Documentation for IDGenerator

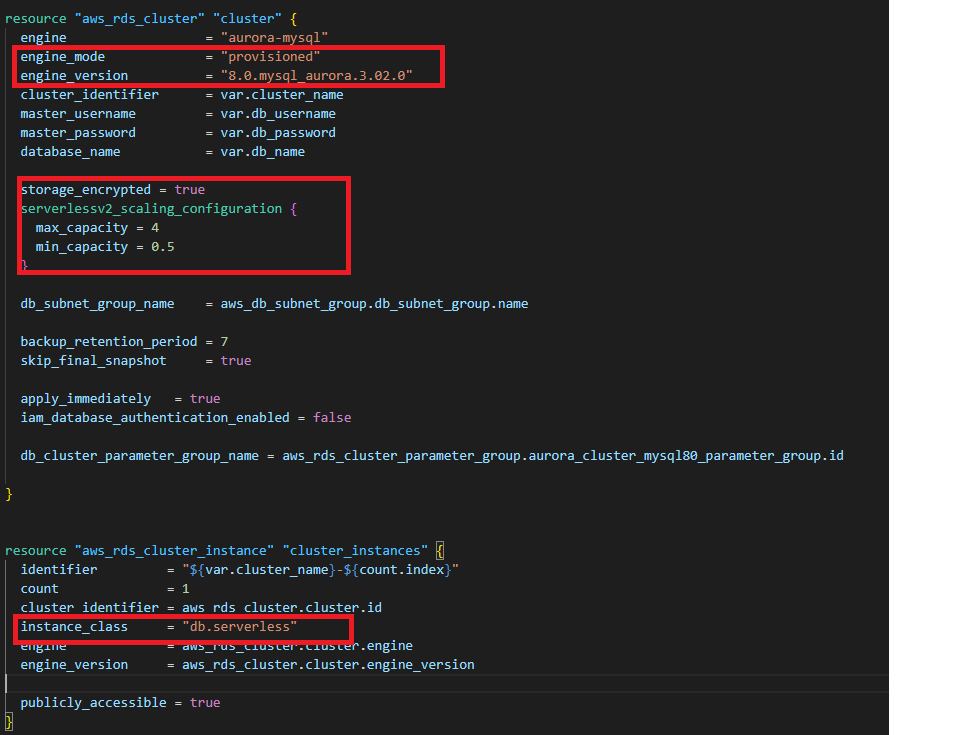
IDGenerator is the AWS Cloud based project which generates different Ids for applications like GuideWire, Surety, Arden and APS.

This document describes the various components of this project. This is followed by an explanation of how the main lambda function of this project works. I also includes the RDS schema that is being created for this project.

# Serverless component

Aws Aurora serverless v2 is an on-demand and autoscaling configuration database. It is highly scalable, cost-efficient, and highly available in different zones. Aurora serverless can start and stop the computation automatically as per requirement and also allocate storage memory efficiently.

Terraform script for Aurora serverless is shown below.



# Database Component

## AWS RDS

AWS RDS is a fully managed AWS database service. It provides cost-efficient, scalable, and manageable relational database. We are using AWS RDS Aurora serverless V2, which is available for Aurora MySQL edition.

# General Component

## AWS VPC

Amazon’s virtual private cloud allows creation of resources for the virtual network and we have complete control over virtual environment for creating subnets, configuration, route tables, and network gateway etc.

## AWS Lambda

AWS Lambda is a serverless computing service, it is used to create lambda functions, self-contained applications that can be written in a language supported by AWS. The execution of the lambda function is efficient and flexible. The lambda function can perform computation tasks, serve web pages, and integrate with other AWS services.

# How Lambda Function works

Lambda function is using 8 different methods like id\_Generate, saveData, assignRange, getData, getDataParams, getDataById, invoke\_lambda, and lambda\_handler.

## id\_Generate

This method generates Id as a sequence number for every application and stores that Id along with application name, sequence type, current date, current time and user information in the SequenceAccountIds and SequencePolicyIds tables according to the sequence type.

## saveData

This method is used to check the Id which was generated most recently by the id\_Generate method. If the Id is out of range then this method deletes that Id from the SequenceAccountIds and SequencePolicyIds tables. If within range it updates the Id in the SequenceAccountIds and SequencePolicyIds tables to keep track of the latest Id generated and to avoid exceeding Id range values.

## assignRange

This method is used to assign new sequence range for SequenceRange table and set status false to the previous range. This method always assigns greater range from the previous range.

## getData

This method gets only latest Id data for all applications from SequenceAccountIds and SequencePolicyIds tables and display that data in the browser.

## getDataParams

This method gets the most recent data for a single application from SequenceAccountIds or SequencePolicyIds table and display that data in the browser.

## getDataById

This method gets data for a specific Id from SequenceAccountIds or SequencePolicyIds table and display that data in the browser.

## invoke\_lambda

This method is checked in the database that SequenceRange table exists or not if table exists in RDS it simply gets all data from the table otherwise it will call another lambda function that will create SequenceRange SequenceAccountIds and SequencePolicyIds tables in RDS and then provide parameters to the other methods that are mention above.

## lambda\_handler

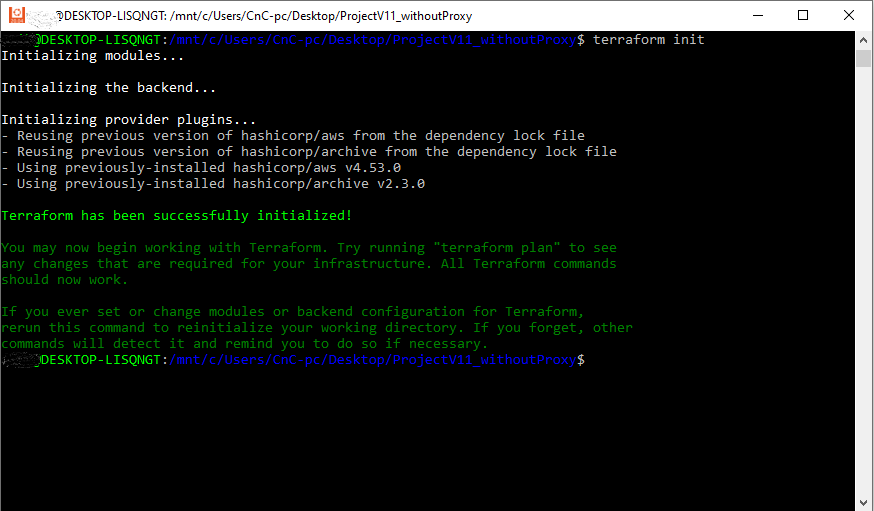
This is the main function of lambda function. It has all information of API request body and request details that is used to invoke lambda.

