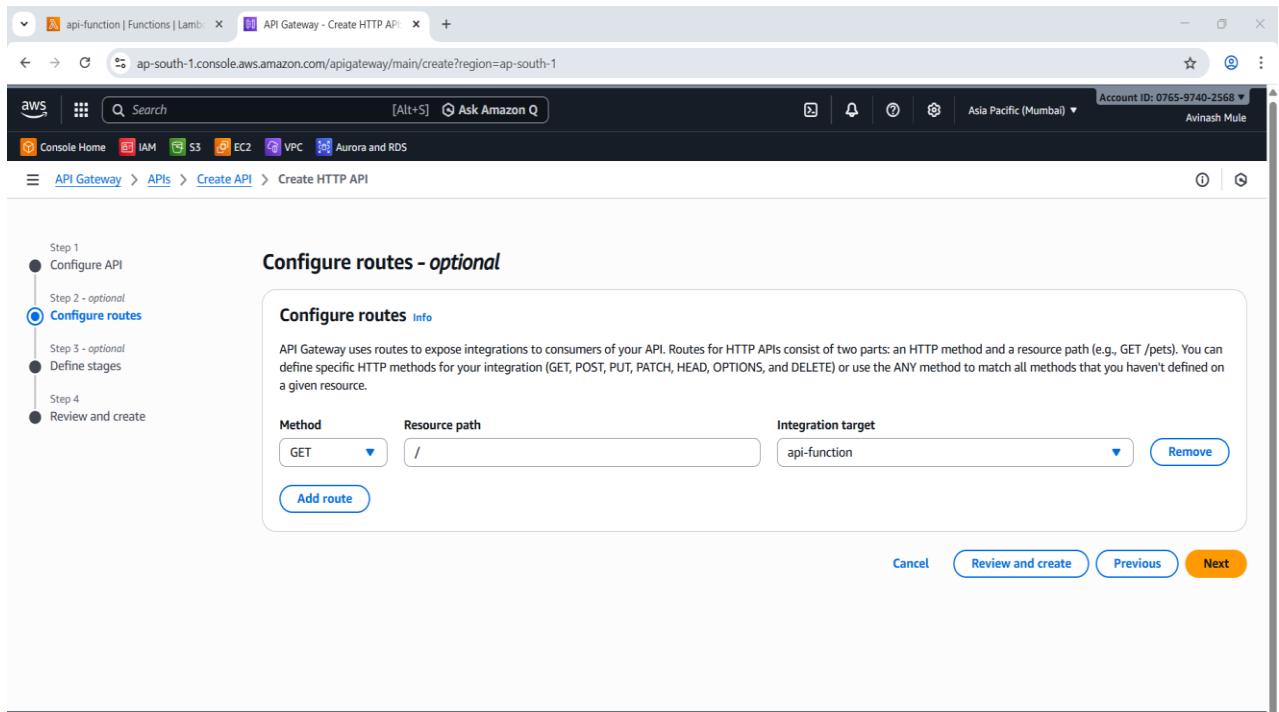
**Integrations (1) [Info](#)**  
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.  
**Lambda** [Remove](#)  
**AWS Region**: ap-south-1    **Lambda function**: arn:aws:lambda:ap-south-1:076597402568:function:api-functi    **Version**: 2.0 [Learn more](#)

- Here we created api gateway and named it my-api-gate.

- We integrated lambda function , selected the lambda function ARN and created API.

3. Now we configure the routes , as we are using HTTP API  
We configure the method , path and integration target.

- We select GET method because we want to return the value when the api endpoint is hit.



4. The api is created now copy the endpoint and hit it on browser.

The screenshot shows the AWS API Gateway - API settings page. The main content area displays the configuration for the API named 'ugyyfuezhb'. Key details include:

- API ID:** uggzfuezhb
- Protocol:** HTTP
- Created:** 2026-01-12
- Default endpoint:** Enabled, https://uggyfuezhb.execute-api.ap-south-1.amazonaws.com
- ARN:** arn:aws:apigateway:ap-south-1::apis/uggyfuezhb

The left sidebar shows the navigation path: API Gateway > APIs > my-api-gate (uggyfuezhb). The sidebar also includes sections for Develop (Routes, Authorization, Integrations, CORS, Reimport, Export) and Deploy (Stages).

**Stages for my-api-gate (1)**

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	<a href="https://uggyfuezhb.execute-api.ap-south-1.amazonaws.com">https://uggyfuezhb.execute-api.ap-south-1.amazonaws.com</a>	g555s2	enabled	2026-01-12

**Tags (0)**

Key	Value

## 5. we can see our lambda function output .

The screenshot shows a browser window with the URL [uggyfuezhb.execute-api.ap-south-1.amazonaws.com](https://uggyfuezhb.execute-api.ap-south-1.amazonaws.com). The page displays the output of the Lambda function:

"Hello I am Avinash Mule !"