REST APIS USING AMAZON AWS API GATEWAY

Pavan Kumar K N[1], L Spoorthi[2] and Dr. Soumya A[3].

Student, Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India[1,2]
Assistant Professor, Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India[3]

Abstract : APIs (Application Programming Interfaces) are becoming increasingly important for businesses in a variety of industries, as they provide the way by which two different applications may connect. APIs empower organizations to develop their organizations more rapidly than any time recently. APIs offer organizations the chance to scale, cultivate development and contact a more extensive crowd.In future APIs have potential to transform businesses as it makes ease integration of backend data and applications.Cloud Computer is the most common computing model nowadays, and most ISPs (Internet Service Providers) have launched cloud offerings.REST interfaces are commonly used to expose cloud services on the web, but also do not provide a formal grounding that can be used to semantically characterize the accessible services.In this paper, we present an analysis of amazon aws api gateway and general methodology for developing RESTful api using aws api gateway

IndexTerms **- Amazon AWS, AWS GateWay, REST API, AWS Lambda, DynamoDB.**

I. INTRODUCTION

The Application Programming Interface (API) is a communication interface for connecting two apps.When we use a smartphone, it connects to the Internet and sends data to a server. The server then retrieves the data, interprets it, and delivers it back to our smartphone with the relevant actions.. The software then analyses the data and displays the information we requested in a legible fashion. All of this happens through an API. APIs make it easier for developers to integrate new application components into an existing architecture, facilitating collaboration between business and IT teams. Business requirements change often in response to fast changing digital markets, where new competitors could decimate an entire industry with a single app.

The Amazon API Gateway service allows us to create, publish, maintain, monitor, and secure WebSocket, HTTP, and REST APIs at any scale. Apis developers can create APIs that connect to Amazon Cloud Services (AWS) or other web services, as well as data stored in the AWS Cloud.RESTful APIs created by API Gateway are HTTP-based, make client-server communication stateless and uses common HTTP methods like DELETE,PUT,POST,PATCH AND GET

II. KEY CONCEPTS

To be able to explore more in the area of APIs using AWS, it is necessary to understand some of the tools that Amazon AWS has to offer. Using the following tools we will be able to efficiently create an API and deploy it in a safe and a secure environment.

2.1. AWS DynamoDB : DynamoDB is a NOSQL Database service provided by Amazon AWS. It is designed to provide quick and easy access to the data stored in the Database. It provides a different set of API capabilities to have programmatic access, given that the authorization to the Database holds good. The Provided GUI management console gives access to the user to easily access tables and configure them according to the requirements.Tables, attributes, and items are the three basic components of DynamoDB. A table has a collection of things, an attribute is the most basic element for storing data without further separation, and an item has a collection of attributes.To address changing application demands, DynamoDB employs eventually consistent and highly consistent reads. Consistent readings do not necessarily yield current data in the end.

The very consistent readings always offer up-to-date information (with the exception of equipment failure or network problems). Consistent readings will eventually become the default mode, requiring a true value in the Consistent Read parameter.

2.2. AWS Lambda : Lambda is a computing service from Amazon Web Services that enables us to run code without the need to set up or manage servers. AWS Lambda executes our code just when it's required, allowing us to scale from a few responses per day to hundreds per second with ease. We only have to pay for computer time we use; when our code is not executing, there is just no price.AWS Lambda provides the infrastructure for our code to be uploaded. It's in charge of keeping the code up to date and triggering it when the appropriate event occurs. It lets us select the storage and timeout requirements for the code.
AWS Lambda may also run many requests in parallel based on events. AWS Lambda provides information such as the number of instances a code was executed, the time it took to run, the amount of memory used, and so forth. AWS CloudWatch captures all logs, which aids in debugging the code and analyzing the program execution.
For composing and deploying code, Lambda offers a variety of alternatives. We may use the AWS web editor, Visual Studio IDE, or Eclipse IDE to write our code. It also offers serverless framework compatibility, which makes creating and deploying AWS Lambda applications a breeze. AWS-cli, in addition to the AWS console, is used to build and deploy code.