# How to Deploy the project

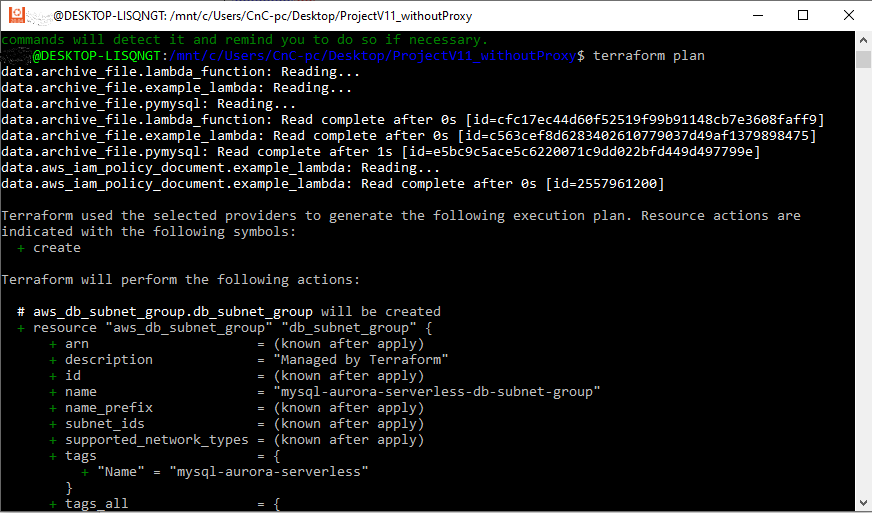
This Project is deployed by Terraform. Terraform is a tool for building, changing, and managing infrastructure in a safe, repeatable way. It is an infrastructure provisioning tool where you can store your cloud infrastructure setup as codes.

The following commands are used to deploy the project:

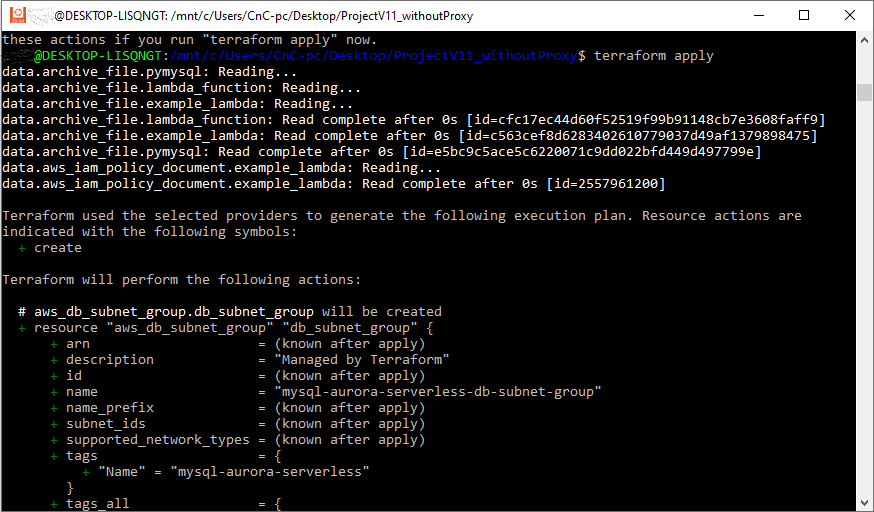
**$ terraform init**

****

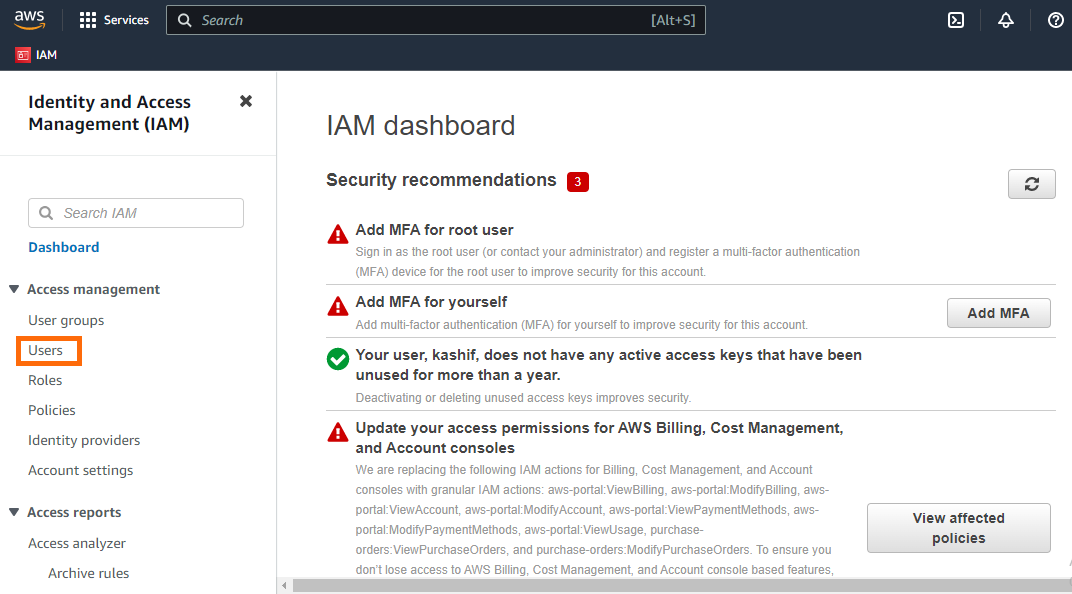
**$ terraform plan**

****

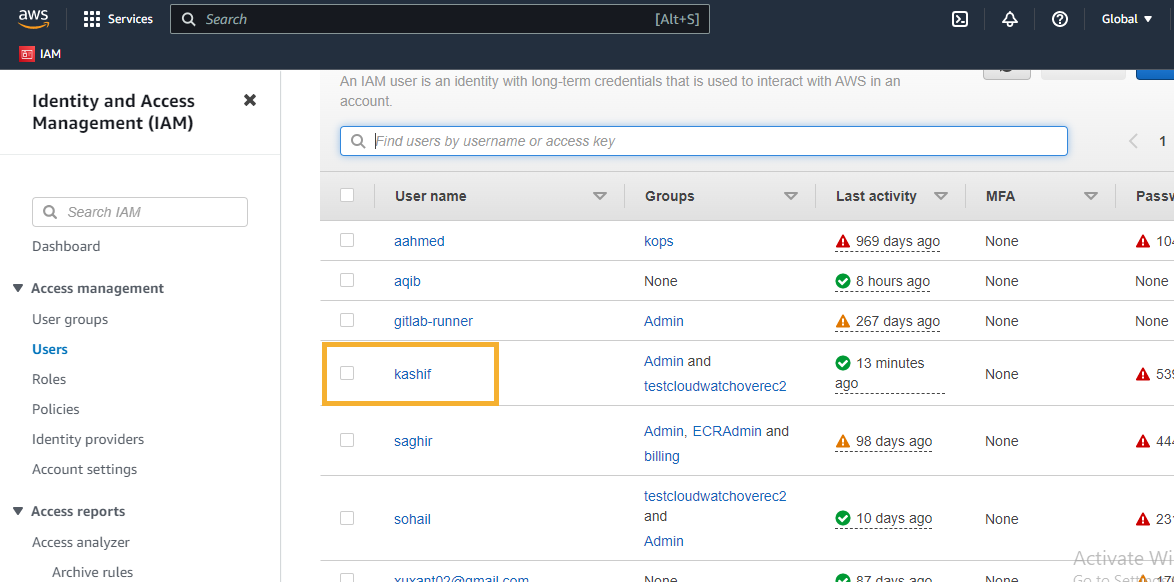
**$ terraform apply**



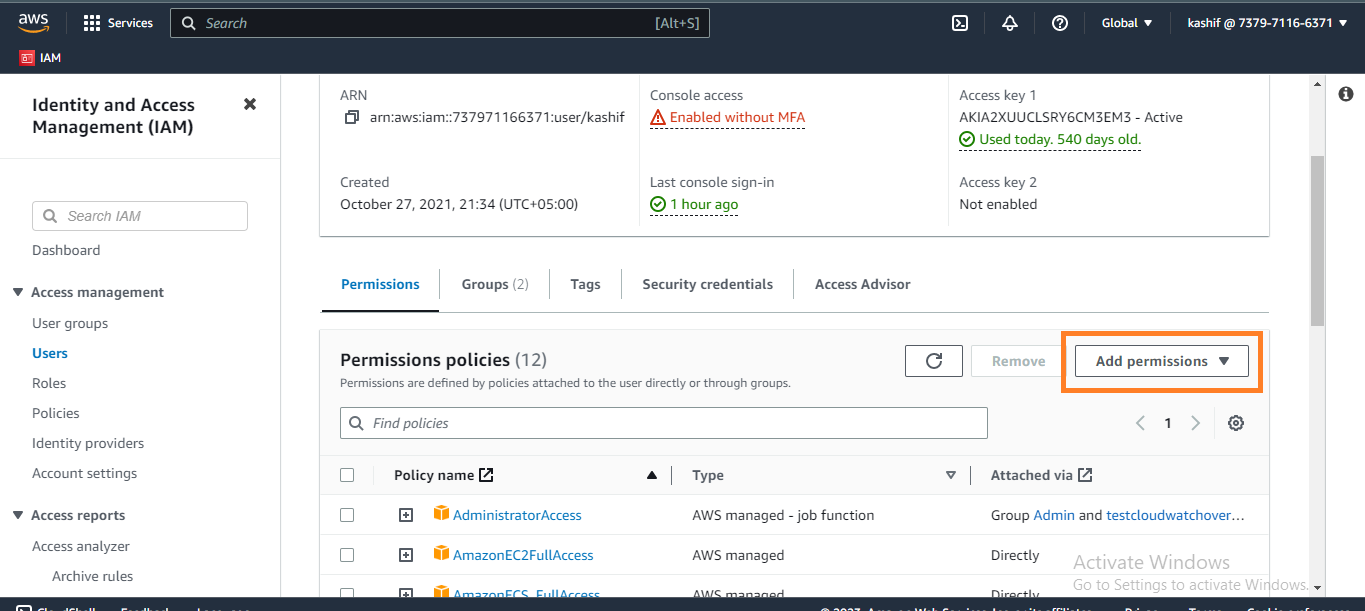
After "**terraform apply**" command completes, Go to Aws IAM Console and select **Users** from left navigation pane.

