Y19 – RELAY LEARNATHON - DEVELOP A PROJECT USING REACT, PYTHON DJANGO WITH DATABASE CONNECTIVITY AND HOST IT IN AWS

Note: Implementation should only be within the free tier limits.

Structure of the Document

BUILDING OF PROJECT

Abstract of Project Building

Your project should have user registration and signup page connected to the dynamodb (for text data) and s3 bucket (for images). Then after login, it should have minimum 4 functionality pages virtually (since React is a single page app) including home page in which some pages may interact with the python Django based backend connectivity which can communicate to your RDS services for primary database connection.

When ever you are doing with the dynamodb related actions, your fronend react will trigger an API gateway's URL and that will trigger an Lambda microservices (lambda functions), which can have a code to interact with the dynamodb database.

When ever you are connecting with your primary RDS databases, your react application will send an axios request to python application, from which the database related query will be executed.

Frontend Development

Programming Language: React

Cloud Service to Host: EC2

Expected to Design: Login and Functionality (Modules) of the System

Backend Development

Programming Language: Python Django

Cloud Service to Host: EC2

Expected to Design:

- It should receive the data from "React" axios request through URL and induce "SQL Query" to do operations on SQL
- It should be doing CRUD operations on RDS Cloud services and retrieve the data to display

Database Connectivity – SQL Based

You may initially have your local MySQL database to work with the python programming, then you can shift the connection to AWS RDS (MySQL) Service.

Cloud Service: AWS RDS (MySQL)

Expected to do:

Create RDS database within Free tier limits

Enable end points to connect with your python backend program

Store software functionality based transactional data in RDS

Modify the security group to allow the traffic which is going to come in to the RDS

Database Connectivity – Non SQL Based

Cloud Service: DynamoDB

Expected to do:

Create DynamoDB table with in free tier limits

Store the user registration based test data in DynamoDB

Storage – Object Way

Cloud Service: S3 Bucket

Expected to do:

Create S3 bucket and enable public access within free tier limits

Store the user registration based image (user photo) in S3 bucket

Passing Data from React to DynamoDB

Cloud Service: API Gateway and Lambda Functions (Serverless Services)

Expected to do:

When ever you are doing with the dynamodb related actions, your fronend react will trigger an API gateway's URL and that will trigger an Lambda microservices (lambda functions), which can have a code to interact with the dynamodb database.

API Gateway:

It is a Restful API URL which can trigger lambda function with any method (like post, get, delete, etc.)

Lambda Function (Serverless AWS Compute Service):

It is expected to have a code which can receive parameters and store the data in DynamoDB

It is expected to do CRUD operations over DynamoDB database

Deployment

You are expected to deploy the application using above services in addition to the consideration of below,

a. CI/CD Deployment
b. Sustainable Deployment

CI/CD Deployment:

Cloud Service: CodeCommit, CodeBuild, CodeDeploy and CodePipeLine

Expected to do:

Integrate CI/CD, So that, any small changes made and integrated will immediately auto deploy in multiple instances (instance group) at a time.

Sustainable Deployment:

Cloud Service: Elastic Load Balancer (ELB) and Auto Scaling Groups

Expected to do:

Your deployment is expected to adjust to utilize the cloud resources according to the load and demand of the market.

The input traffic is expected to deviate to multiple EC2 instances

The EC2 instances are expected to scale up and down as per the demand fluctuates in market

INTERVIEW PERSPECTIVE JUSTIFICATION

Reason for EC2 Choosing:

We are going to host full application which required full 24/7 working and EC2 is the cheaper option in AWS Cloud.

Reason for Lambda Choosing:

We are going to do some micro services for pushing the registration data into DynamoDB. It required only when the user is registering that. In this case, we need some micro service which will cost us only during execution and not on the other time. AWS Lambda is the best option available here.

Reason for RDS Choosing:

For any relational transaction database (complex database transactions) operations, RDS is best suitable.

Reason for DynamoDB Choosing:

I would like to separate the database for new user registration, so that, with out any load issue, new users can use (another database) “no sql” database with lesser cost, since it is only charging for execution time of your request (not for listening, like RDS) and storage with database table size.

Reason for S3 Choosing:

In our DynamoDB case, the cost is also associated with the storage and table size, to reduce any image based storage in DynamoDb, we are going with S3 buckets to store images and the URL of S3 bucket object may be stored in DynamoDb.

Reason for Choosing React as Frontend:

It is a single thread application, which will work as a single page app. So, it will increase the speed of loading next next virtual pages in the same page.

Reason for Choosing Python as a Backend:

It had wide library support and open forums with good execution speed with lesser number of lines to code.

FLOW OF THE PROJECT DEPLOYMENT IN AWS CLOUD

Compute, Storage, Database, API Gateway Service

React Deployment: EC2-1

Python with RDS Code: EC2-2

Primary Database for Application: RDS

User Registration Process Storage: DynamoDB

User Registration Process Image Storage: S3 Bucket

DynamoDB Write and Read code through: Lamda Microservices

Triggering of Lambda Services through: API Gate Way

CICD Integration and Deployment in Multiple Instances

Source file should be in Code Comit Master Repository

Testing should be done on Code Build

Deployment should be happened through Code Deploy

Code Pipe Line to integrate all the CICD Process

Load Balancing and Scaling of the Application

Use elastic load balancer and auto scaling group with templates, Cloud Watch, Alarms, SNS

PROJECTS

PS01:The Financial Management System:
The Financial Management System manages and monitors the flow of capital through the enterprise, including financial transactions, accounting and financial metrics.

PS02: The Facilities Management System:
The Facilities Management System acquires, develops, constructs and maintains enterprise facilities to provide a suitable working environment.

PS03: The Equipment Management System:
The Equipment Management System specifies, installs, calibrates and maintains equipment utilized to deliver the firm's value proposition.

PS04:The Employee Management System:
The Employee Management System qualifies, hires, monitors, develops and terminates employees to provide capable personnel that deliver their assigned organization responsibilities.

PS05:The Information Management System:
The Information Management System controls information, security and data related to the business requirements, including hardcopy, electronic and web.

PS06:The Application Conversion Management System:
This handles the conversion of files from Word to PDF, JPEG to pdf etc., Functionality is to convert from one format to another format

PS07:The Customer Development System:
The Customer Development System manages the customer experience, from first contact with the new prospect or lead through the first order for product or services, with ongoing account management and maintenance.

PS08:The Supplier Development System :
The Supplier Development System manages and monitors suppliers of materials and services to the enterprise, covering the complete supplier life cycle, from qualifying, setup, contact, development, and management through disqualification.

PS09:The Operations Management System:
The Operations Management System executes, delivers and manages the enterprise creation of customer value, including products and services quality, responsiveness, and economic value.

PS10:The Service Management System :
The Service Management System manages postproduction services, including installation, maintenance and service management, and customer follow-up, complaint handling and resolution.

PS11: Resume Builder :
Students need to provide their details and various templates will be provided by administrator.

PS12:Tribal People Support Management System :
Help the tribal artisans, not only meet their domestic requirements, but also help them in marketing their products either at home or at village or to businessmen. Despite best efforts to survive on their own these tribal households need market support for the survival of their traditional handicrafts.

PS13:Time Management System :
Time management is the act of planning, controlling and finally executing specific activities. A time management system is a combination of processes, tools, techniques, and methods and focuses on Personal Time management and Project Time Management.

PS14:University Event Management System :
Students can register various events organized by Department and administrator can generate the final winners and students will have a provision to download the certificate and generate client profiles.

PS15:Orphanage Management System :
The system manages records of children, guardians/caretakers, donors, sponsors, volunteers, adoptive-parents and adoption. The system allows the user to add, modify, delete and print the entry records.

PS16:Entertainment Management System :
Should enable the clients to register for the latest happenings around the city and attend for those events by registering for that

PS17:E-Commerce Applications :
Like local products to retail products, all types of products they can purchase in online mode.

PS18:Agricultural products Rural Entrepreneurship Management System :
Link agricultural products to global buyers. Making Agriculture attractive for Rural Youth by using Technology. Utilization and value addition in farm waste or by-products into value added products for urban markets. Agricultural product based value-added projects for promoting rural entrepreneurship.

PS19:Support Management System for Farmers :
The Main objective of Support Management system for farmers is to manage the details of medicines, treatments for the plants. The purpose of the project is to reduce the manual work for managing the medicines and tracking all the details of plants.