2.3. AWS Cloudwatch : Amazon CloudWatch keeps track of the functions executed in AWS Lambda. Whenever the Lambda function is invoked, it assists in recording any requests made to it. The following piece of code will attach the message associated with a timestamp to the cloud watch so that it will be easier to fetch and monitor the particular event.

```
console.log("Cloud Watch log monitoring");
```

2.4. AWS API Gateway : Gateway is made up of Lambda functions that work together to form a serverless learning management system.
API Gateway is a highly scalable service that makes publishing, maintaining, monitoring, and securing APIs at whatever size simple for developers.
We may establish an API that works as an "entryway" for apps to access data, business logic, and other resources with only a few keystrokes in the AWS Management Console.
As an API Gateway offers a graded pricing mechanism for API calls, it is highly relatively inexpensive and effective. The cost of an API request is as low as $1.51 per million requests, but we might save money if we reduced the number of requests. We won't have to worry about EC2 or Autoscaling groups responding to API requests. An API Gateway is self-scaling. The most important aspect of API Gateway is caching. The endpoints are cached, which lowers the time it takes for our API to respond to queries. It is, without a doubt, a significant factor in setting the price of a service.
We may also protect our API Gateway from security threats. API Gateway validates inbound requests using a variety of authorization methods, including the Identity Access Management service and the Lambda function. An IAM is coupled to a gateway, which provides API interaction tools such as AWS credentials.

2.5. AWS RDS : RDS is Relational Database Service, which is available as a web service. It simplifies the setup and management of a relational database in the cloud. It offers a very cost-effective managed service for the industry's major RDBMS software. We don't need to acquire a server or install any database software because of Amazon AWS online service. Amazon RDS is in charge of software patching, backups, recovery, and automatic failure detection.
In order to provide a controlled customer experience, Amazon RDS will not grant shell access to database instances. It also blocks access to advanced privileges-required system operations and tables. We may have backups done automatically when we need them, or we may build our own backup snapshot explicitly. These backups can be used to recover a database. The Amazon RDS restoration method is dependable and quick.
We could use MariaDB, PostgreSQL, MySQL, Microsoft SQL, or Oracle, all of which we are familiar with. Using AWS Identity and Access Management to create users and permissions in conjunction with the security in our database package, we can effectively manage who has access to our RDS databases. To better protect our databases, we may put them in a VPC.

2.6. AWS IAM : AWS Identity and Access Management is a web application service that allows us to handle access to Amazon AWS services in a safe and secure manner. IAM allows us to manage who is authenticated (registered on) and allowed (granted rights) to access shared services. Without sharing our password or access key, Others may be granted access to our AWS account to administer and use services.. For distinct resources, we can provide various permissions to various persons.We may grant full access to Amazon S3, Amazon Elastic Compute Cloud, Amazon Redshift, Amazon DynamoDB, and other Amazon AWS services to specific users. IAM features may be used to offer secure credentials for apps running on EC2 instances. AWS Security, Access Management and AWS Identity Token are free services available with an Amazon AWS account

III. ARCHITECTURE

Amazon API Gateway is a Node.js-based closed-source Software-as-a-service ,available only in Amazon AWS. In the AWS ecosystem, Amazon API Gateway can be considered a backplane. API Gateway allows apps to access business logic, data, or functionality from their backend services, such as Amazon Elastic Compute Cloud workloads, RDS, DynamoDB, AWS Lambda code, real-time communication or any website apps.Use an API Gateway-created HTTP endpoint to connect to a service that doesn't have a native integration. The ideal method to accomplish this is to create an AWS Lambda function that contains all of the integration's business logic, and then connect that Lambda function to an API Gateway HTTP endpoint.CloudWatch provides us with meaningful insights and data to monitor our apps, optimise resource utilisation, respond to system-wide performance changes, and get a single view of operational health.