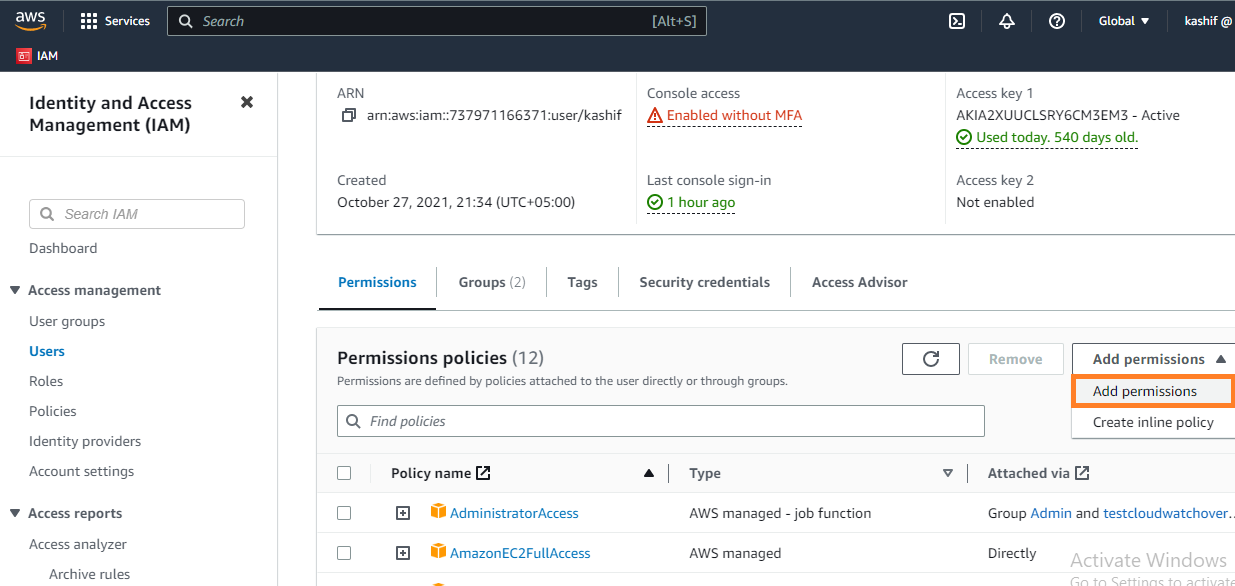


Select your User name in my case user name is kashif.

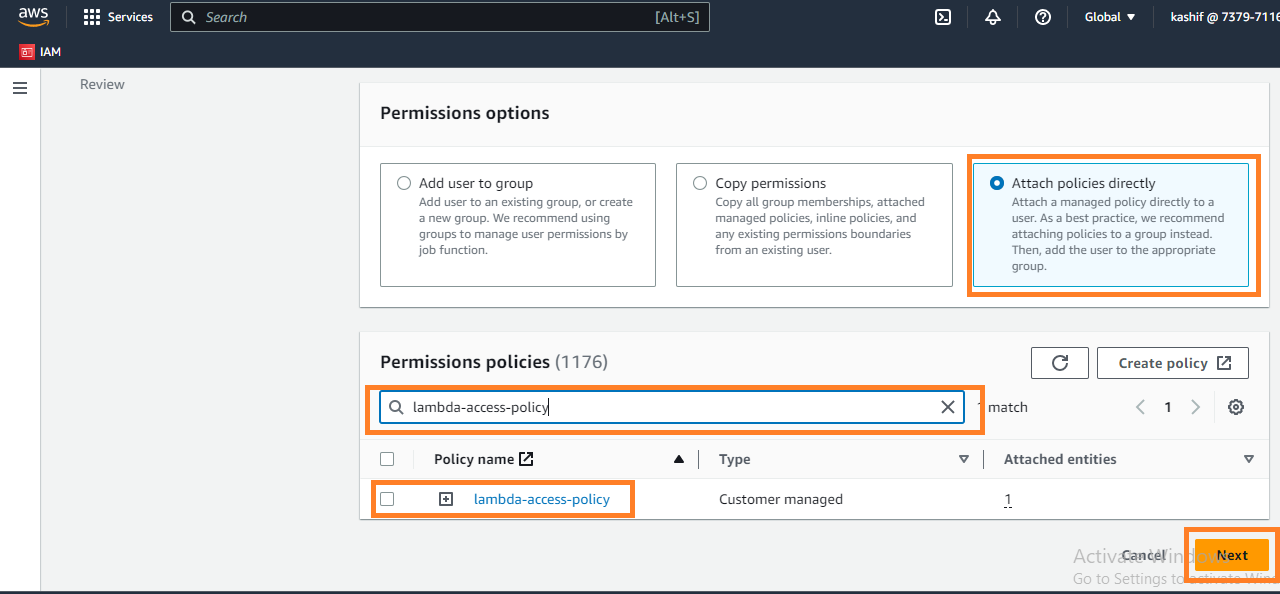


Click on Add Permission dropdown and select Add Permission option.