PS20:The Enterprise Management System :
Enterprise project management is the practice of managing multiple projects on a companywide scale. It requires developing standard processes to streamline project management across the company. The purpose of EPM is to link the company's goals and objectives with ongoing projects to ensure the company is directing its resources to the right places at the right time

PS21:Donation Management System :
People who need fund can raise a fund-raising campaign, after verifying the details of the campaign, the requests can see it on the clients dashboard. They can support for those campaigns through various plans.

PS22:Indian Culture Information Management System :
Nowadays people are not aware of historical monuments and their importance and most of them do not know about our prestigious Indian Culture. The specialty of historical places is innovation in architecture. The important characteristics of Indian culture are civilized communication, beliefs, values, etiquette, and rituals

PS23:Election Management System :
A System for improving the quality of elections. This can be useful to promote and protect the civil and political rights of participants in elections. It can lead to the correction of errors or weak practices, even

while an election process is still under way. It can deter manipulation and fraud or expose such problems if they do occur.

PS24:Reduce the Food Wastage and Improve the Food Security :
The number of hungry people in the world has increased over the last few years. One in nine people in the world go hungry each day and suffer from nutritional deficiencies as a result. In previous years, food security has been the biggest threat to the overall health of the human population. And 2020 saw the most severe increase in global food insecurity because of the COVID-19 pandemic, impacting vulnerable households everywhere. So, how can you solve this problem with innovation and advancement in Technology

PS25:Work Productivity Management System :
Productivity at its basis signifies the rate of output per unit of input. The phrase echoes an era where people were seen as elements in a production line, measured by the amount of time it would take them to produce a single item. But in today's knowledge-based economy, this measure of productivity is no longer viable, making it increasingly difficult to evaluate people's performance. There are a whole host of productivity challenges that we are now facing.

PS26:Learning Management System :
A learning management system (LMS) is a powerful eLearning tool for businesses and educational institutions to train employees, customers, and students. A LMS is just a tool and the effort to wield it effectively presents a host of challenges like Role Delegation, User Onboarding, time management, IT (Information Technology) Resource Management, Content creation and Publishing, etc. Build an effective solution to overcome the Challenges.

PS27: Recycled Waste Management System :
Recycling involves the collection of used and discarded materials, processing these materials and making them into new products. It reduces the amount of waste that is thrown into the community dustbins thereby making the environment cleaner and the air fresher to breathe. Develop an idea to effectively maintain the environment by recycling process

PS28:Handloom Fashion Products Management System :
The handloom sector is one of the major symbols of the cultural heritage of India. Apart from that, it continues to be an important source of livelihood, especially for women, who form around 70% of the weavers or allied workers in the sector. So, design an idea to promote the handloom industry to the global markets.

PS29:Electricity Billing System :
Electricity Billing System is a Computer Operated Billing Software that tells Electricity Usage Time, interval billing on daily, weekly, monthly and interval basis. This Project also incorporates distribution channel billing and provides flexibility to handle different accounts and bill calculations.

PS30:Online Medical System:
This project is to create a direct line of communication between doctors and patients. The project is known as "Virtual Medicine Home." By using this application, patients can book online appointments with their preferred doctors, and doctors can offer healthcare suggestions, e-prescription, and view the patient's medical records, lab reports, etc.

PS31:Live Quiz Management System :

This allows organizations to host live games with various question types such as multiple-choice, true/false, images or short answers. Supervisors can gain visibility into the number of players or questions, view students' performance and generate reports in real-time to identify difficult questions.

PS32:Event Management system :
The Event management system project will work and update the event's records, participant's records, all expenditures during an event, staff and employee's record. It will also keep the track record of event and other events that have been carried out.

PS33:Online Organic health Food Store Management System :
Organic food products and other organic ingredients are grown without the use of pesticides, synthetic fertilizers, sewage sludge, or ionizing radiation. Conventional fruits and vegetables are often sprayed with pesticides. When you buy such fruits and vegetables, these stubborn chemicals remain on the food.

PS34:Online News Portal :
To publish the news in online mode and send notifications to the users as soon as any new update and display the news based on various categories like sports, trading..

PS35:Weather Monitoring management System :
Weather monitoring is also important not just in defining present climate, but also for detecting changes in climate and forecasting of approaching dangerous weather like storms, cyclones, floods. Develop an application which keeps track of weather day to day and alert for any natural calamities

PS36:Placement Interaction Management system :
This system is developed to assist the placement and training cell of Karpagam Arts and Science College. The main purpose of this project is to store the student's details and to find all the eligible candidate from UG and PG, satisfying the specified criteria.

PS37:Diet Management System :
A balanced diet is one that fulfils all of a person's nutritional needs. A balanced diet provides all the nutrients a person requires, without going over the recommended daily calorie intake. Design a solution to detect macronutrient deficits in children, and adolescents, to enable timely interventions and corrections to avoid future complications in growth and development.

PS38:Logistics Management System :
Logistics management is a supply chain management component that is used to meet customer demands through planning, control and implementation of the effective movement and storage of related information, goods and services from origin to destination.

PS39:Student Grievance Addressal Management System :
The system functions to investigate the grievances lodged by any student. Students may approach the cell to voice their grievances regarding academic matters, health services, library and other services. Anyone with a genuine grievance may approach the Co-ordinator or member of the Student's Grievance cell, Grievance cell is formed in order to keep the healthy working atmosphere amongst staff and students, Admin can view the resolving status of all grievance

PS40:Customer Relationship Management System :
The main purpose of CRM is to improve the relationship with the customers by using different module like analysis, customer service and others. The objective to do this project is to develop a system which can help the organizations to decrease their defection rate of customers. Because the lower defection rate means the bigger customer base, which lead to more profit for the organization.

PS41:Course Management System:
Course Management System Software is an online management application, built as a software development project. Its main purpose is to make efficient interaction between students and instructors in college during the period of submission of assignments, projects, thesis, and for getting appropriate feedback from instructors.

PS42:Stock Exchange Management System :
A stock exchange helps companies raise capital or money by issuing equity shares to be sold to investors. The companies invest those funds back into their business, and investors, ideally, earn a profit from their investment in those companies. Solve the stock exchange problems like Lack of Professionalism, Domination of Financial Institutions, Poor Liquidity, Domination by Big Operators, Less Floating Stocks, Speculative Trading, etc.

PS43:Smart Technology Information Management System :
Recently, many practices using smart technologies impacted farming and produced impressive results, but they are not known to many farmers in the world. So, let the farmers know what those smart technologies are and where we to use those smart technologies and let them know the impact and cost of the smart technology.

PS44:Artwork management System :
Art galleries can showcase the history of a particular culture in a visual manner. As such, this visual history provides a snapshot of what life was like at a particular period through the artwork. For instance, art may provide information about the people living in a particular time

PS45:Refurbished products management System:
Selling the Refurbished products in online mode either community based, or Location based.

PS46:Health Care Management System :
The e-Healthcare management system is a customizable, comprehensive, and integrated Hospital Management System designed to manage all hospital operations. The ideal client base for the hospital is Healthcare facilities, multi-specialty clinics, and medical practitioners. This system is reliable and flexible in all aspects, so new features and modules can be easily integrated into the system in future.

PS47:Travel and Tourism Management System :
The Main objective of this Travel & Tourism project is to make the travel easy and comfortable for the users right from finding the routes and buses to till the booking of the tickets and about various travel agencies and their packages

PS48:Supply Chain Management System :
Supply Chain management System is the web-based system is designed to run on any computer. It automates the system of communication between the management or admin, dealers and clients of the organization/company. Additionally, it enables the user of the application to view issues via LAN/Internet. Based on the category of the user – administrator or employee, various parts of system are made accessible to the users.

PS49:Gallery Automation :
To create various templates and organize the photos in those templates and share them via social platforms.

PS50:Spilt the Bill Application :
A group of people can effectively use to track their expenditures for the month and can add the expenditures and to whom we need to share the bill.

