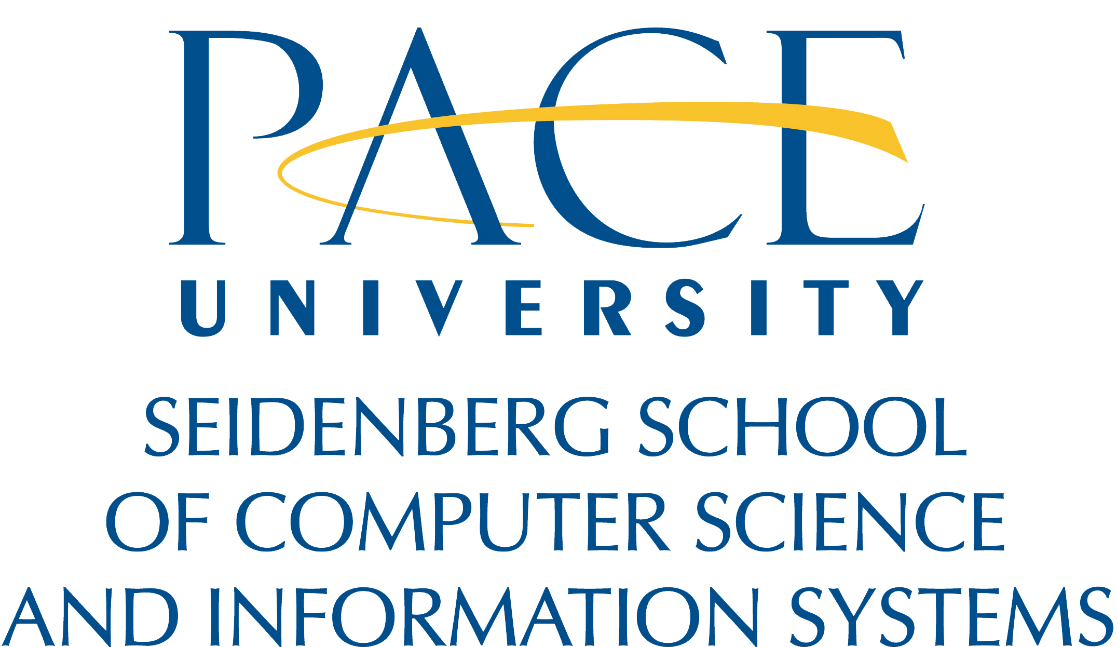
****

cs – 610 final report file   
amazon api gateway

Submitted to – dr. altion simo

Pace university, new york

ishan anand (U01506654)

cs-

12/09/2018

**ABSTRACT**

Amazon API Gateway is a subsidiary of Amazon Web Services (AWS) that provides [on-demand](https://en.wikipedia.org/wiki/Software_as_a_service) [cloud computing](https://en.wikipedia.org/wiki/Cloud_computing) [platforms](https://en.wikipedia.org/wiki/Computing_platform) to individuals, companies and governments, on a paid subscription basis.

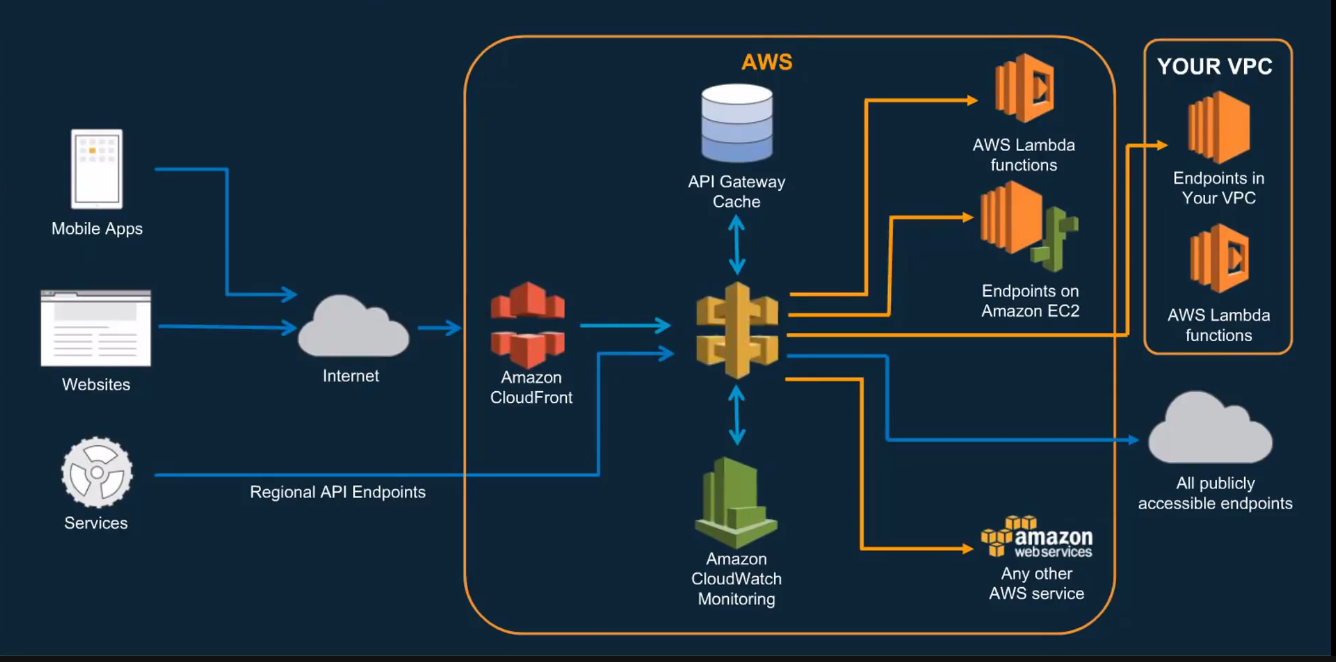
Amazon API Gateway helps deliver strong developer backend. The API Gateway is robust, secure and also helps developer to securely connect mobile and web applications that run on AWS platform as well as outside the AWS. Also, API Gateway provides support to several backend services like Lambda, EC2, etc. that can help in developing Micro-Services in an Integrated environment.