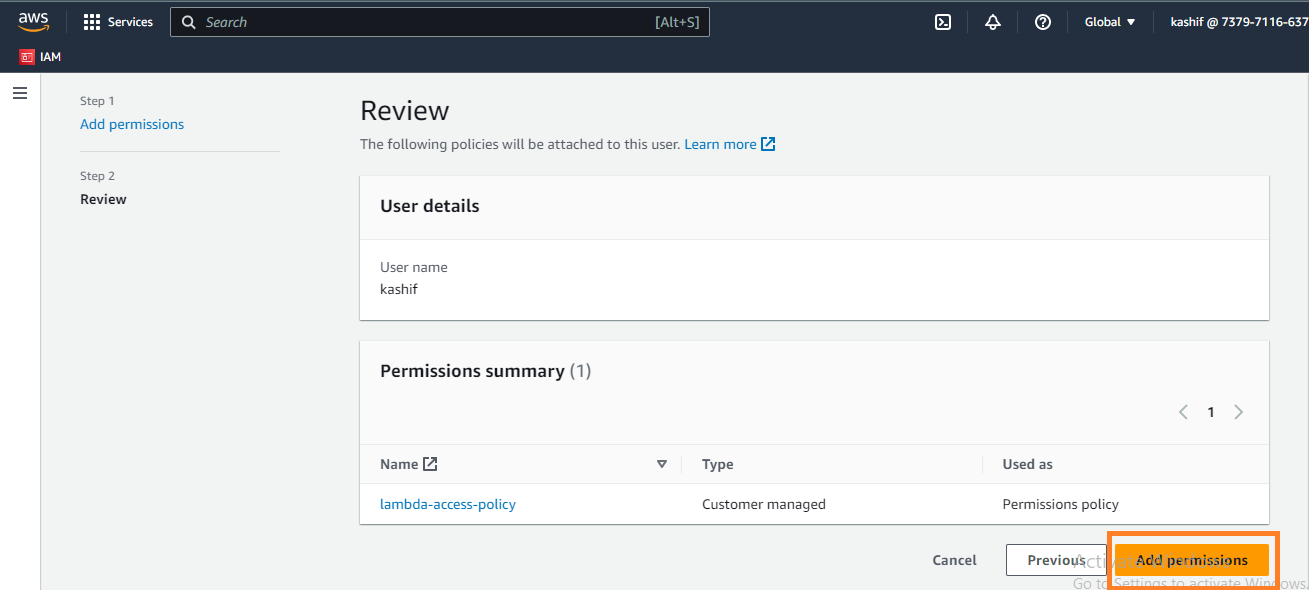




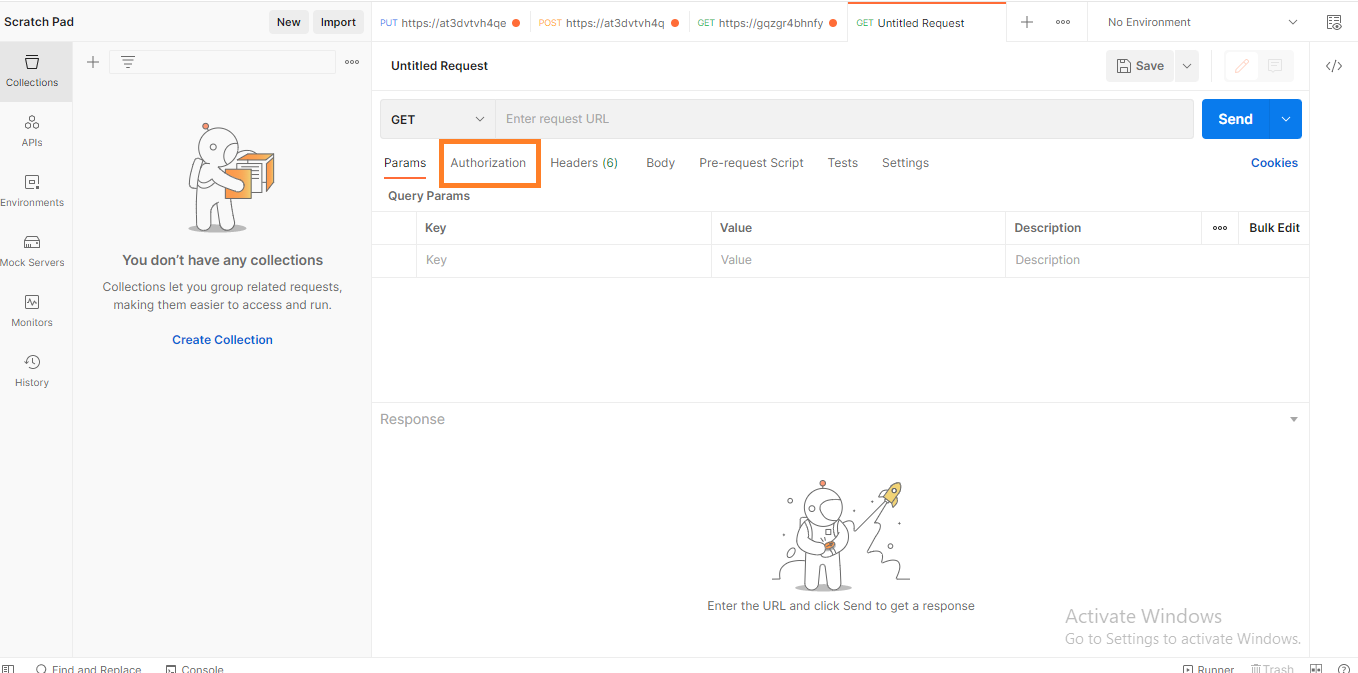
Select option **Attach policies directly,** search **lambda-access-policy** from search bar, Select **lambda-access-policy** and click on **Next** Button.



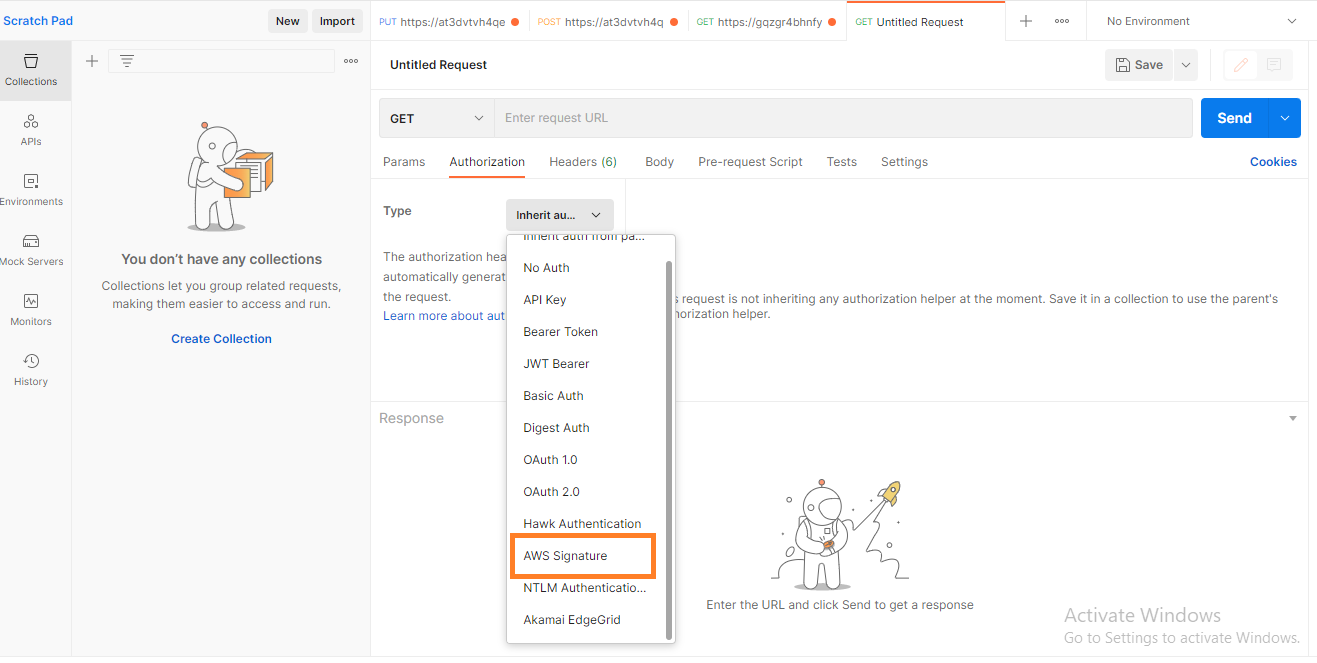
Click on **Add permission** button.