SCHEDULE

30th MAR 2022 (Complete ONLINE)

06:00AM – Review 0 (Idea of Implementation)

08:30AM - Session AWS Connectivity with react, python, rds, dynamodb, lambda, apigateway, s3 bucket

10:30AM – Project Development commences

02:30PM – Review 1

06:00PM - Short Quiz (AWS Quiz 11)

09:30PM – Review 2

01:30AM - Closure of Project Development for the day

31st MAR 2022 (HYBRID)

(ON Campus)

09:15AM – Project Development commences

10.30AM - Review 3

01:30PM - Short Quiz (AWS Quiz 12)

(Online)

06:30PM - Review 4

01:30AM - Closure of Project Development for the day

01st APR 2022 (ON-CAMPUS)

09:15AM - Project Development commences

11:00 AM to 11.20 AM – Final Mock Test (MCQ) (AWS Final Quiz)

04:00PM – Review 5 - Final Review of Project

REVIEW DETAILS WITH RUBRICS

Review 0 - Understanding

Understanding of Business System

Identification of Modules with User Research and User Interface using Figma

Rating of Teamwork

Review1 - React Front End Development

Modules Transition to Frontend

Dashboard Design

Rating of Teamwork

Review 2 - React EC2 to API Gateway and further to Lambda and DynamoDB

Lambda to DynamoDB

API Gateway Integration with Lambda

React EC2 to API Gateway call with Parameters

React Frontend to DynamoDB Implementation

Implementation of Different CRUD Operations

Improvement in overall design of the System

Rating of Teamwork

Review 3 - React EC2 to S3 Bucket to store images

IAM User Access Key Generation

React EC2 to S3 Bucket to Store Images

Improvement in overall design of the System

Rating of Teamwork

Review 4 - React EC2 to Python EC2 and further to AWS RDS

React to Python Communication

Python to RDS Communication

Implementation of Different CRUD Operations

Improvement in overall design of the System

Rating of Teamwork

Review 5 – CI/CD Integration and deployment with all services in cloud

Compute, Storage, Database, API Gateway Deployment

CI/CD Deployment

ELB and ASG Deployment

Rating of Overall Project

Rating of Teamwork

REFERENCE LINKS

Python with s3 upload image

https://javascript.plainenglish.io/how-to-upload-files-to-aws-s3-in-react-591e533d615e

https://medium.com/bilesanmiahmad/how-to-upload-a-file-to-amazon-s3-in-python-68757a1867c6

React Support Link:

https://fullstackopen.com/en/#course-contents

SAMPLE APPLICATION GIT LINK

Sample React and Python Based Application – Coded for this Requirement

https://github.com/balajee-rm/aws/blob/main/AWS%20AND%20AZURE%20FULL%20APP.zip

STEP BY STEP COMPLETE GUIDANCE

Creating EC2 Machine for REACT RDS (Linux Based Client)

In terminal put the commands

sudo yum update

curl -sL https://rpm.nodesource.com/setup_14.x | sudo bash -

sudo yum install -y nodejs

node –version

for creating new react app the commands are “npx create-react-app part1” and “cd part1”

past the “rdsclient” App.js code to created “part1” App.js

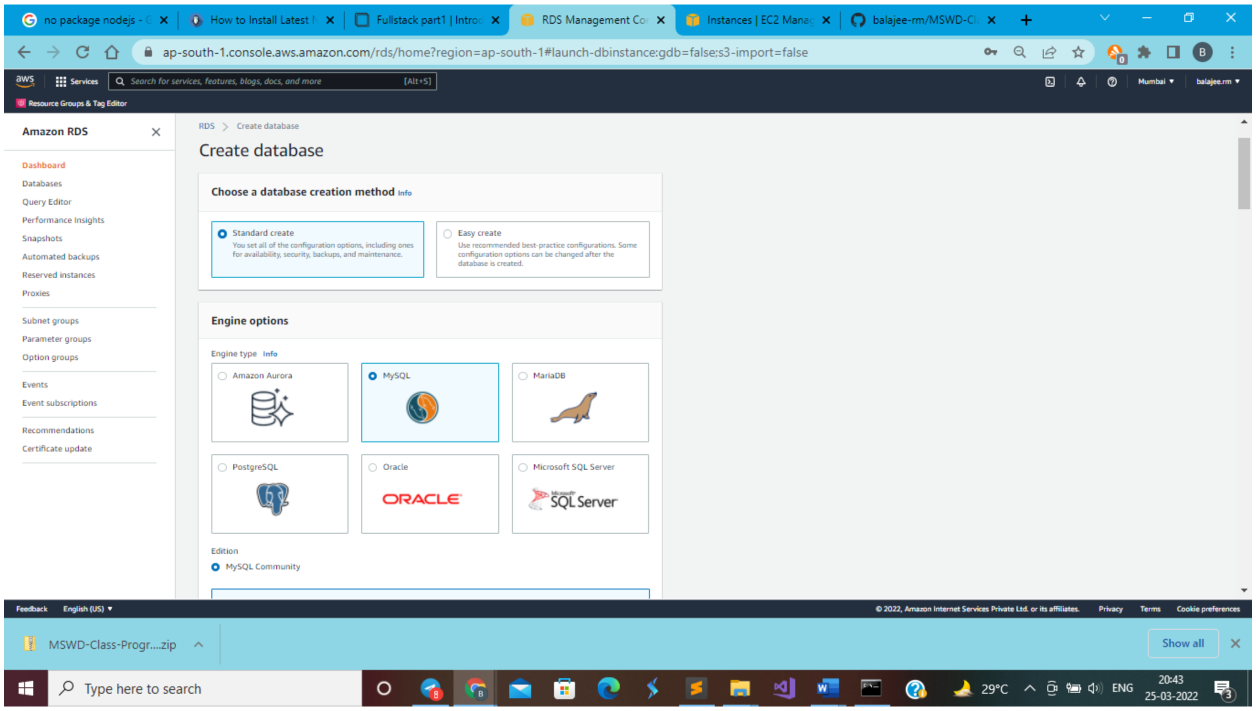
npm install axios

go to package.json file, add “start": "PORT=8080 react-scripts start"

npm start

Createing RDS (MySQL)

Choose standard create and and MYSQL



RDS > Create database

Create database

Choose a database creation method

[✓ Standard create] You set all of the configuration options, including ones for availability, security, backups, and maintenance.

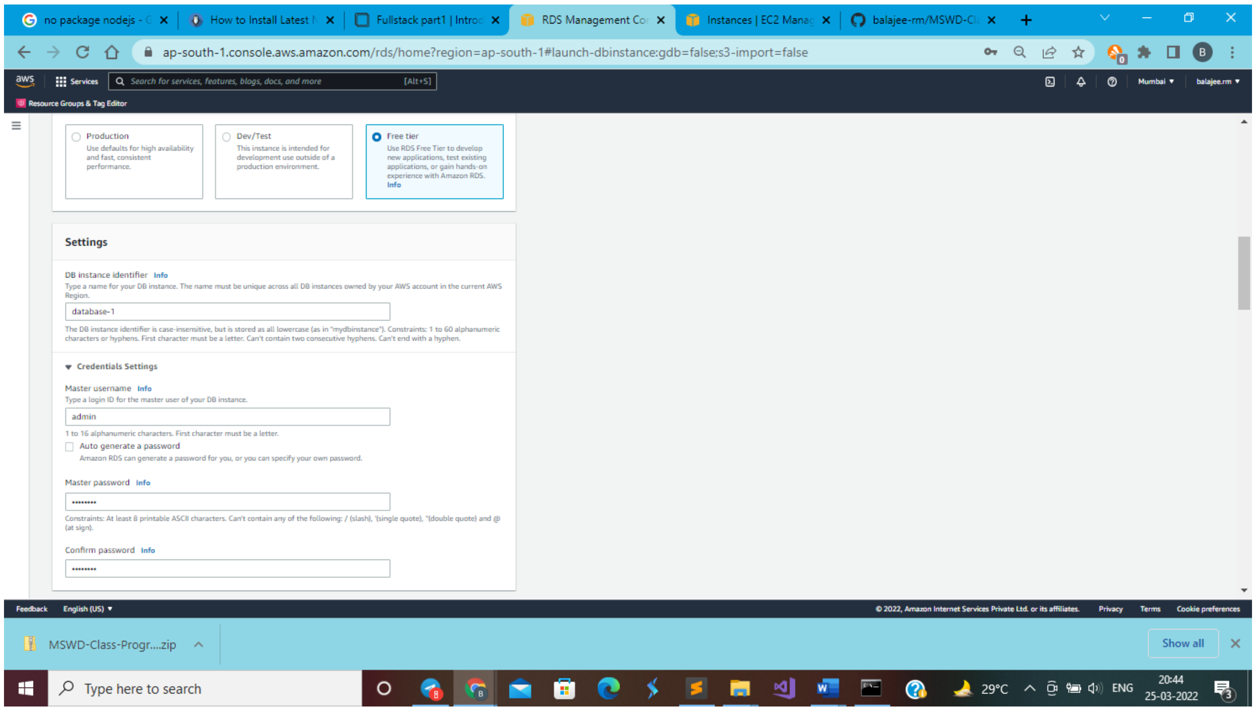
Easy create: Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type: Amazon Aurora, [✓ MySQL], MariaDB, PostgreSQL, Oracle, Microsoft SQL Server

Edition: [✓ MySQL Community]

Choose free tier label and give paswords



Production: Use defaults for high availability and fast, consistent performance.

