API Gateway

19 June 2023 22:37

What's API :

API stands for Application Programming Interface. It is a set of rules and protocols that allows different software applications to communicate and interact with each other. An API defines how different software components should interact, specifying the methods, data formats, and protocols that can be used for communication.

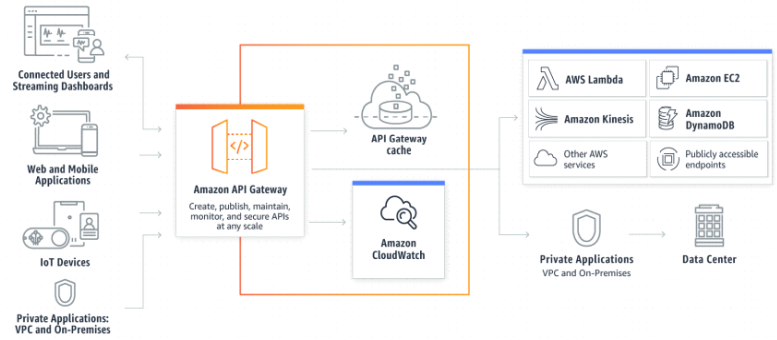
APIs can be classified into different types based on their purpose and usage. Some common types include:

- **Web APIs**: These APIs are designed for communication over the web and are commonly used in web development. Examples include REST APIs, SOAP APIs, and GraphQL APIs.
- **Library/APIs**: These are APIs provided by software libraries or frameworks that developers can use to access specific functionalities or resources within those libraries.
- **Operating System APIs**: These APIs provide a way for applications to interact with the underlying operating system. They offer access to system-level services and resources, such as file systems, network connections, and hardware devices.
- **Third-Party APIs**: These are APIs provided by external organizations or services that allow developers to integrate their applications with third-party platforms, services, or data sources. Examples include social media APIs, payment gateway APIs, and mapping APIs.

Here are some common examples:

- **Social Media APIs**: Platforms like Facebook, Twitter, and Instagram provide APIs that allow developers to access their services and integrate social media functionalities into their applications. These APIs enable developers to post content, retrieve user data, interact with social graphs, and perform various social media-related tasks.
- **Payment Gateway APIs**: Companies like PayPal, Stripe, and Braintree offer APIs that enable businesses to integrate online payment functionalities into their applications or websites. These APIs allow developers to securely process transactions, handle payments, and manage user accounts.
- **Mapping APIs**: Services like Google Maps, Mapbox, and OpenStreetMap provide mapping APIs that developers can use to embed interactive maps, geolocation, and routing functionalities into their applications. These APIs allow developers to display maps, find directions, geocode addresses, and perform spatial queries.
- **Weather APIs**: Companies like OpenWeatherMap and Weatherbit provide weather APIs that allow developers to retrieve real-time weather data and forecasts for specific locations. These APIs enable developers to incorporate weather information into their applications, such as displaying current conditions, weather forecasts, or creating weather-based notifications.
- **Travel APIs**: Platforms like Expedia, Skyscanner, and Airbnb offer APIs that enable developers to access their travel and accommodation services. These APIs allow developers to search for flights, hotels, car rentals, and other travel-related information, as well as book reservations and retrieve booking details.
- **Financial Market APIs**: Services like Alpha Vantage, Yahoo Finance, and Bloomberg provide APIs that offer access to financial market data, including stock quotes, historical prices, market indices, and company information. These APIs enable developers to build financial applications, algorithmic trading systems, and portfolio management tools.
- **Cloud Service APIs**: Cloud providers like Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure offer APIs that allow developers to manage and interact with their cloud services. These APIs enable developers to create and manage virtual machines, storage resources, databases, and various cloud-based services

What is Amazon API Gateway, Main Features and Core Concepts.



What is Amazon 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 backend services. Using API Gateway, we can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway also supports containerized and serverless workloads as well as web applications.

So we can say that, we can create, deploy, and manage a RESTful APIs in order to expose backend HTTP endpoints, AWS Lambda functions, or other AWS services with using Amazon API Gateway.
API Gateway creates RESTful APIs that are HTTP-based. And enable stateless client-server communication and also implement standard HTTP methods such as GET, POST, PUT, PATCH, and DELETE.