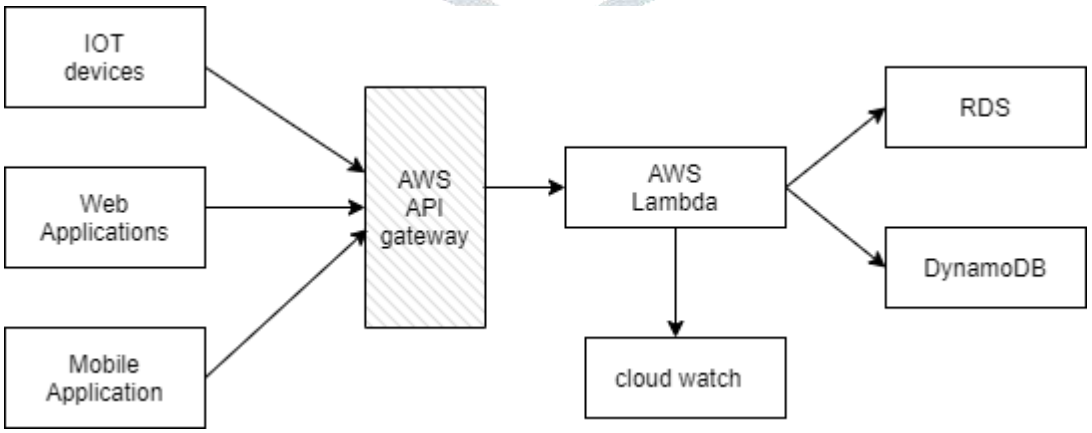

Fig-1: AWS API gateway architecture

IV. FEATURES.

Amazon AWS API gateway exhibits a number of features as follows-

- It works with both stateless (REST) and stateful (WebSocket) APIs.
- It includes a developer portal, which the API developer can utilise to publish their APIs.
- It has high-level, and flexible authentication methods, which include Policies for AWS Identity and Access Management, Amazon Cognito user pools, Lambda authorizer functions, and are all available.
- It is known to provide lowest possible latency for the API requests and API response compared to other cloud services providers.
- It is very cost effective, charging only for what has been used.
- API Gateway provides various tools for monitoring, and testing and to perform regular health checks.
- It provides a secure environment to host an API by offering IAM i,e Identity and access management tool, OIDC and OAuth2 support.
- It provides the flexibility to build an API by using HTTP APIs or REST APIs.
- Provides an effortless access to all of the tools with an easy to use management console

V. METHODOLOGY.

The following steps are taken as the standard approach to creating a well built API, capable of handling all the Create, Delete, Update, Read operations using AWS. For this we will make use of specific AWS tools such as

- AWS Amazon API Gateway
- AWS Lambda
- AWS DynamoDB

The general steps followed to create and implement an API is as follows,

- Create a DynamoDB Table.
- Create a Policy and Role for lambda function
- Create a Lambda function
- Configure API Gateway and deploy
- Testing the API to make sure of CRUD operation

The above mentioned steps are the basic fundamental steps to create a simple API and have it up and running very quickly.
As the first step suggests, first we need to create a table in DynamoDB. To do this, Just login to the amazon AWS account. Search for DynamoDB. Locate the create table option. Here we need to provide a Table Name which will be used throughout the course of this paper. Also we may need to provide the Primary key while creating a new table.
Once the table is created, We need to take a note of the ARN i.e Amazon Resource Name. Using the ARN we will be able to create a policy better suited to the table. Which will be covered in the following step.
The second step, We need to create policy and roles for the lambda function. To do this first head over to IAM services in the AWS management console, Click on Policies and Create Policies. Here we are able to create Roles and Policies for a particular resource. Creating this with JSON is much simpler, Just align the permissions, Actions, Resources within the JSON and with this we now can create a policy. To specify the resources, we can do so by selecting the ARN we acquired from the table we created in the DynamoDB. Once it is done, We can review the policies we have created, Create a policy by giving it a name and a brief description, then clicking Create a Policy.

Making sure that we have created the policy, we need to create a role. We can do so by logging into the management console and heading over to Roles in IAM. Click on create a new role and choose Lambda as the use case. Next we need to attach the permissions. As we have already created the policy, we can attach that to the lambda. Search for the policy and select it. The next step is to give the newly formed role a name and a brief description, then click Create Role.
The third step is to write a lambda function. To do so, go to the administration console and type lambda into the search box. Click on create a function and give a simple function name. Select language to use to write our function, the simple option is to go with Node.js but we can pick from a wide array of different languages. Once that is done, now we need to pick a role for the Lambda. We have already created the role with customized permissions, we can select the same role. And click on create function.
Once the function is created, We can modify the code that represents that function by modifying the index.js file. While doing so DynamoDB provides a different set of functions to perform the get, insert, update, delete operations. All of these functions can be imported from the dynamodb package for Node.
While coding for the function make sure to cover all the CRUD operations. The following explains the structure of the Lambda function.
Create an object of the class, AWS.DynamoDB.DocumentClient(). The created Object will have the capabilities to implement all the Dynamodb operation such as,