Dev/Test: This instance is intended for development use outside of a production environment.

[✓ Free tier] Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.

Settings

DB instance identifier: Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username: Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. First character must be a letter.

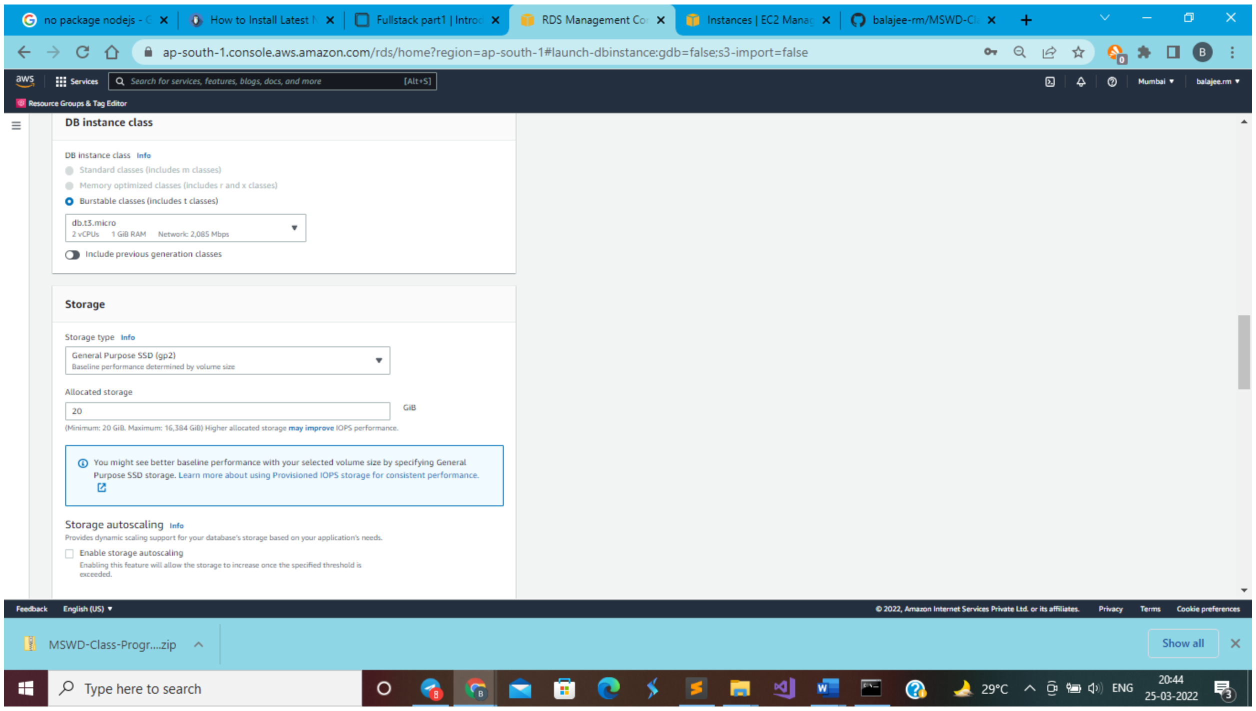
☐ Auto generate a password: Amazon RDS can generate a password for you, or you can specify your own password.

Master password: ••••••••

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm password: ••••••••

Ensure db.t2.micro and uncheck the “Enable storage autoscaling”



DB instance class

DB instance class Info

Standard classes (includes m classes) [✓ Burstable classes (includes t classes)] Memory optimized classes (includes r and x classes)

db.t3.micro
2 vCPUs 1 GiB RAM Network: 2,085 Mbps

Include previous generation classes

Storage

Storage type Info

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage

20 GiB

(Minimum: 20 GiB. Maximum: 16,384 GiB) Higher allocated storage may improve IOPS performance.

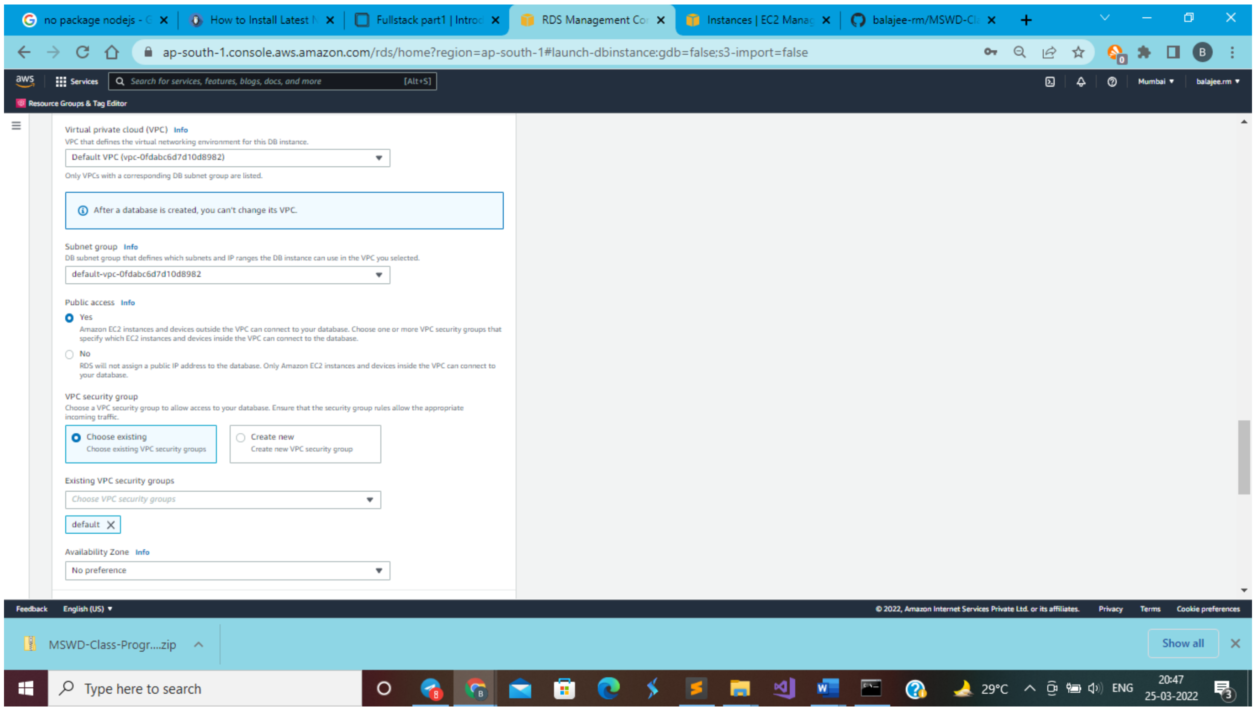
You might see better baseline performance with your selected volume size by specifying General Purpose SSD storage. Learn more about using Provisioned IOPS storage for consistent performance.

Storage autoscaling Info

Provides dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling
Enabling this feature will allow the storage to increase once the specified threshold is exceeded.

Give public access yes



Virtual private cloud (VPC) Info

VPC that defines the virtual networking environment for this DB instance.