There are 3 main types of APIs in Amazon API Gateway;

- HTTP API
- REST API
- Websocket API

HTTP API

HTTP APIs are the best choice for building APIs that only require API proxy functionality.

REST API

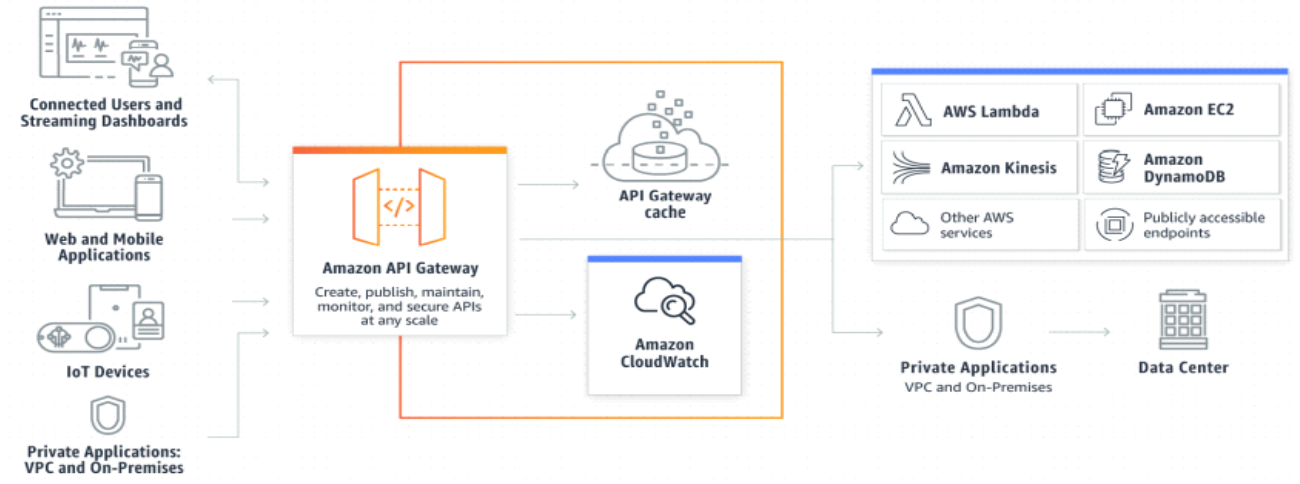
Restful APIs for require API proxy functionality and API management features.

Websocket API

Websocket API for building real-time two-way communication applications, such as chat apps and streaming dashboards.

Architecture of API Gateway

API Gateway acts as a “front door” for applications to access data, business logic, or functionality from your backend services. API Gateway handles all tasks involved in accepting and processing thousands of concurrent API calls. When you create APIs in Amazon API Gateway, it also provides an integrated and consistent developer environment.



Main Features of API Gateway

Amazon API Gateway has lots of features, but we will share only important ones:

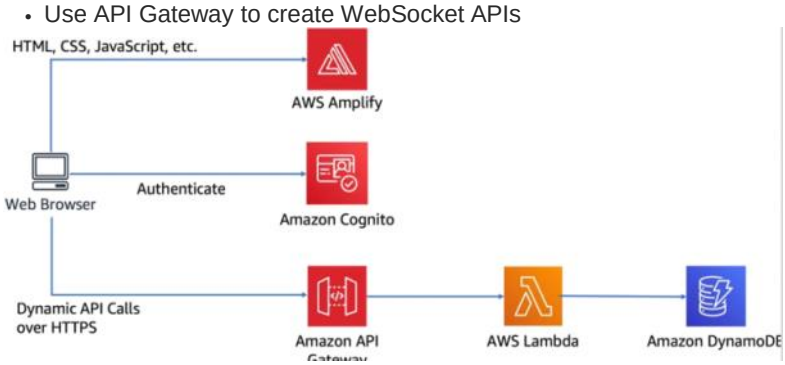
- Support for stateful WebSocket and stateless HTTP and REST APIs.
- Provide different authentication methods like AWS IAM, Lambda authorizer and Amazon Cognito user pools.
- Supports OAuth2 and OpenID Connect protocols.
- Built-in CORS support and automatic deployments
- Provide Developer portal for publishing Restful APIs
- Canary release deployments for safely rolling out changes.