**INTRODUCTION TO AMAZON API GATEWAY**

Amazon API Gateway is an AWS service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. You can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud.

**Fig:1 – Functionality of AWS API Gateway**

API Gateway can be considered a backplane in the cloud to connect AWS services and other public or private websites.



**Fig:2 - The following diagram shows API Gateway architecture**

# **BUILDING API-DRIVEN MICRO-SERVICES WITH AMAZON API GATEWAY**

Micro-services is a software architectural style in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. These services are small, highly decoupled and focus on doing a small task, facilitating a modular approach to system building.

Micro services can vary depending upon the use-case, technology and the overall problem for which the project is being created to solve in regard to business.

**BASIC API TECHNOLOGY STACK**

The technology stack of an API consist of presentation layer (front-end) that includes accessing the API and also, it includes a backend from where actual technology represents business applications, code, databases and data stores.

AWS API supports number of backend in its service. Some of the backend services includes:

* AWS Lambda
* Endpoints on Amazon EC2, whether it is an actual EC2 instances or container based applications that can put any HTTP or HTTPS endpoint of any choice
* VPC – Backend services od API gateway also includes VPC that may include endpoints in VPC or the Lambda functions
* All publicly accessed endpoints.

**SETTING UP LAMBDA AS AN INTEGRATED** **PLATFORM IN API GATEWAY TO DEPLOY SERVERLESS API**

# AWS Lambda is an [event-driven](https://en.wikipedia.org/wiki/Event-driven_programming), [server-less computing](https://en.wikipedia.org/wiki/Serverless_computing) platform provided by [Amazon](https://en.wikipedia.org/wiki/Amazon.com) as a part of the [Amazon Web Services](https://en.wikipedia.org/wiki/Amazon_Web_Services). The purpose of Lambda, as compared to [AWS EC2](https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud), is to simplify building smaller, on-demand applications that are responsive to events and new information.

# To build an API with Lambda integrations, you can use either the Lambda proxy integration or the Lambda custom integration.

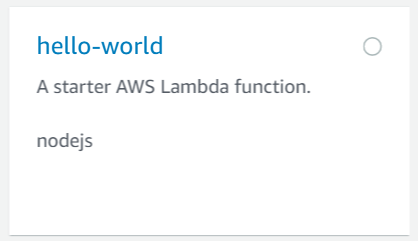
The Anatomy of the Lambda function is as follows:

**Fig:3 - The following diagram shows Anatomy of Lambda Function**

**Create the backend Lambda function**

To create the randomNumber Lambda function for Lambda custom integration:

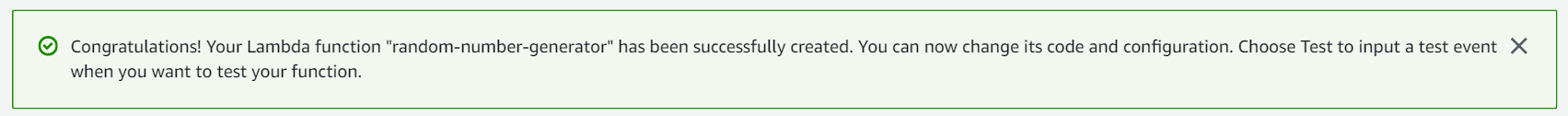
1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create a function.
3. Choose Blueprints.
4. In the blueprints pane, select the Hello World function.



**Fig:4 – Selecting Hello World Function**

1. Select Configure.
2. Create a function name, following by Role.
3. Create function.

After this, a confirmation like this will appear:

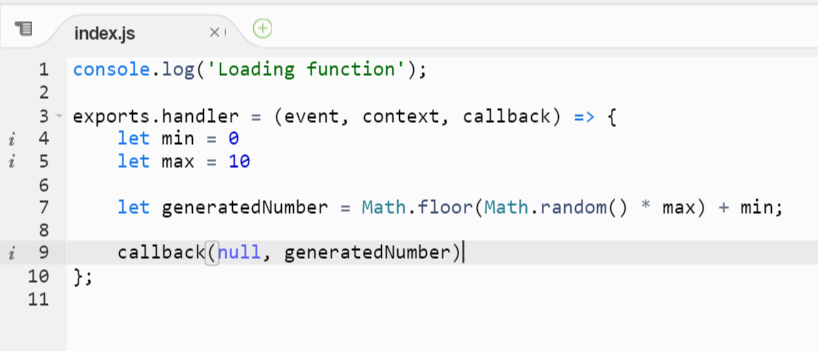


**Fig:5 – Confirmation to build Lambda Function**

**CODING THE BACKEND LAMBDA FUNCTION**

To code the randomNumber Lambda function for Lambda custom integration:

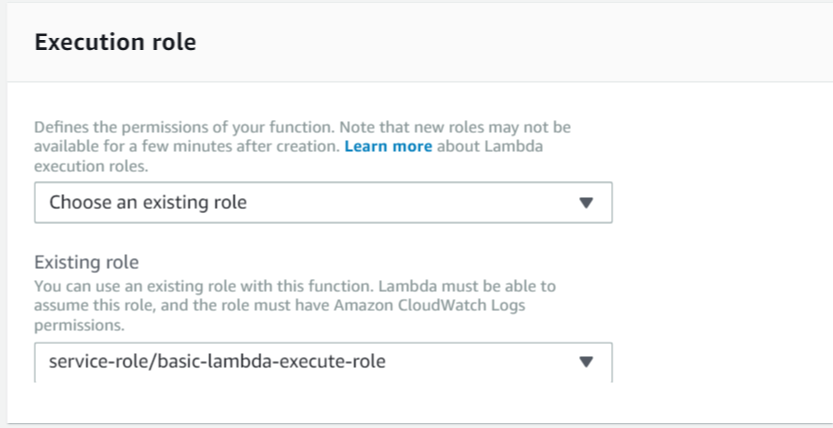
1. In the function code, set code entry type to edit code inline.
2. Set runtime to Node.js.8.10
3. Set Handler name to index.handler
4. Now in the environment, type the following code by replacing the code for index.js

  
**Fig:6 – Java-script code used to build Lambda function**

**TESTING THE FUNCTION FOR BACKEND LAMBDA FUNCTION**

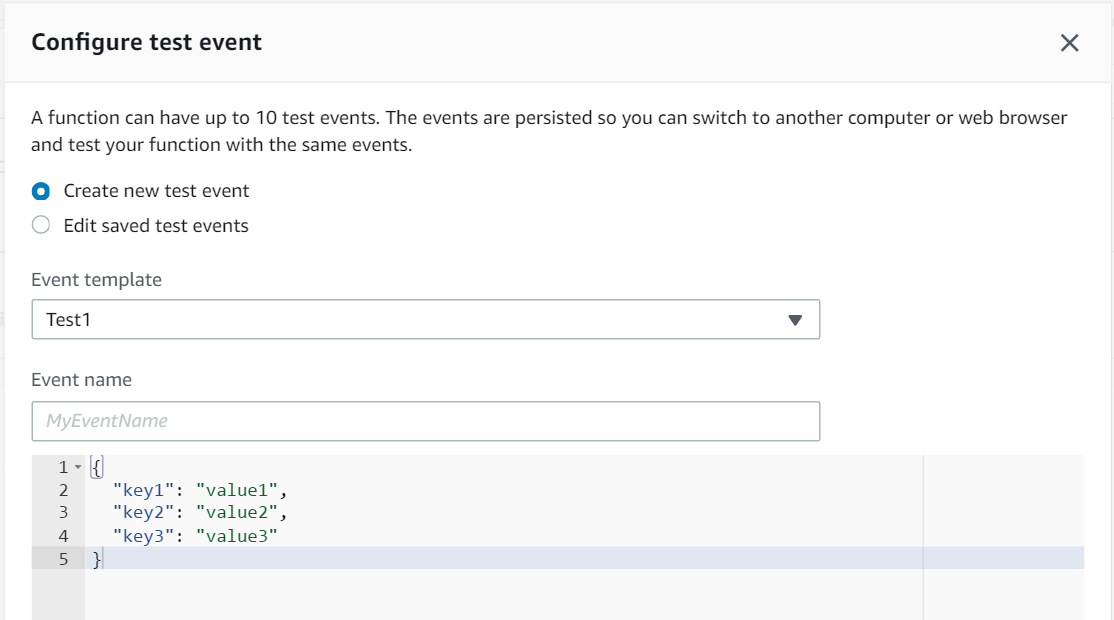
To set up the environment and testing the function:

1. For Runtime, choose Node.js 8.10.
2. Create an execution role for the function.



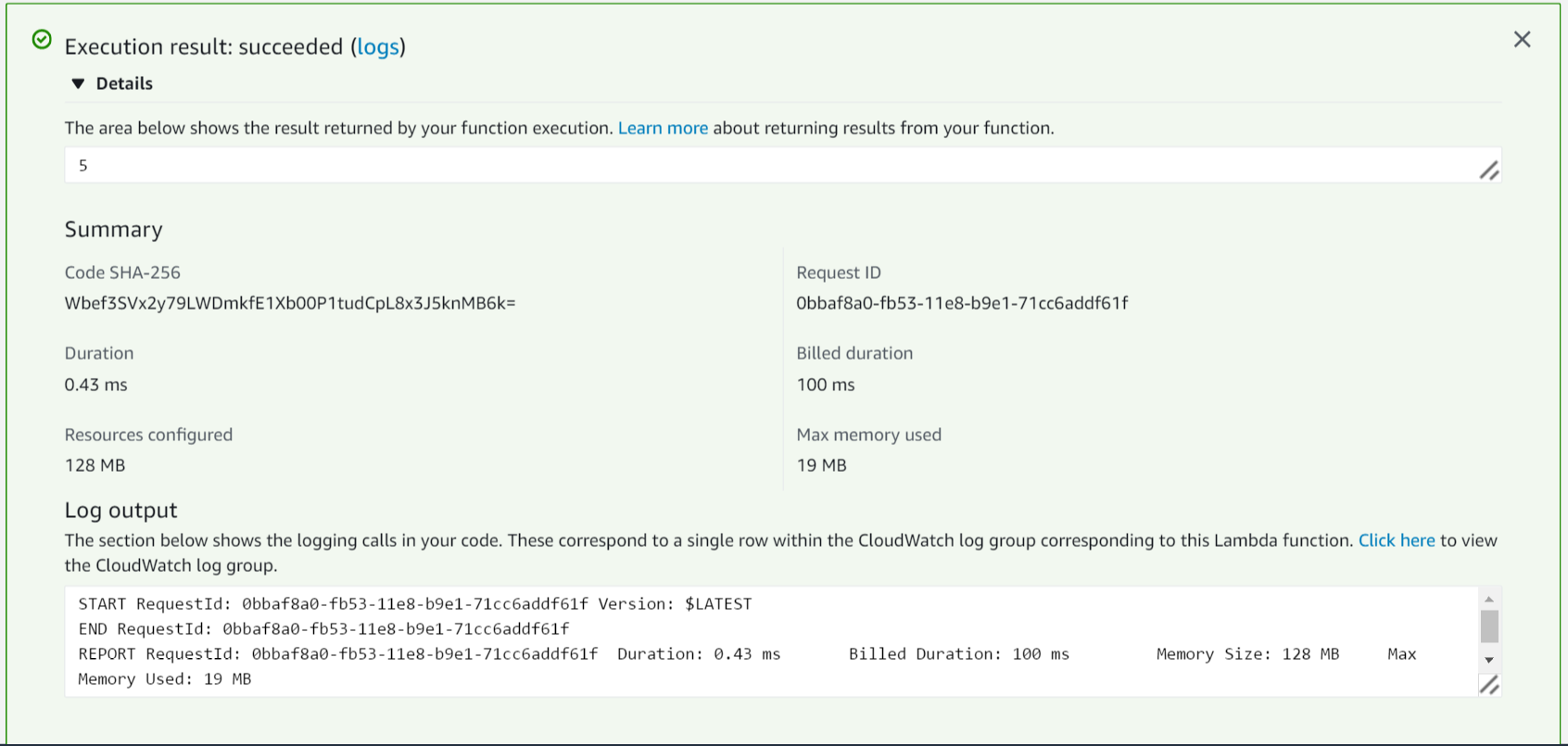
**Fig:7 – Selecting Execution role**

1. Save the changes.
2. After this, in order to test the Function, Select Test.
3. In order to configure Test events, select Create new Test event.
4. Select event template.
5. Select event name and proceed.



**Fig:8 – Configuring Test Events**

1. After proceeding the course, the function will start to compile and will give an output like this:



**Fig:9 – Output of the Lambda function**

**CONNECTING LAMBDA TO API GATEWAY**

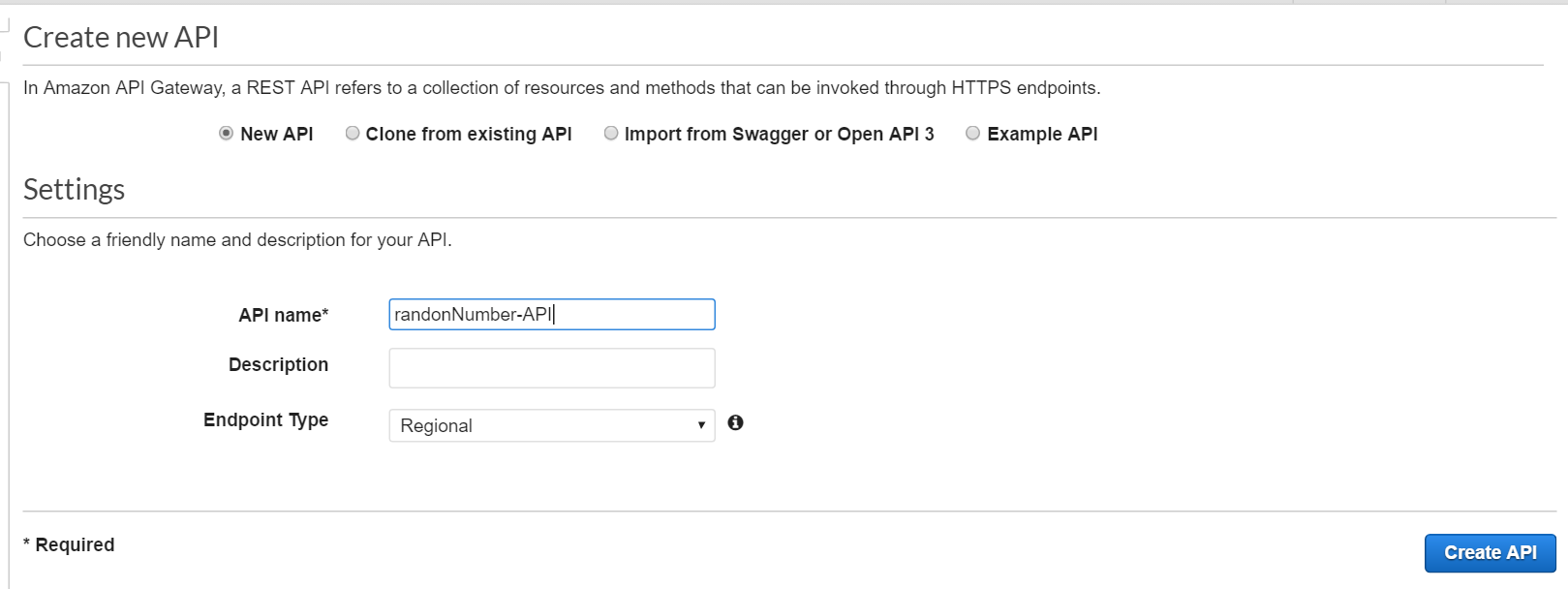
The process to connect lambda to the API gateway consists of the following two steps. Any of these steps can be used to connect Lambda to the API Gateway. The following steps include:

* Using Triggers from the Lambda Framework
* Creating API in the API Gateway

**Using API Gateway to Import Lambda Function**

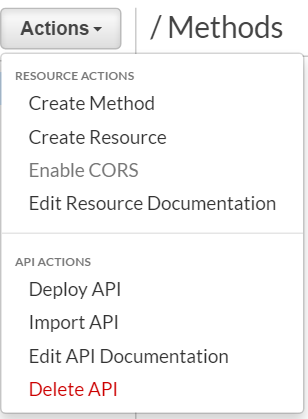
Using the API Gateway console to build an API that enables a client to call Lambda functions through the Lambda custom integration. To create an API, choose Create new API (for creating the ﬁrst API) or Create API (for creating any subsequent API). Next, do the following:

1. Choose New API.
2. Type a name in API Name.
3. Choose Create API.



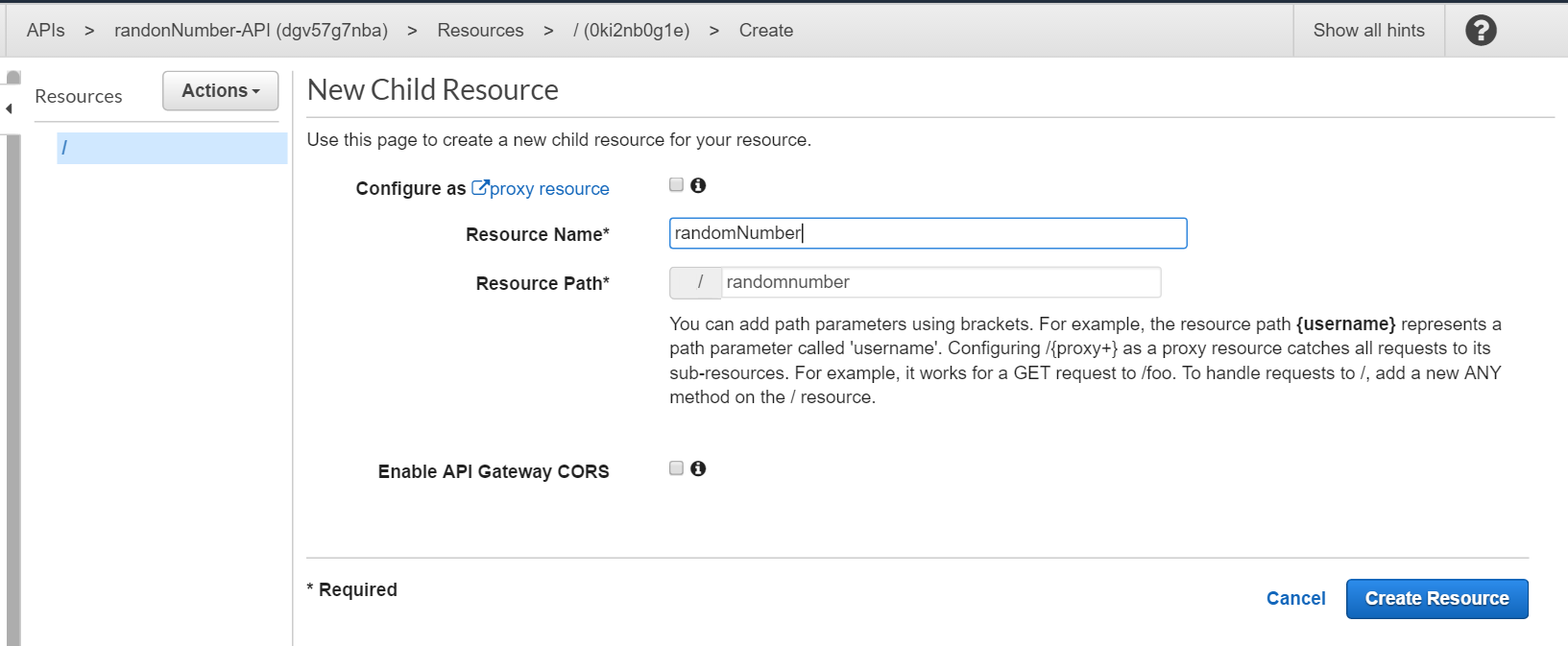
**Fig:10 – Creating new API**

1. Go to Actions and create a new Resource.



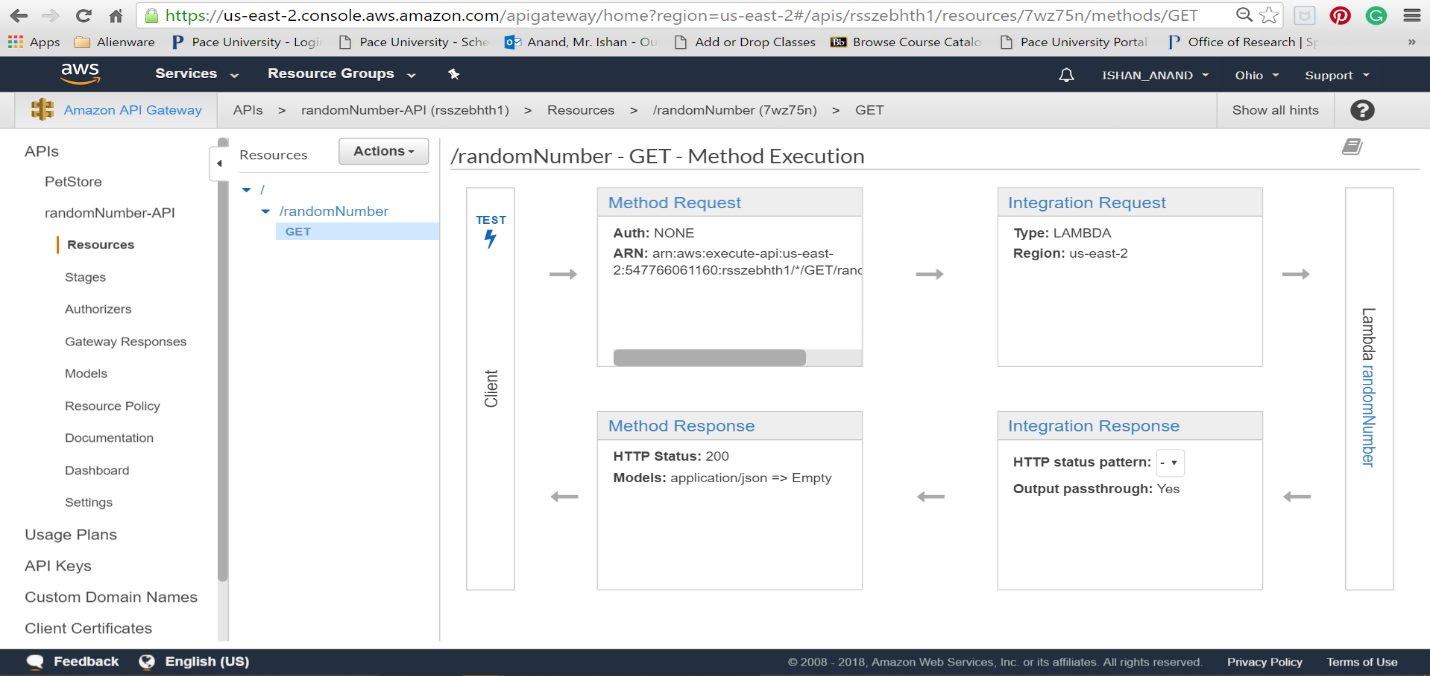
**Fig:11 – Action’s pane to create Resources and Methods**

1. Name that resource
2. Create Resource



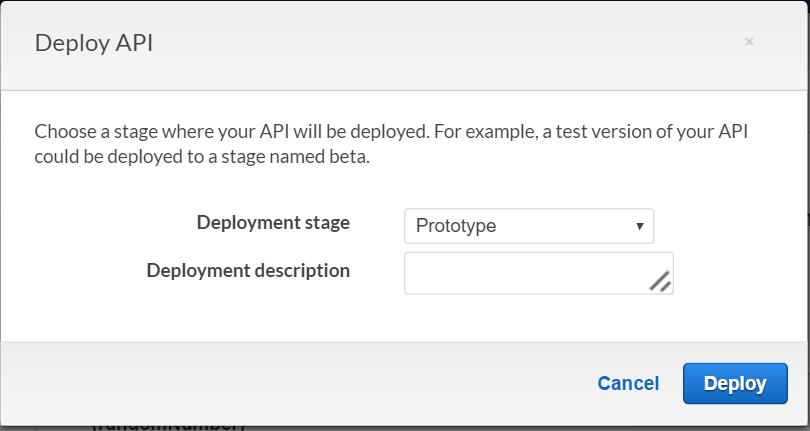
**Fig:12 – Creating child resources**

1. Now Create a method under the resource and make a GET method.
2. Set Integration type to Lambda Function
3. Click to add permissions
4. Create a new Stage
5. Deploy