Default VPC (vpc-0fdabc6d7d10d8982)

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

Subnet group Info

DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default-vpc-0fdabc6d7d10d8982

Public access Info

[✓ Yes]
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group

Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

[✓ Choose existing] Choose existing VPC security groups

Create new
Create new VPC security group

Existing VPC security groups

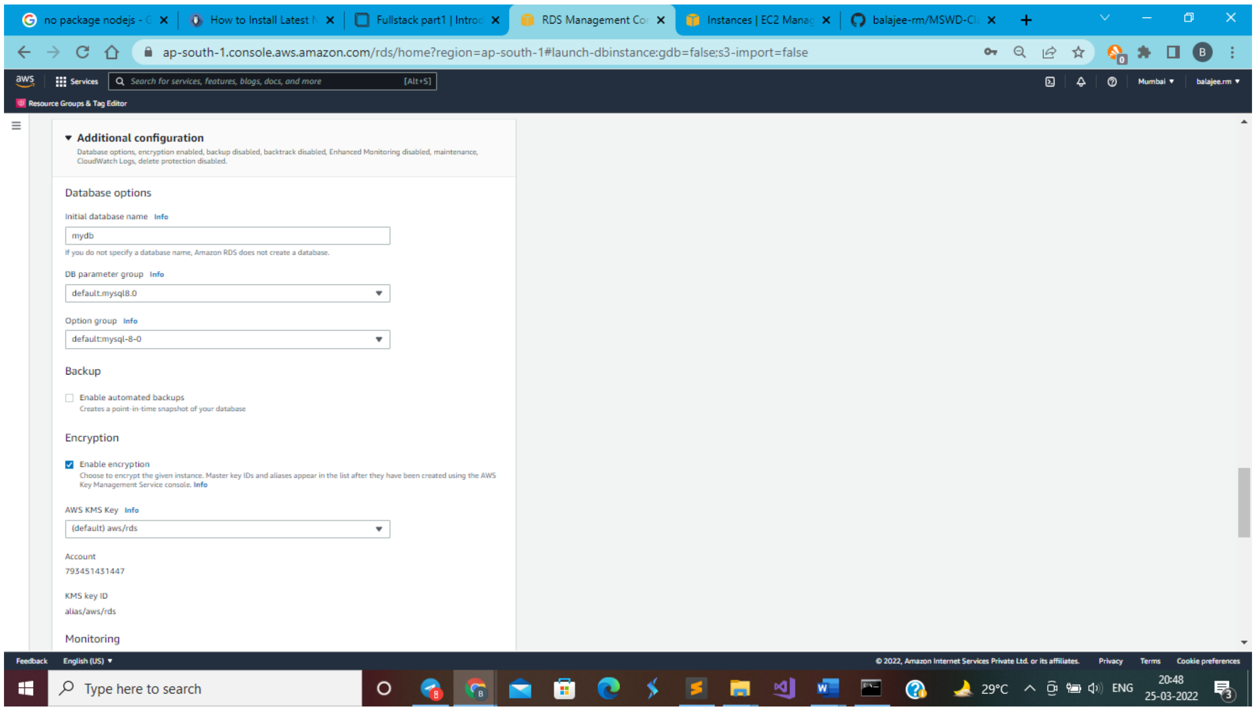
Choose VPC security groups

default

Availability Zone Info

No preference

Give initial database name as “mydb” and uncheck “Enable automated backups”


Additional configuration

Database options, encryption enabled, backup disabled, backtrack disabled, Enhanced Monitoring disabled, maintenance, CloudWatch Logs, delete protection disabled.

Database options

Initial database name: mydb

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group: default.mysql8.0

Option group: default:mysql-8-0

Backup

☐ Enable automated backups
Creates a point-in-time snapshot of your database

Encryption

☑ Enable encryption
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console.

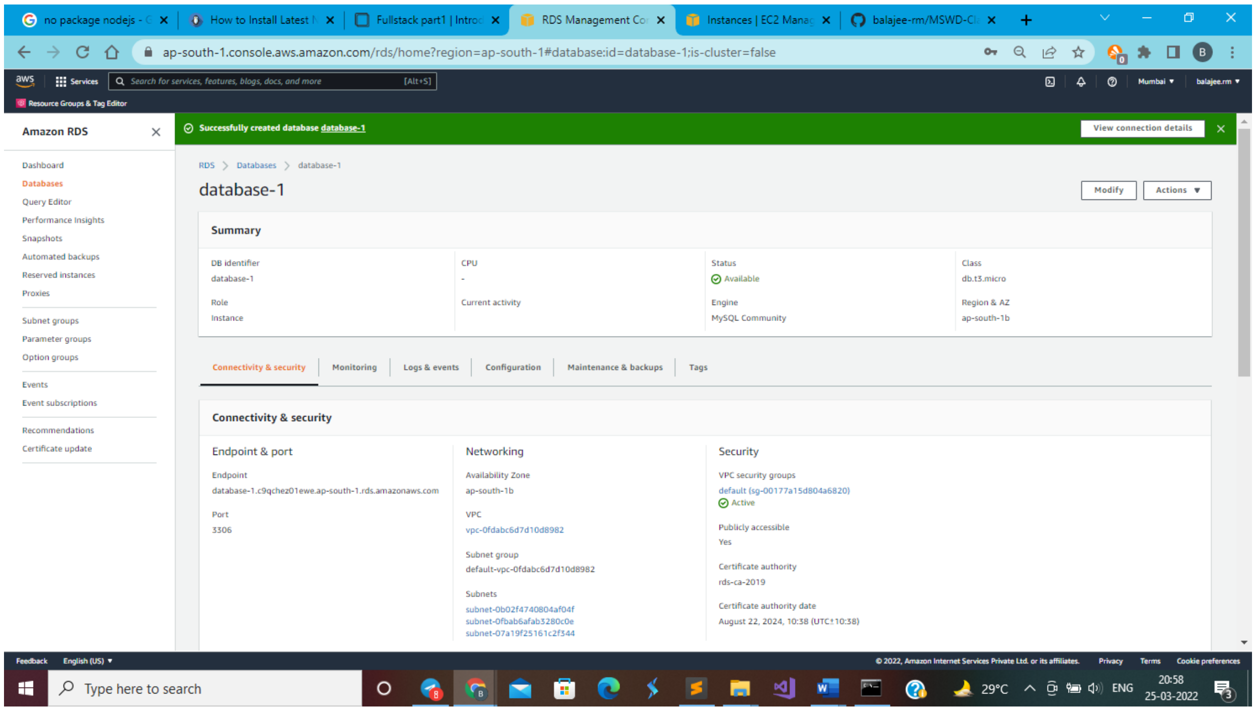
AWS KMS Key: (default) aws/rds

Account: 793451431447

KMS key ID: alias/aws/rds

Monitoring

Create the database and open it


Successfully created database database-1

RDS > Databases > database-1

database-1

Summary

DB identifier: database-1	CPU: -	Status: Available	Class: db.t3.micro
Role: Instance	Current activity	Engine: MySQL Community	Region & AZ: ap-south-1b

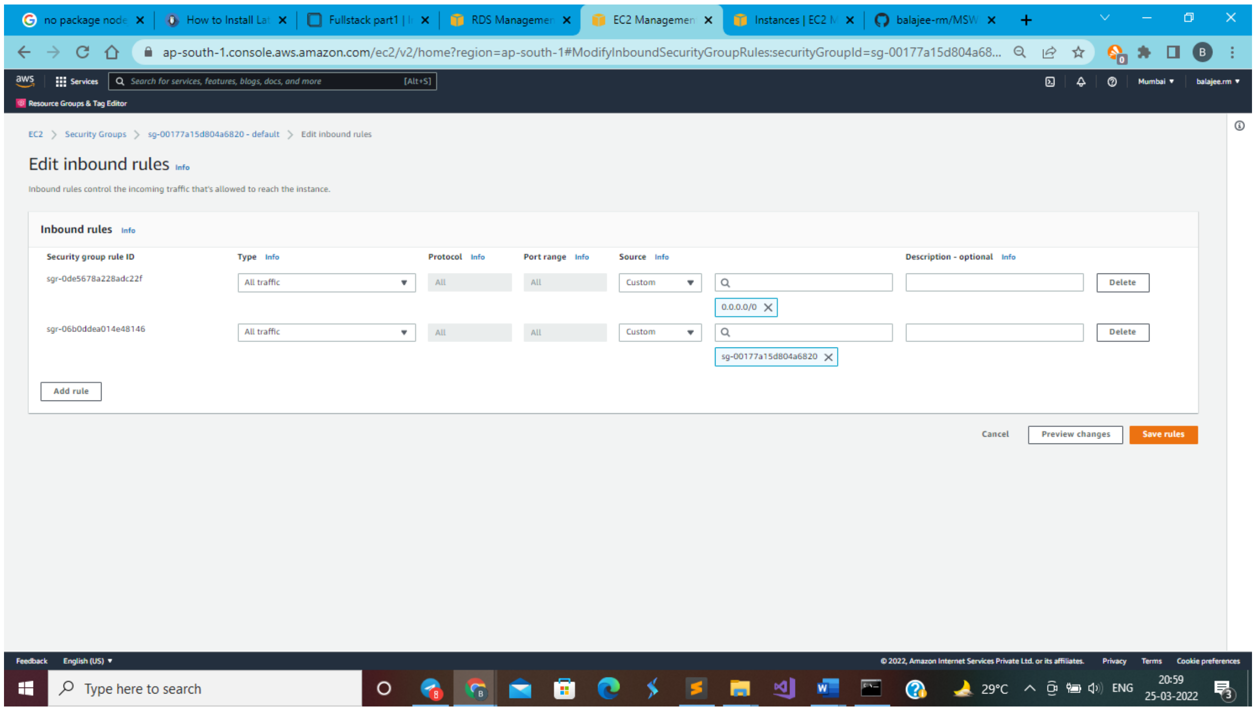
Connectivity & security

Endpoint & port
Endpoint: database-1.c9qchez01ewe.ap-south-1.rds.amazonaws.com
Port: 3306