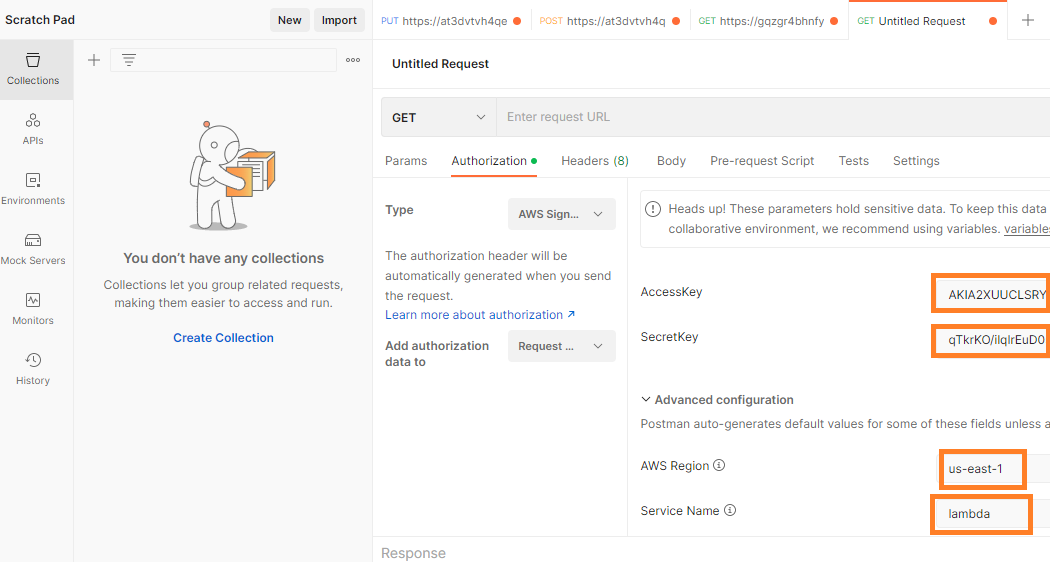
Before call lambda function url, open postman and click on **Authorization** tab.



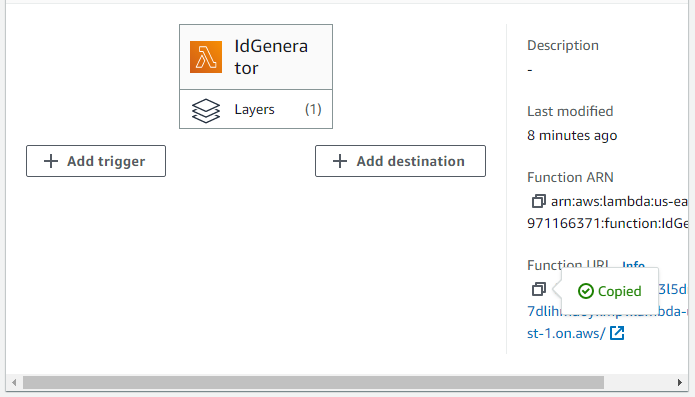
Select authorization type **AWS Signature**.

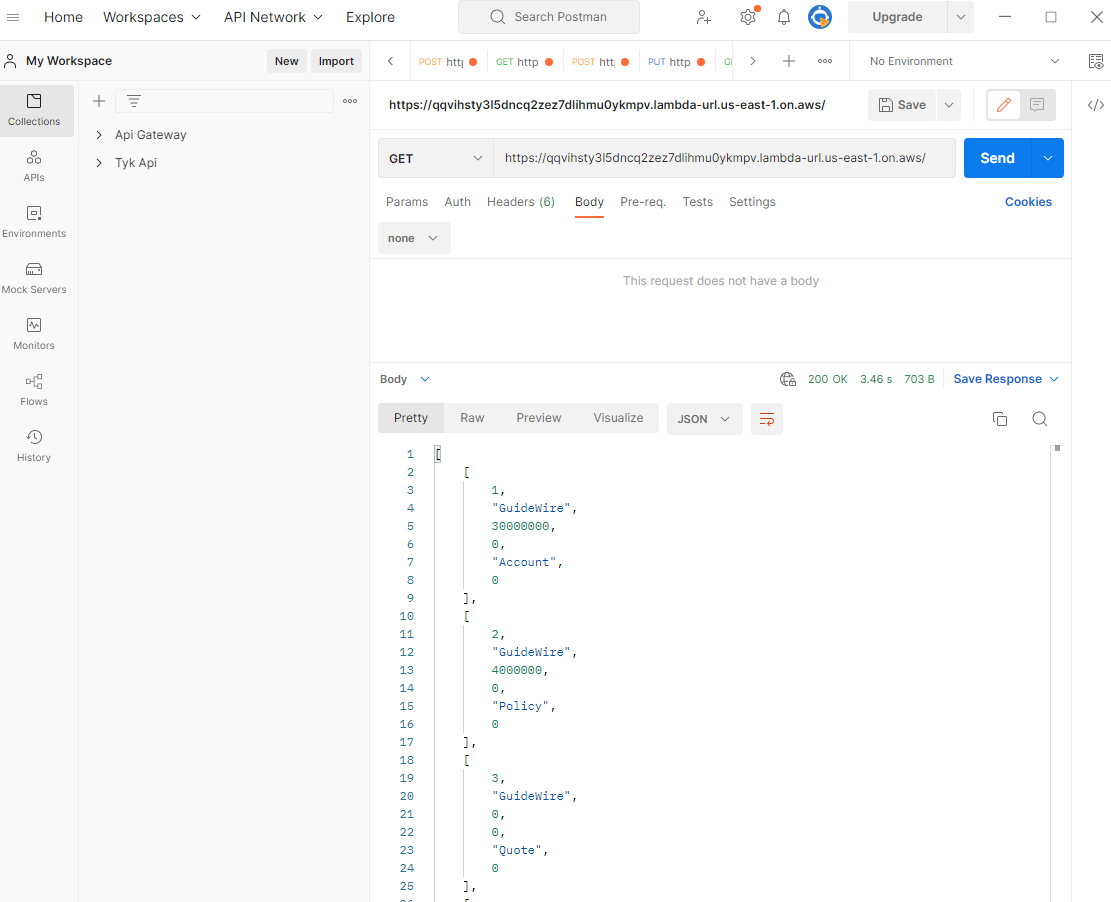


Insert **aws AccessKey**, **SecretKey**, **Region** and **Service Name**.



Now copy lambda function url from lambda function console and paste it in postman and send GET request from postman.