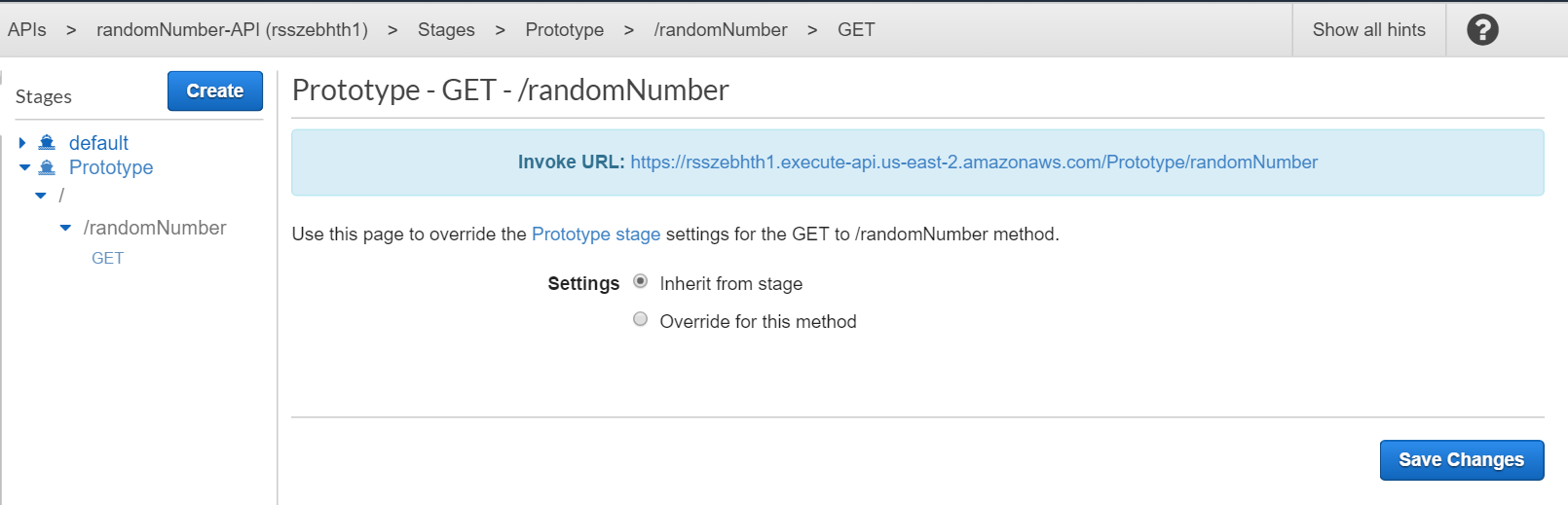
**Fig:13 – GET Method Pane**

1. Go to actions and deploy API
2. Deploy API by selecting Deployment Stage.



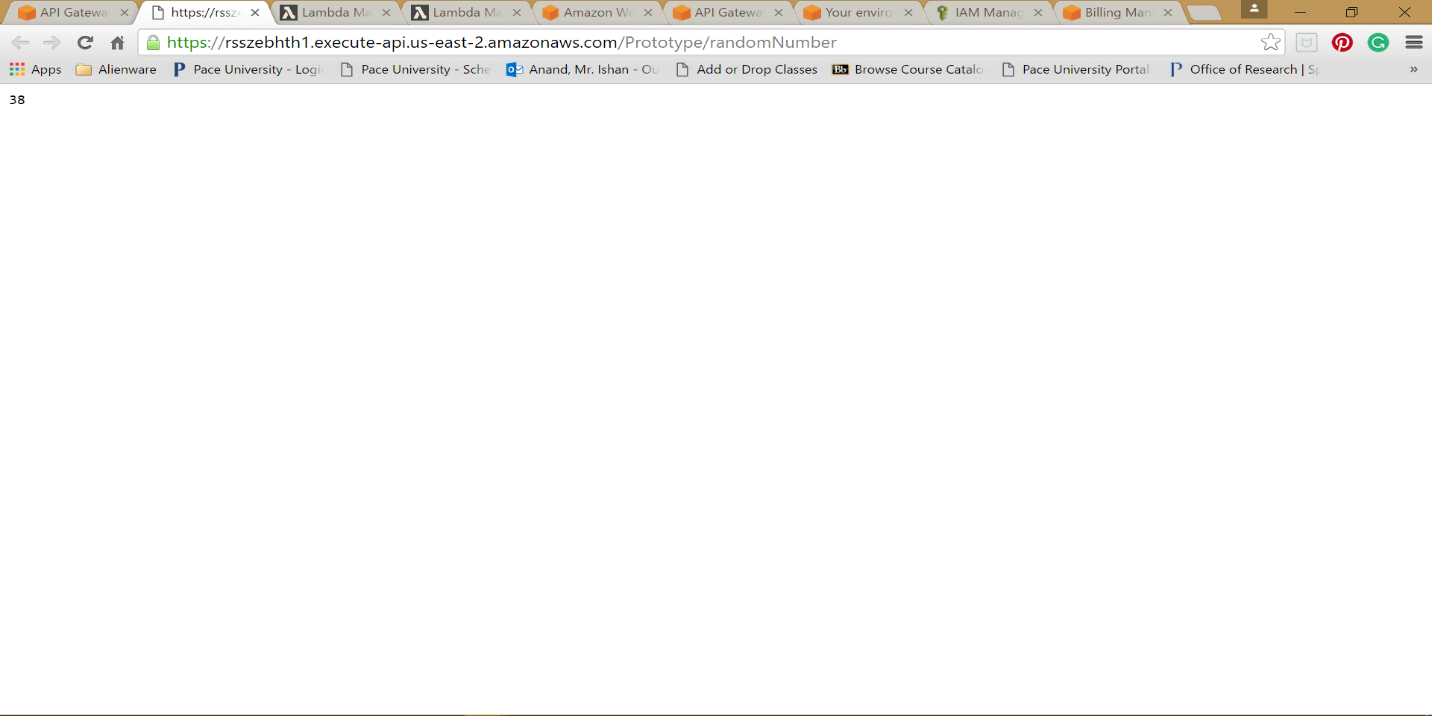
**Fig:14 – Test stages for Deploying API**

Go to GET method and click on the URL in order to check the deployed API.