Networking
Availability Zone: ap-south-1b
VPC: vpc-0fdabc6d7d10d8982
Subnet group: default-vpc-0fdabc6d7d10d8982
Subnets: subnet-0b02f4740804af04f, subnet-0fbab6afab3280c0e, subnet-07a19f25161c2f344

Security
VPC security groups: default (sg-00177a15d804a6820) Active
Publicly accessible: Yes
Certificate authority: rds-ca-2019
Certificate authority date: August 22, 2024, 10:38 (UTC±10:38)

Go to security group shown in the pic and edit the inbound rules to all traffic



EC2 > Security Groups > sg-00177a15d804a6820 - default > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0de5678a228adc22f	All traffic	All	All	Custom	0.0.0.0/0	
sgr-06b0ddea014e48146	All traffic	All	All	Custom	sg-00177a15d804a6820	

Creating ec2 for Python RDS (Linux based Server)

Django based hosting

Support Link 1: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-django.html#python-django-setup-venv

Support Link 2: https://realpython.com/django-setup/

Commands in Terminal (Linux Based)

```
sudo yum update

sudo yum install python3

pip3 list

pip3 install virtualenv

pip3 install awsebcli

virtualenv virt

source ~/virt/bin/activate

// for deactivation – go to the folder where virtual file is created and then give “deactivate”

// to remove a directory “rm -r <directory_name>”

pip3 install django==2.1

//to verify

pip freeze

django-admin startproject proj1
```

//the above command will create the directory as below,

```
~/proj1
  |-- proj1
  |   |-- __init__.py
  |   |-- settings.py
  |   |-- urls.py
  |   `-- wsgi.py
  `-- manage.py
```

cd proj1

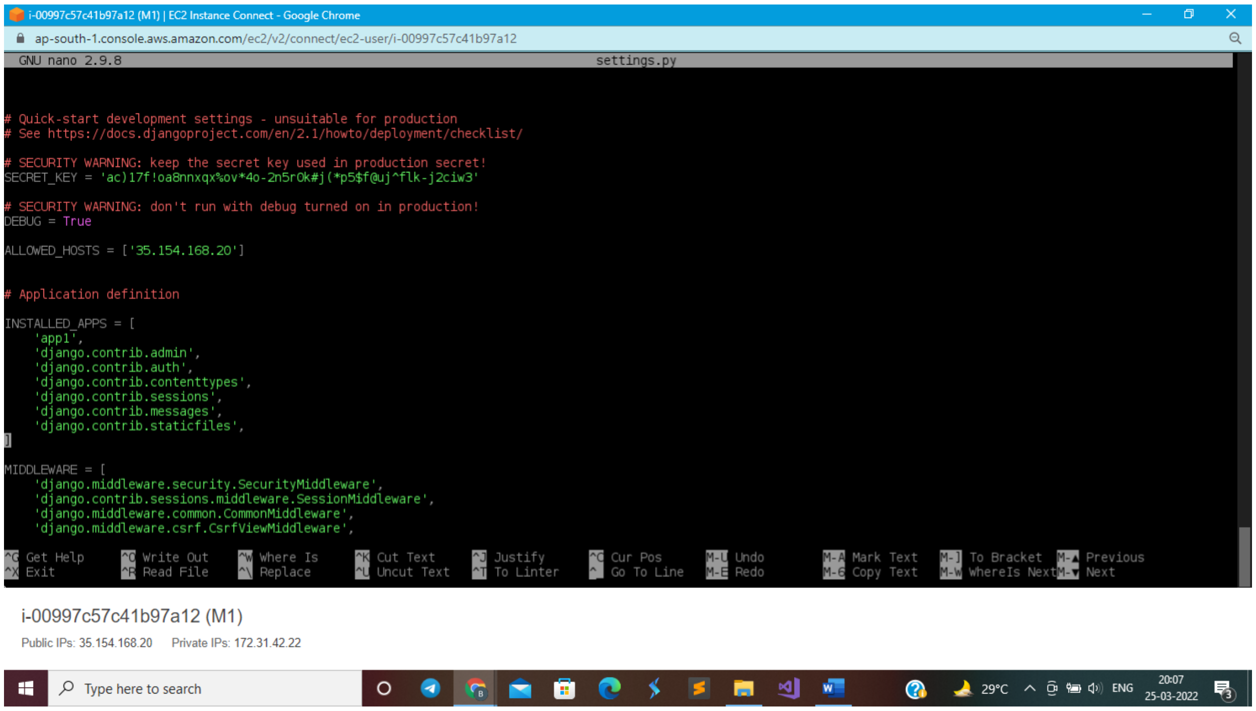
python manage.py startapp app1

it will create a following directory structure

```
proj1/
|
├── app1/
|   |
|   ├── migrations/
|   |   └── __init__.py
|   |
|   ├── __init__.py
|   ├── admin.py
|   ├── apps.py
|   ├── models.py
|   ├── tests.py
|   └── views.py
|
├── proj1/
|   ├── __init__.py
|   ├── asgi.py
|   ├── settings.py
|   ├── urls.py
|   └── wsgi.py
|
└── manage.py
```

go to settings.py in second “proj1” and add “app1” in the installed apps

go to settings.py in second “proj1” and add “ALLOWED_HOSTS=['35.154.168.20']” this is a public ip of the instance



```
  GNU nano 2.9.8                                                             settings.py


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'ac)17f!oa8nnxqx%ov*4o-2n5r0k#j(*p5$f@uj^flk-j2ciw3'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['35.154.168.20']


# Application definition

INSTALLED_APPS = [
    'app1',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos    M-U Undo   M-A Mark Text  M-] To Bracket  M-▲ Previous
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Linter ^_ Go To Line M-E Redo   M-6 Copy Text  M-W WhereIs Next M-▼ Next
```

i-00997c57c41b97a12 (M1) | EC2 Instance Connect - Google Chrome
ap-south-1.console.aws.amazon.com/ec2/v2/connect/ec2-user/i-00997c57c41b97a12
i-00997c57c41b97a12 (M1)
Public IPs: 35.154.168.20 Private IPs: 172.31.42.22
Type here to search
29°C
ENG
20:07
25-03-2022

Come out to first "proj1" folder and type

sudo nano runserver.py

//create the file "runserver.py" in parallel to manage.py and past the code below to change the port from 8000 to 8080

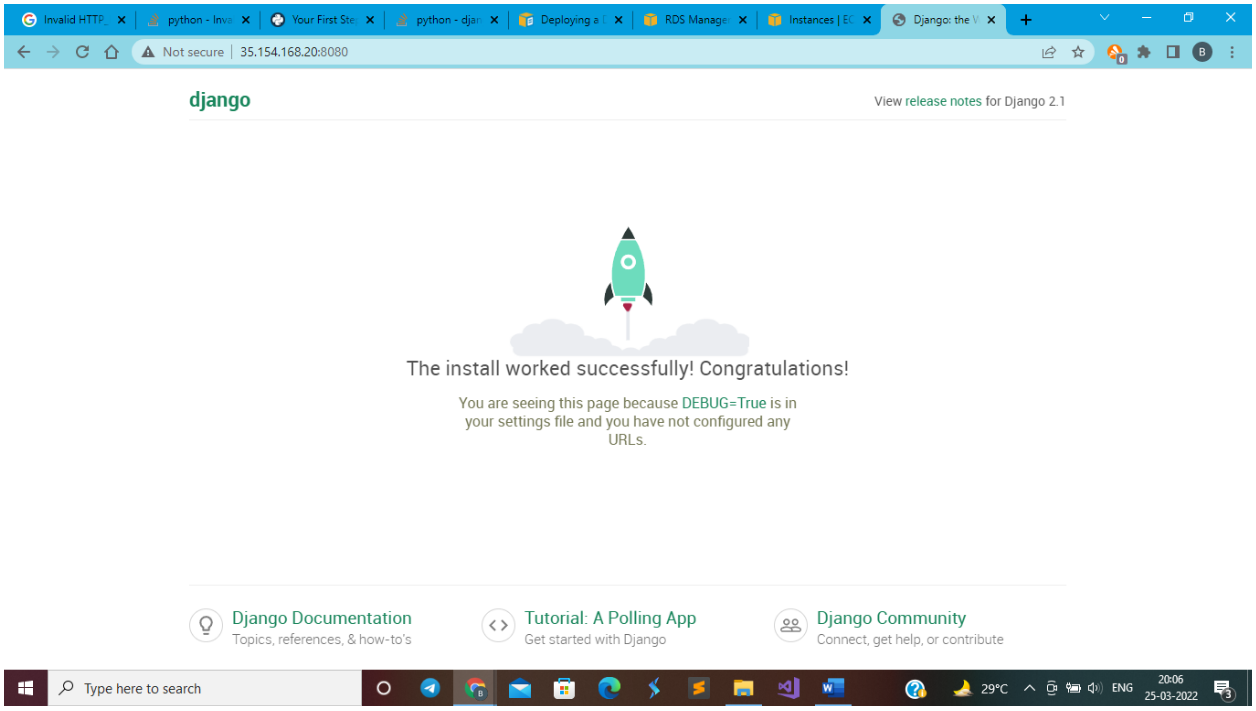
```
#!/bin/bash
exec ./manage.py runserver 0.0.0.0:8080
```