- Logging and monitoring of API usage and API changes
- Ability to use IaC tools like AWS CloudFormation templates to enable API creation

Amazon API Gateway Use Cases

There are 3 main use cases for Amazon API Gateway

- Use API Gateway to create HTTP APIs
- Use API Gateway to create REST APIs
- Use API Gateway to create WebSocket APIs



404 Not Found (amazon.com)

First use case is Use API Gateway to create HTTP APIs. The main use case of Amazon API Gateway is creating HTTP APIs. HTTP APIs enable you to create RESTful APIs with lower latency and lower cost than REST APIs.
Second one is Using API Gateway to create REST APIs. An API Gateway REST API is made up of resources and methods. A resource is a logical entity that an application can access via a resource path.
The last use case is Use API Gateway to create WebSocket APIs. In a WebSocket API, the client and the server can both send messages to each other at any time.

Amazon API Gateway Core Concepts

We will see these Core Concepts directly into AWS Management Console when creating API Gateways. The main core concept is API Gateway. But also we can API deployment, API endpoint and Proxy integrations are other core concepts that we can configure when create API Gateways.

- API deployment
  A snapshot of your API Gateway API. The deployment must be associated with one or more API stages for clients to use it.
- API endpoint
  Hostname for an API deployed to a specific Region in API Gateway.
- Proxy integration
  A simplified API Gateway integration configuration.

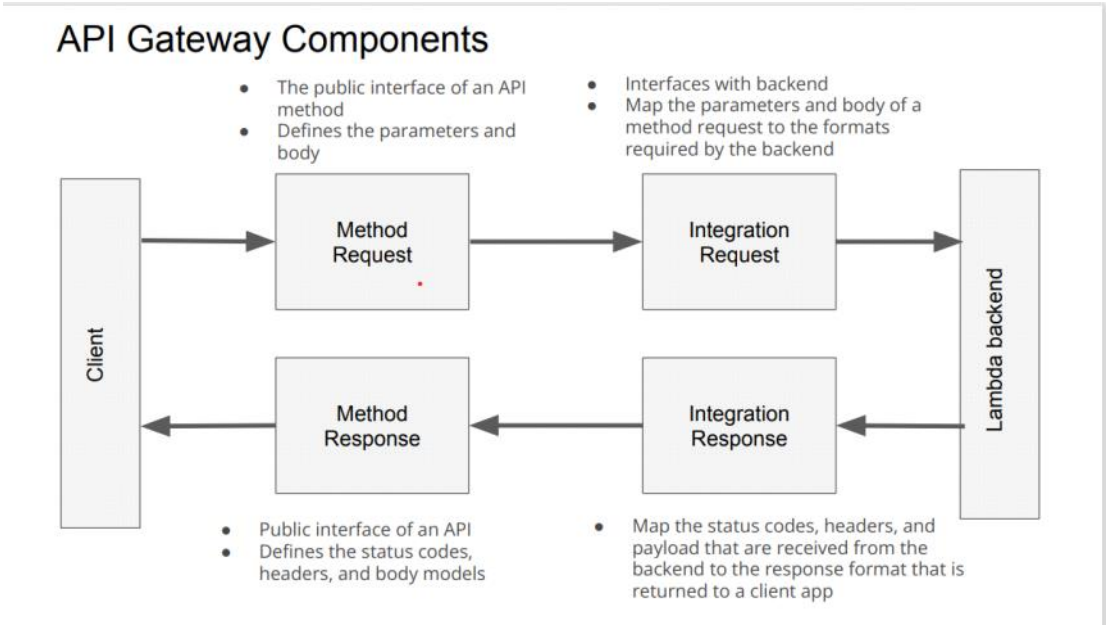
API Gateway Components - Continued

Invoke URL - URL to invoke the API

Usage Plan - A usage plan provides selected API clients with access to one or more deployed APIs. You can use a usage plan to configure throttling and quota limits, which are enforced on individual client API keys

API Developer - Your AWS account that owns an API Gateway deployment (for example, a service provider that also supports programmatic access.) App Developer - An app creator who may or may not have an AWS account and interacts with the API that you, the API developer, have deployed. App developers are your customers. An app developer is typically identified by an API key.