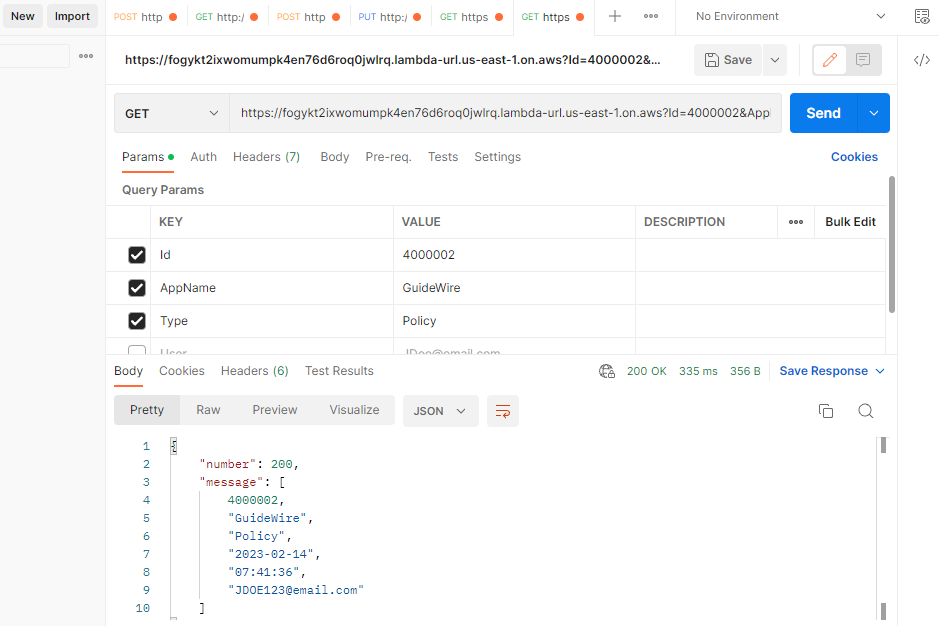




To get data from database based on the **Id, s**end a parametrized **GET** request. Url of the request looks like below:

**https://fogykt2ixwomumpk4en76d6roq0jwlrq.lambda-url.us-east-1.on.aws?Id=4000002&AppName=GuideWire&Type=Policy**

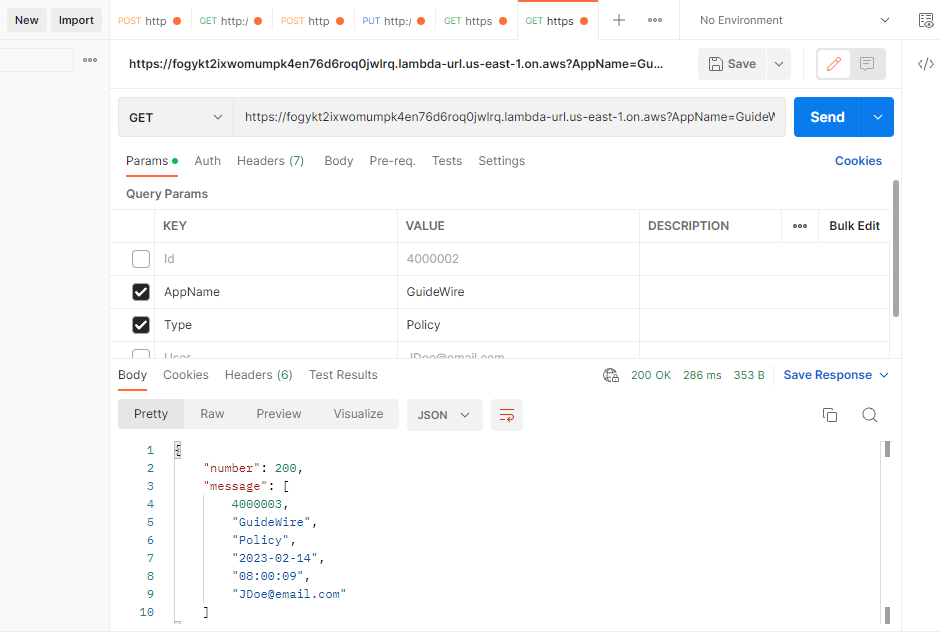
It returns detail of the Id for the AppName and Type provided (if it exists)



To get last record of any application from the database, send a parametrized **GET** request. Url of the request looks like below:

**https://fogykt2ixwomumpk4en76d6roq0jwlrq.lambda-url.us-east-1.on.aws?AppName=GuideWire&Type=Policy**

It returns audit detail of last Id generated for the provided AppName and Type.



In the **POST** method of Lambda function, send parameters in JSON format.

Details of REST API parameters:

**AppName**: This is the application name e.g. **GuideWire**, **Surety**, **Arden** and **APS**.

**Type**: This is Id Type e.g. **Account**, **Policy** and **Quote**.

**User**: This contains user details e.g. email address or any other system info

* Send POST request with the following parameters:

{

“User”:”JDOE123@email.com”,

“AppName”: “GuideWire”,

“Type”: “Account”

}

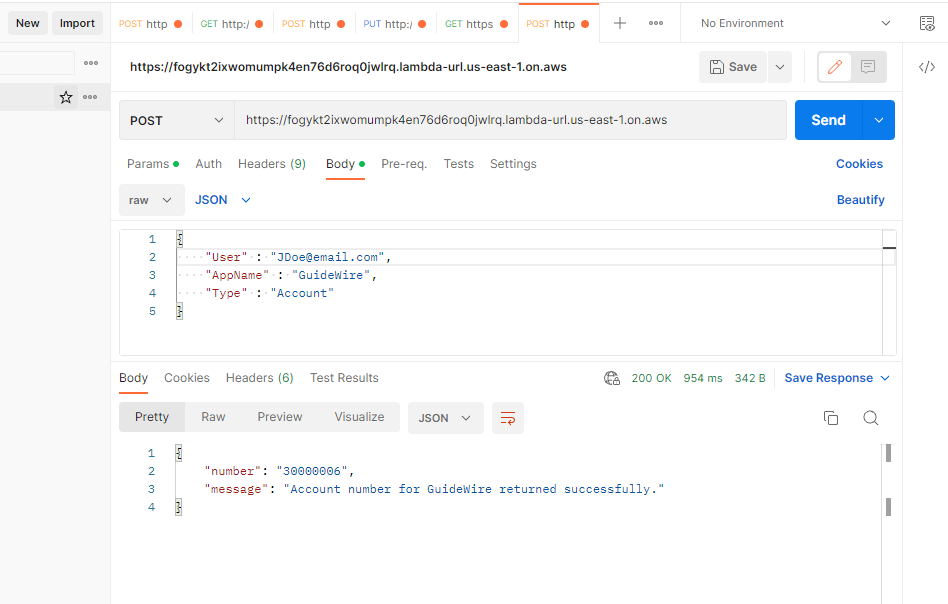
* It request returns data in json format as shown below:

{

“number” : ” 30000006”,

“message”: “Account number for GuideWire returned successfully.”