**Fig:15 – Invoked URL for GET Function**

1. Now, the Output of the Lambda function with Integration with API will look like this:



**Fig:16 – API based output to the Lambda function in API Gateway**

**BENEFITS OF API GATEWAY**

API Gateway helps you deliver robust, secure, and scalable mobile and web application backends. API Gateway allows you to securely connect mobile and web applications to business logic hosted on AWS Lambda, APIs hosted on Amazon EC2, or other publicly addressable web services hosted inside or outside of AWS. With API Gateway, you can create and operate APIs for backend services. For example, you don't need to develop and maintain infrastructure to handle authorization and access control, traﬃc management, monitoring and analytics, version management, and software development kit (SDK) generation.

API Gateway is designed for web and mobile developers who want to provide secure, reliable access to backend APIs for access from mobile apps, web apps, and server apps that are built internally or by third-party ecosystem partners. The business logic behind the APIs can be provided by a publicly accessible endpoint that API Gateway proxies call, or it can be entirely run as a Lambda function.

**DEVELOPER EXPERIENCES**

There are two kinds of developers who use API Gateway: app developers and API developers.

1. An app developer builds a functioning application to call AWS services by invoking API methods in API Gateway.
2. An API developer creates and deploys an API to enable the required functionality in API Gateway. The API developer must be an IAM user in the AWS account that owns the API.

The app developer does not need to have an AWS account, provided that the API either does not require IAM permissions or supports authorization of users through third-party identity providers supported by Amazon Cognito identity federation. Such identity providers include Amazon, Amazon Cognito User Pools, Facebook, and Google. Creating and Managing an API Gateway API

An API developer works with the API Gateway service component for API management, to create, conﬁgure, and deploy an API. Each API includes a set of resources and methods. A resource is a logical entity that an app can access through a resource path.

For example, /incomes could be the path of a resource representing the income of the app user. A resource can have one or more operations that are deﬁned by appropriate HTTP verbs such as GET, POST, PUT, PATCH, and DELETE. A combination of a resource path and an operation identiﬁes a method of the API. For example, a POST /incomes method could add an income earned by the caller, and a GET / expenses method could query the reported expenses incurred by the caller. A method corresponds to a REST API request that is submitted by the user of your API and the response returned to the user. The app does not need to know where the requested data is stored and fetched from on the backend. The API interfaces with the backend by means of integration requests and integration responses.

For example, with DynamoDB as the backend, the API developer sets up the integration request to forward the incoming method request to the chosen backend. The setup includes speciﬁcations of an appropriate DynamoDB action, required IAM role and policies, and required input data transformation. The backend returns the result to API Gateway as an integration response. To route the integration response to an appropriate method response (of a given HTTP status code) to the client, you can conﬁgure the integration response to map required response parameters from integration to method. You then translate the output data format of the backend to that of the frontend, if necessary. API Gateway enables you to deﬁne a schema or model for the payload to facilitate setting up the body mapping template.

As an API developer, you can create and manage an API by using the API Gateway console, described in Getting Started with Amazon API Gateway, or by calling the API Gateway REST API. There are several ways to call this API. They include using the AWS Command-Line Interface (CLI), or by using an AWS SDK. You can also use a REST API client, such as Postman, to make raw API calls. In addition, you can enable API creation with AWS CloudFormation templates or API Gateway Extensions to OpenAPI . For a list of regions where API Gateway is available, as well as the associated control service endpoints, see Regions and Endpoints. Calling an API Gateway API

An app developer works with the API Gateway service component for API execution, named executeapi, to invoke an API that was created or deployed in API Gateway. The underlying programming entities are exposed by the created API. There are several ways to call such an API. You can use the API Gateway console to test invoking the API. You can use a REST API client, such as CURL or Postman, or an SDK generated by API Gateway for the API to invoke the API. Be aware of the diﬀerences between the apigateway and execute-api API Gateway service components. Reference the appropriate service component name when you select one while, for example, setting IAM permission policies for building or calling an API.

# Citations

Amazon. (2018, 04). *Create a Lambda Function with the Console*. Retrieved from docs.aws.amazon.com: https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html

*Amazon-API Gatewasy*. (n.d.). Retrieved from aws: https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis/f64lo4qjc5/stages/Prototype/resources/~1randomnumber/methods/GET

*api-gateway documentation*. (2018, 09). Retrieved from developerguide: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

AWS. (2018, 06). *Building Lambda Functions with Node.js*. Retrieved from docs.aws.amazon.com: https://docs.aws.amazon.com/lambda/latest/dg/programming-model.html

*aws-Lambda*. (2018, 09). Retrieved from docs.aws.amazon.com: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html

*How API Gateway Works*. (2018). Retrieved from aws.amazon: https://aws.amazon.com/api-gateway/

wikipedia. (2017, 04). *AWS\_Lambda*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/AWS\_Lambda