Object.get(payload, callback): for Read operation
Object.update(payload, callback): for Update operation
Object.delete(payload, callback): for Delete operation
Object.put(payload, callback): for Create Operation

Simple implementation of the above functions in a switch statement, we can create the conditions required to make all the CRUD operations possible with the DB.Once the implementation of the Lambda function is done, we can save the function.

The next step is to create API Gateway using Amazon AWS API Gateway and deploy our API. To do so, login to the management console of AWS and navigate to the AWS API Gateway.Click on create API and select the type of the API as REST API.
Select a name and a short description for the API and click on create API.
Once that is done we will be navigated to the API Configuration Page. For the API, we'll need to construct Methods and Resources. Create a resource by passing in the Lambda function we just constructed. Select a method from the conventional GET, POST, PUT, DELETE, and the Lambda function we developed for the same purpose, then click Create to create the action and link it with our API Gateway.
Then we can deploy the API by clicking the deploy option from the action dropdown. Keep a note of the invoke URL which will be used to invoke the API gateway.
The next step is Testing, Testing of the created API can be done using Postman or Insomnia.
We can send API requests using the API URL that we acquired and perform all the operations. As a return we will receive the appropriate response from the API Gateway. Following are the API Responses,
A successful API call will return :

```
{
  "statusMessage": "SUCCESS"
}
```

An API call where the targeted element does not exists will return :

```
{
  "statusMessage": "NOT_FOUND"
}
```

An API call where the selected payload a;ready exist in the selected DynamoDB will return :

```
{
  "statusMessage": "DUPLICATE"
}
```

The next and final step is to monitor and regular health checks to make sure that the API is working as expected. To do this we may use CloudWatch. Cloud Watch is another Amazon AWS tool which stores the log files for everything that goes on in the AWS. We can review and check for the exact timestamp and lookup logs to keep track of our API. Another helpful tool is API Gateway itself, it gives us a lot of information and options to fine tune our API and perform regular health checks. Using this we can draw up the history of the API use, Where and Who has used the API to do What can also be cherry picked and stored as logs for future monitoring purposes.

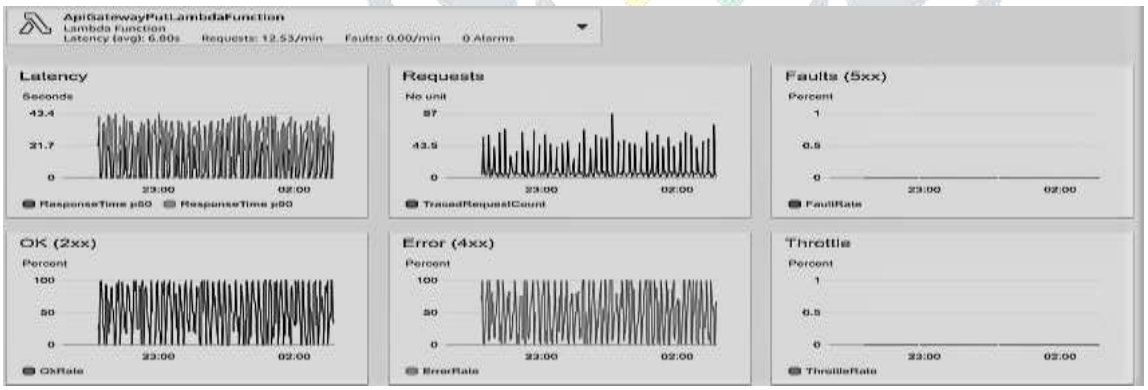

Fig- 2. AWS API gateway monitoring

VI. COMPARISON OF AMAZON AWS API GATEWAYS AND OTHER API GATEWAYS

6.1 Comparison between Amazon Aws Api gateway and Azure Api gateway-

Azure and AWS provide similar services,Azure supports hybrid and multi cloud environments with a single place to manage all our APIs.Both are good tools for deploying, administering, and monitoring APIs.Authentication, CORS, rate restriction and quotas, caching, transformation, logging, mocking and many other services are available.Azure api gateway is more suitable if we are using Microsoft Azure and has good support for Azure functions.