}



Assign new range for the applications as follow:

Details of PUT request parameters:

**StartRange**: This is the starting point of new sequence range e.g. **80000001**.

**EndRange**: This is the end point of new sequence range e.g. **90000001**.

**UserRole**: This is the name of user role e.g. **Admin**, **HR** and **etc**.

**Type**: This is Id Type e.g. **Account** and **Policy**.

**NOTE:**

Only **Admin** can assign new range other user can not assign.

* Send PUT request with the following json parameters:

{

"StartRange": " 30000001",

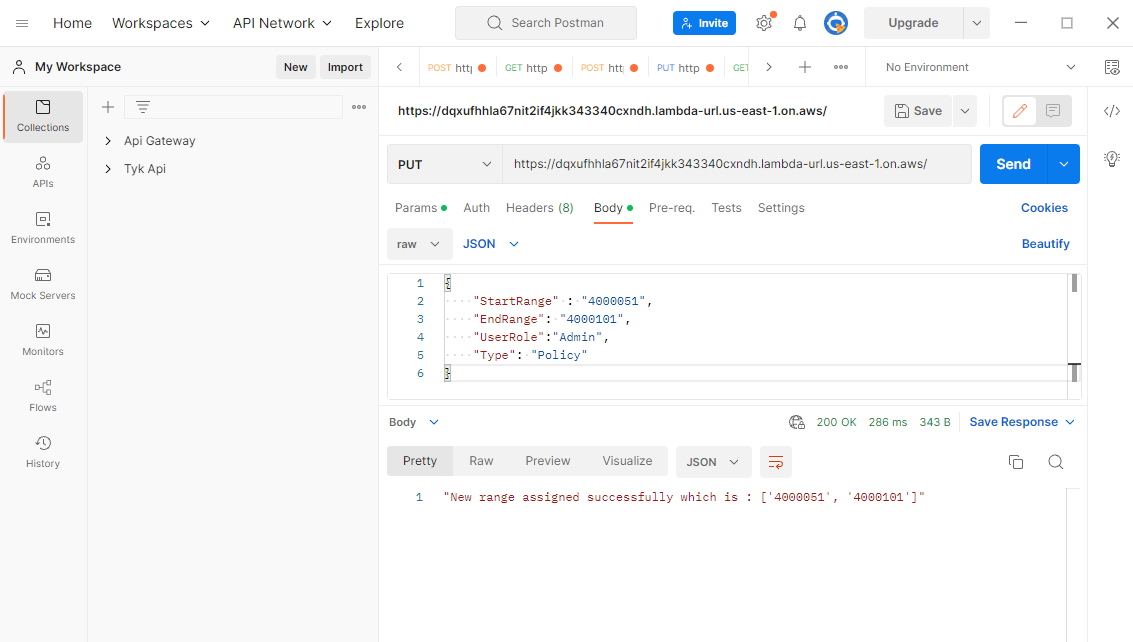
"EndRange": " 80000000",

"UserRole": "Admin",

"Type": "Policy"

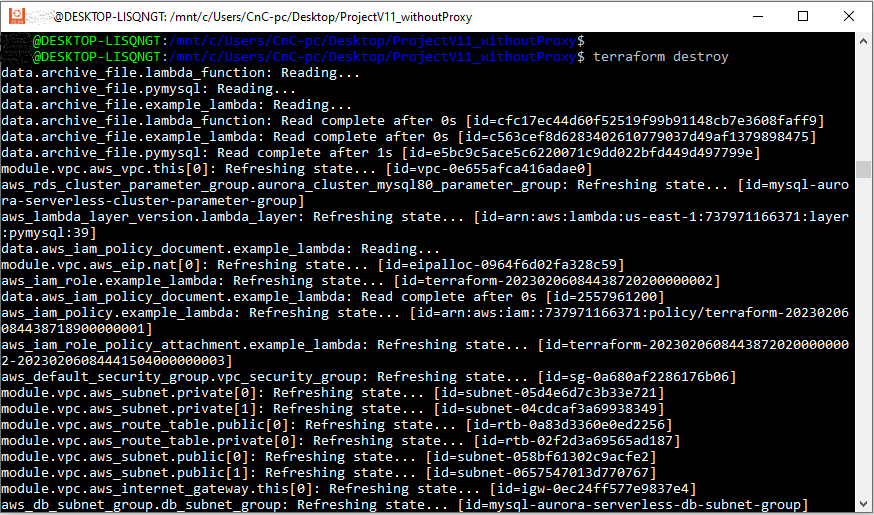
}

* The above request assigns the new range for all applications.



In case complete project is to be deleted, run the following command:

**$ terraform destroy**



**Schema for RDS**

Lambda function creates following 3 tables in the RDS database:

1. SequenceRange
2. SequenceAccountIds
3. SequencePolicyIds

### **SequenceRange**

Columns in the SequenceRange table are ( Id, StartRange, EndRange, SequenceType and Status ) and this table is used to store the starting point, ending point, sequence type and status for all applications in the table.

### **SequenceAccountIds**

Columns in the SequenceAccountIds table are ( Id, AppName, Type, Date, Time and User ) and this table is used to generate a new Id for Account and save application name, sequence type of id, current date, time and user information in that table.

### **SequencePolicyIds**

Columns in the SequencePolicyIds table are ( Id, AppName, Type, Date, Time and User ) and this table is used to generate a new Id for Policy and save application name, sequence type of id, current date, time and user information in that table.