Resources (could be different projects/business areas) Methods (GET, POST etc.) - Each method along with resources, are deployed to stages, with invoke url for each method under each resource in each stage



Query Parameters : Query parameters to be appended to the path. (https://ap-south-1.console.aws.amazon.com/console/home?nc2=h_ct&region=ap-south-1&src=header-signin#)

parameterName: Name of the query parameter

type: The data type of the query parameter. Supported types are string, number, or boolean.

repeating: Set to **True** if more than one value is expected for the query parameter.

required: Set to **True** if query parameter is a required configuration. The trigger reports an error if no value(s) are provided to the required query parameter.
Note: By default, query parameters are not mandatory. When you create an API Gateway, you must explicitly navigate to the API Gateway console and change the settings.

Tech Examples :

- Creating API with Lambda Integration

Create Get method with below Lambda Code

Get Example with Lambda

Create API
Create Get method with below Lambda Code
Test get method using Console
Test get method using Postman

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

Post Example with Lambda

Create API
Create Post method with below Lambda Code
Test Post method using Console
Test Post method using Postman
(Send request as Post request and message as Body)

```
{
  "Country" : "India"
}
```

```
import json
def lambda_handler(event, context):
    # TODO implement
    print(event)
    return 'Hello from ' + event['Country']
```

Demo Query Parameters

Create API EndPoint
Create Resource
Create Get Method
GoTo Method Request and URL Query String Parameters - > Add Query String -> Add nameOfCountry as a parameter and Mark it's as required.
GoTo Request Parameter "Validate query sting parameters and headers"
Test and Show Error
GoTo IntegrationRequest and GoTo Mapping Templates and addmapping templates

give ContentType as "application/json"..

```
{
   "Country" : "$input.params('nameofCountry')"}
}
```

And Test it again with Query String as

nameOfCountry=India
nameOfCountry=US

Output : -
"Hello from India"
"Hello from US"

AWS Cross Account Lambda Call

Create Lambda in One Account
Create API Gateway Get Request in one account
Copy ARN of Lambda function and paste while adding Lambda to API Gateway
Create Cloud9 Environment to Run the Command while adding Lambda to API Gateway

```
aws lambda add-permission
      --function-name "arn:aws:lambda:ap-south-1:722123124527:function:SecondAccountLambda"  --source-arn "arn:aws:execute-api:ap-south-1:347763580245:k338zfe868/*/GET/testresource"  --principal
apigateway.amazonaws.com  --statement-id 51211ef8-9645-467e-80ac-52a11be9c2a9  --action lambda:InvokeFunction
```

Lambda Version and Alias

Create Lambda Function
Create Different Versions Till V1 To V4 with Different Codes
Create Alias with 50 % traffic to V1 and V4 to and show the output.

API traffic Splitting Using Lambda Version/Alias (API Gateway)

Create Lambda Function
Create Different Versions Till V1 To V4 with Different Codes
Create Alias with 50 % traffic to V1 and V4 to and show the output.

Use this as Lambda function at the time of API gateway creation : versionTest:${stageVariables.lambdaAlias}

And once you click on save it will give below permission updating thing

Run this code in Cloud9.

```
aws lambda add-permission
      --function-name "arn:aws:lambda:ap-south-1:347763580245:function:versionTest:TestAlias"  --source-arn "arn:aws:execute-api:ap-south-1:347763580245:830xfkfo0a/*/GET/testresource"  --principal
apigateway.amazonaws.com  --statement-id a5593ef1-185f-47dc-b01b-984a321ce944  --action lambda:InvokeFunction
```

Make sure to change Red highlighted one.

API Gateway Canary Deployment

Create two Lambda Functions "Hello1" and "Hello2"
Create API Gateway like get API with "Hello1"
Deploy API and test all request are going to one path..
Create Canary Deployment and Deploy again to Dev(Canary) and test
Move 50-50 % Percentage of To older version and newer version