6.2 Comparison between Amazon Aws Api gateway and Kong-

Kong is an open source api gateway whereas Amazon Aws api gateway is not an open source.Kong is an on-premise,Aws Api is a cloud hosted.On-premise deployments can take longer to plan and maintain than cloud deployments. However, due to the extra hop, cloud hosted solutions might add delay and degrade service availability if the vendor goes down.Kong has a wide

choice of installation options, including pre-made containers like Docker and Vagrant, so we can get a deployment up and running quickly.When dealing with hundreds of microservices that compose our API and various types of consumers calling our APIs, Kong's concept of services, routes, and consumers has plenty of flexibility. This provides us to associate plugins and transformations with a certain route or even a single consumer.

6.3 Comparison between Amazon Aws Api gateway and Tyk.io-

Tyk.io is an open source api gateway.Tyk is available as a cloud-hosted SaaS solution or as an on-premise solution.we can deploy instances on Heroku or AWS.Key management, quotas, rate limits, API version, and access control are all available in Tyk, however there are no integrated billing options. To perform administrative functions, Tyk provides both a REST API and a web dashboard. While they do offer an extension list, Tyk does not have the same level of community or plugin hub as Kong. They do, however, keep their gateway well-designed and try to keep it as lean as possible.

VII. SUMMARY

The paper talks about how easy it is to make use of Amazon API Gateway and create and manage a simple REST API. Amazon AWS is a complete solution which provides various tools for all our cloud needs. It provides us with many tools with well prepared documentations to use to create our own cloud applications. Amazon AWS Gateway is another such tool using which we can easily build and deploy REST APIs. AWS tools are highly configurable and compatible with each other, hence by using those tools effectively and efficiently we may create and manage our APIs. By making use of AWS DynamoDB, AWS Lambda, AWS IAM and AWS API Gateway we can create a simple API using which we can perform all the CRUD operations on a DynamoDB and manage the created API . The paper describes how easy it is to do so and describes the architecture and comparison of the API Gateway with other services provided by different cloud service providers.

REFERENCES

[1] Amazon AWS API gateway. http://docs.aws.amazon.com/apigateway
[2] RESTdesc website, http://restdesc.org
[3] J. Kopecky , K. Gomadam, T.Vitvar: hRESTS: an HTML Mi-croformat for Describing RESTful Web Services.
Proceedings of the 2008 IEEE/WIC/ACM International Conference on WebIntelligence (WI-08), 2008, pp. 619-625.
[4] Lathem, K. Gomadam, and A.P. Sheth, "'SA-REST and(S)mashups: adding semantics to RESTful services,"' in Proc.of the Int. Conf. on Semantic Computing 2007 (ICSC), IEEE2007, pp. 469-476.
[5] J. Kopecky, T. Vitvar, D. Fensel, K. Gomadam: hRESTSe MicroWSMO. Technical report, available at http://cms-wg.sti2.org/TR/d12/, 2009.
[6] D. Renzel, P. Schlebusch, and R. Klamma, "Today‟s top „RESTful‟ services and why they are not RESTful", WISE, 2012.
[7] F. Petrillo, P. Merle, N. Moha, and Y.G. Guéhéneuc, "Are REST APIs for Cloud Computing Well-Designed? An Exploratory Study." ICSOC
2016, Springer International Publishing, 2016.
[8] Rodríguez, Carlos, et al. "REST APIs: A Large-Scale Analysis of Compliance with Principles and Best Practices." International Conference
on Web Engineering, Springer, 2016.
[9] F. Haupt, D. Karastoyanova, F. Leymann, and B. Schroth, "A model driven approach for REST compliant services", ICWS, 2014.
[10] F. Haupt, F. Leymann, and C. Pautasso. "A conversation based approach for modeling REST APIs." WICSA 2015 , IEEE, 2015.