sudo chmod +x runserver.py

./runserver.py

// no need of this command "python manage.py runserver" due to above one

put "public_ip_address:8080" in browser


django
View release notes for Django 2.1
The install worked successfully! Congratulations!
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

Now break the server and go to urls.py in your second proj1 directory and have these codes

Code

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

  path('admin/', admin.site.urls),

  path('', include('app.urls')),

]

Install “filezilla client” and connect your EC2

Now go to “app1” directory and past and replace the docs (urls.py, views.py, test.py, models.py, forms.py, apps.py, admin.py)

Inside app directory, create a folder called “templates” and past the below files (display.html, index.html, page2.html, upload.html)

Now in putty terminal, go to “apps.py” under app1 directory and replace the text “app” to “app1”.

Now go to “views.py” in app1 directory using terminal and open it.

Change this line “con = sql.connect(host='localhost', user='root', password='root', database='mydb');” with your data of RDS.

Now go to “settings.py” under second proj1 folder and add replace these code,

Code

DATABASES = {

```
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        #'ENGINE': 'django.db.backends.sqlite3',
        #'NAME': BASE_DIR / 'db.sqlite3',
        'HOST': 'localhost',
        'USER': 'root',
        'PASSWORD': 'root',
        'NAME': 'mydb',
    }
}
```

In the above code, provide your RDS details

Then go to and edit the "__init__.py" file in your project origin dir(the same as settings.py)

Add code:

```
import pymysql
pymysql.install_as_MySQLdb()
```

Now, go to your terminal and install the following

```
pip3 install mysql-client
pip3 install mysql-connector-python
pip3 install pymysql
```

now run application as "./runserver.py"

you can able to communicate with the database now by adding "/upload" and show it by adding "/display" in the url.

Python Flask Way

In terminal put the commands

```
sudo yum update
sudo apt install python3
sudo apt install python3-pip
pip list
pip install flask
pip install mysql-client
pip install mysql-connector-python
pip install flask_cors
pip install ndg-httpsclient
```

pip install pyopenssl

pip install pyasn1

copy the “pythonserver” folder to ec2

sudo nano server.py

Replace the RDS “end point” and fulfil the connection string in “server.py” file under python server

Change the port to “8080”

Run as “python server.py” inside the pythonserver directory

put the “public_ip_address:8080/create_table” in browser new tab, then after “public_ip_address:8080”

Utilize Existing EC2 Machine for DynamoDB and S3 (registration page)

sudo yum update

curl -sL https://rpm.nodesource.com/setup_14.x | sudo bash -

sudo yum install -y nodejs

node –version

for creating new react app the commands are “npx create-react-app part1” and “cd part1”

past the “dynamodbclient” App.js code to created “part1” App.js

npm install axios

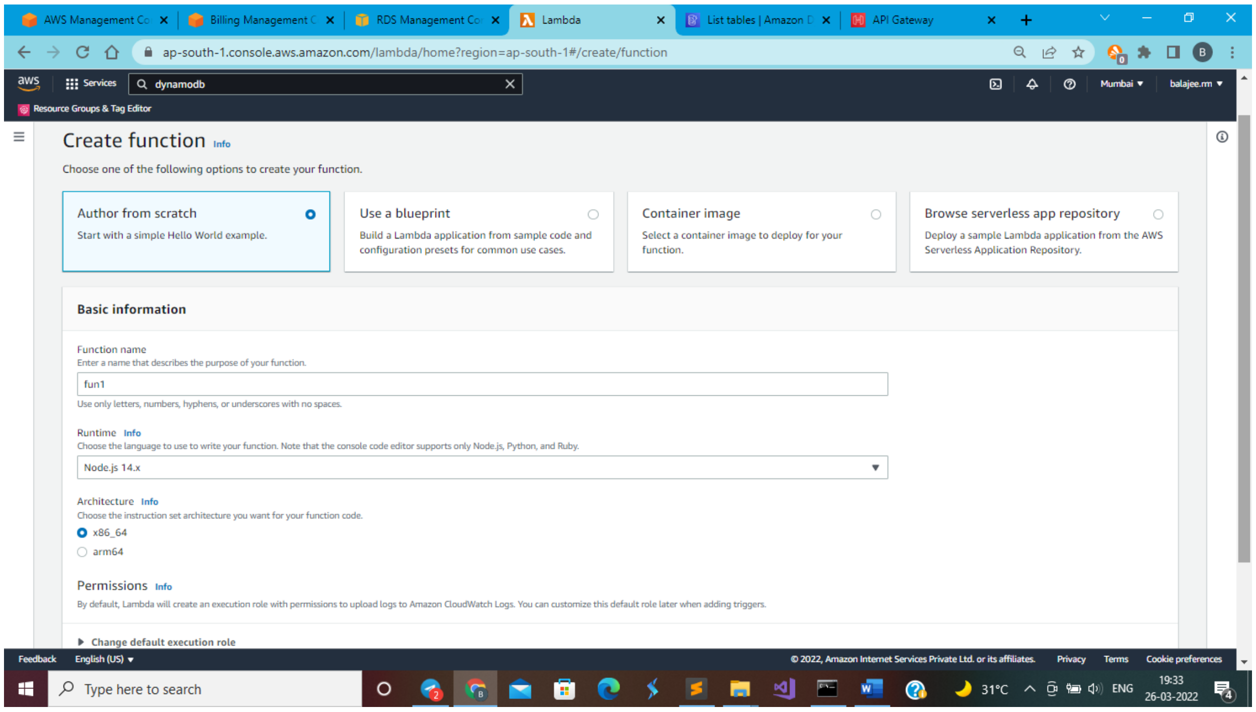
go to package.json file, add “start": "PORT=8080 react-scripts start"

npm start

Lamdba and DynamoDb Creation

Create Lambda function with name “fun1” as node.js function

Go to lambda dashboard and click on “create function”



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/create/function

Create function Info

Choose one of the following options to create your function.

[✓ Author from scratch] Start with a simple Hello World example.

Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.

Container image: Select a container image to deploy for your function.

Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

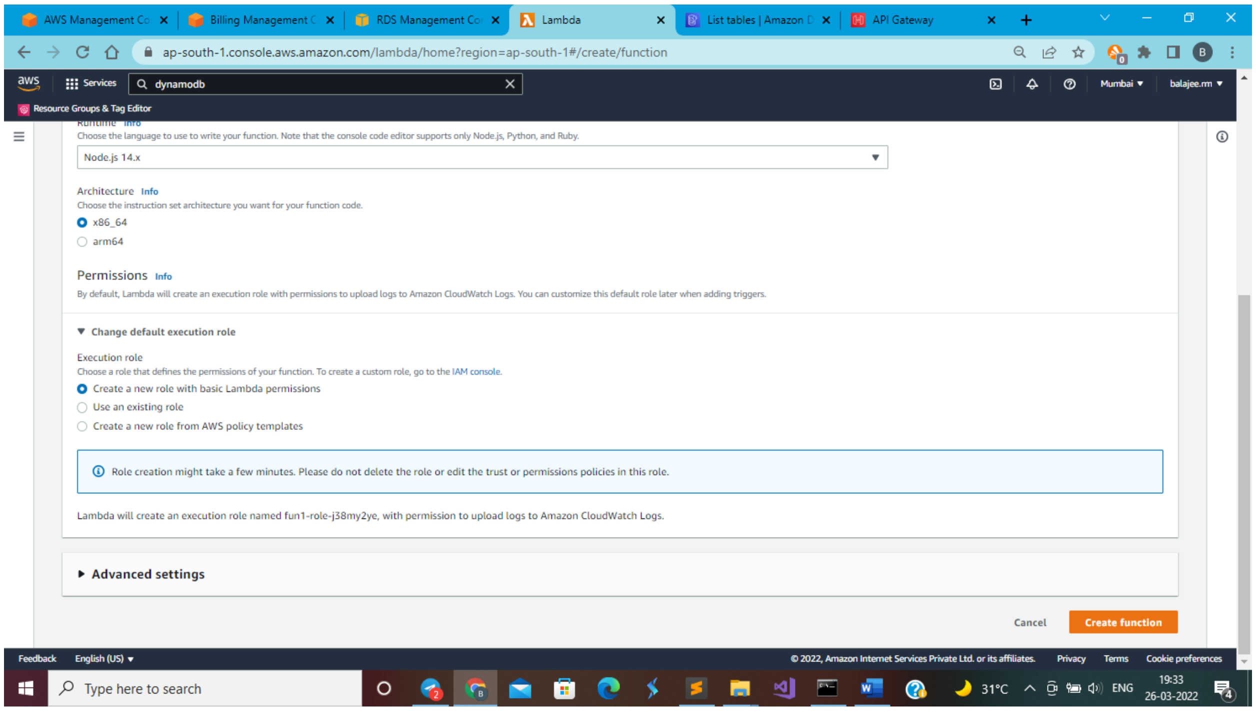
Function name
Enter a name that describes the purpose of your function.
fun1
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 14.x

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ Change default execution role



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/create/function

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 14.x

Architecture Info

Choose the instruction set architecture you want for your function code.

[✓ x86_64] arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

[✓ Create a new role with basic Lambda permissions] Use an existing role; Create a new role from AWS policy templates

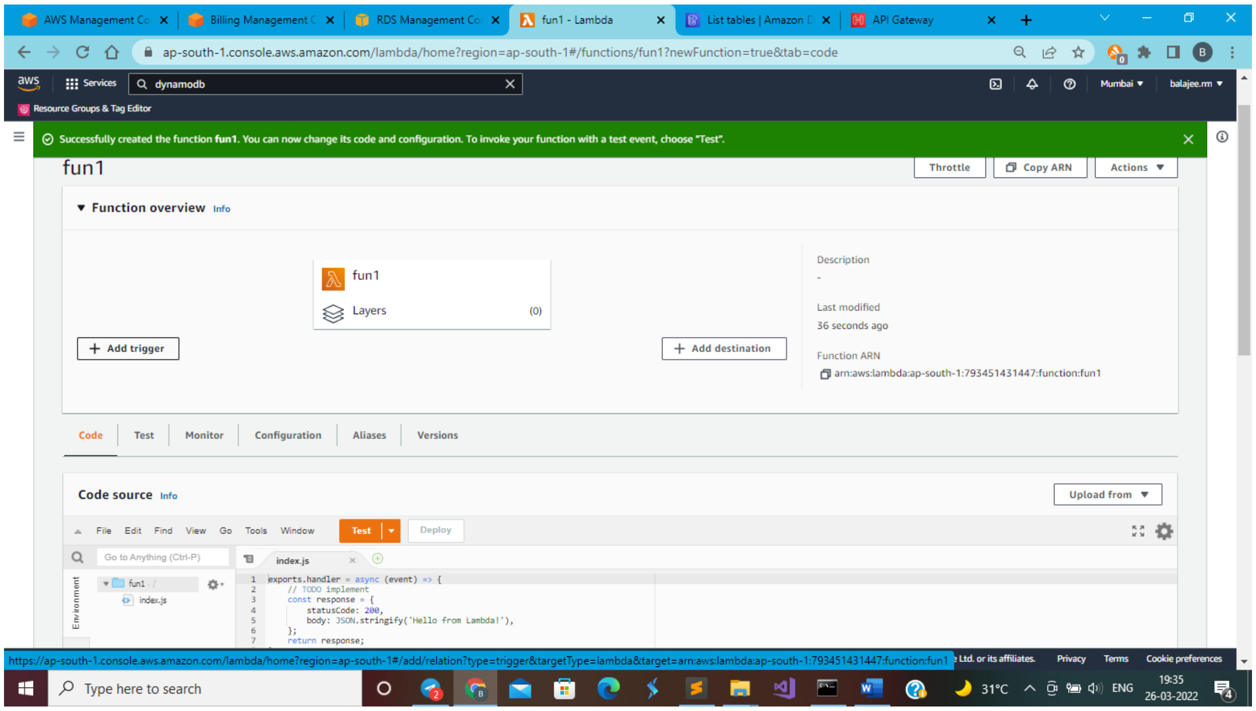
Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named fun1-role-j38my2ye, with permission to upload logs to Amazon CloudWatch Logs.

Advanced settings

Cancel Create function

ADD API TRIGGER



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/fun1?newFunction=true&tab=code

Successfully created the function **fun1**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

fun1

Function overview Info

Description: -

Last modified: 36 seconds ago

Function ARN: arn:aws:lambda:ap-south-1:793451431447:function:fun1

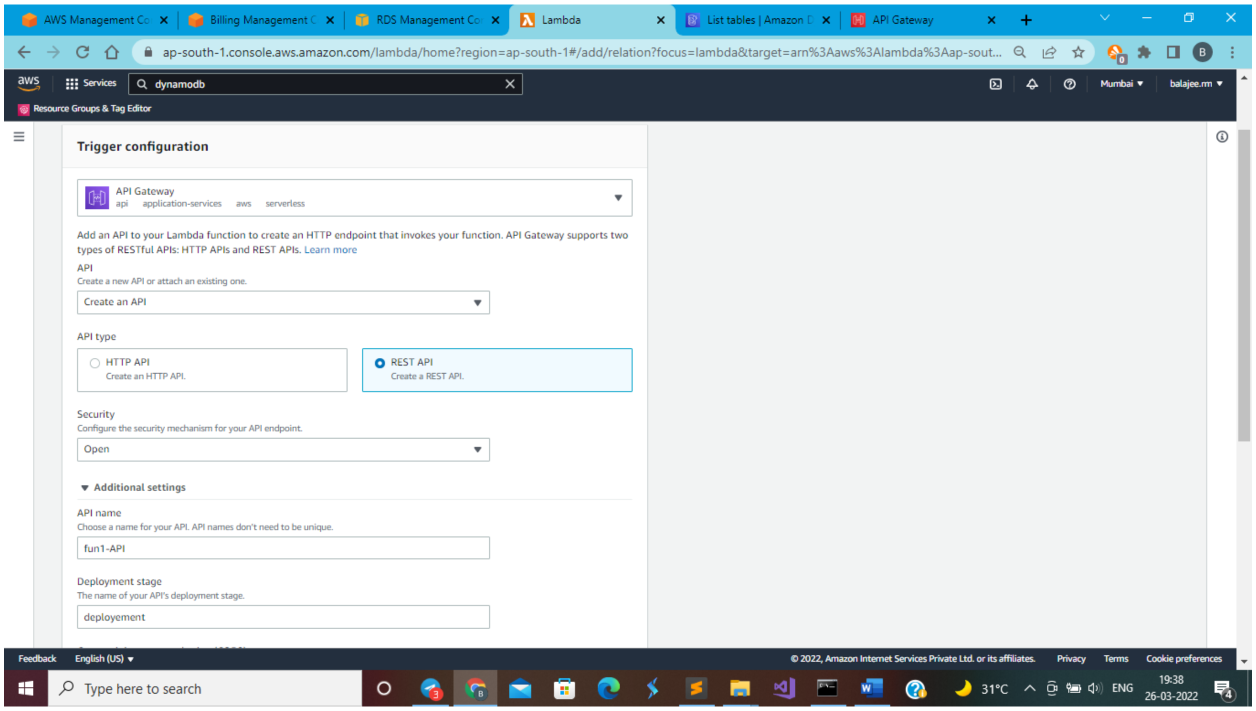
Code source Info

index.js

```
exports.handler = async (event) => {
    // TODO implement
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
```

https://ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/add/relation?type=trigger&targetType=lambda&target=arn:aws:lambda:ap-south-1:793451431447:function:fun1

Click on “add trigger” button and do the following


Trigger configuration

API Gateway
api application-services aws serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. Learn more

API
Create a new API or attach an existing one.
Create an API

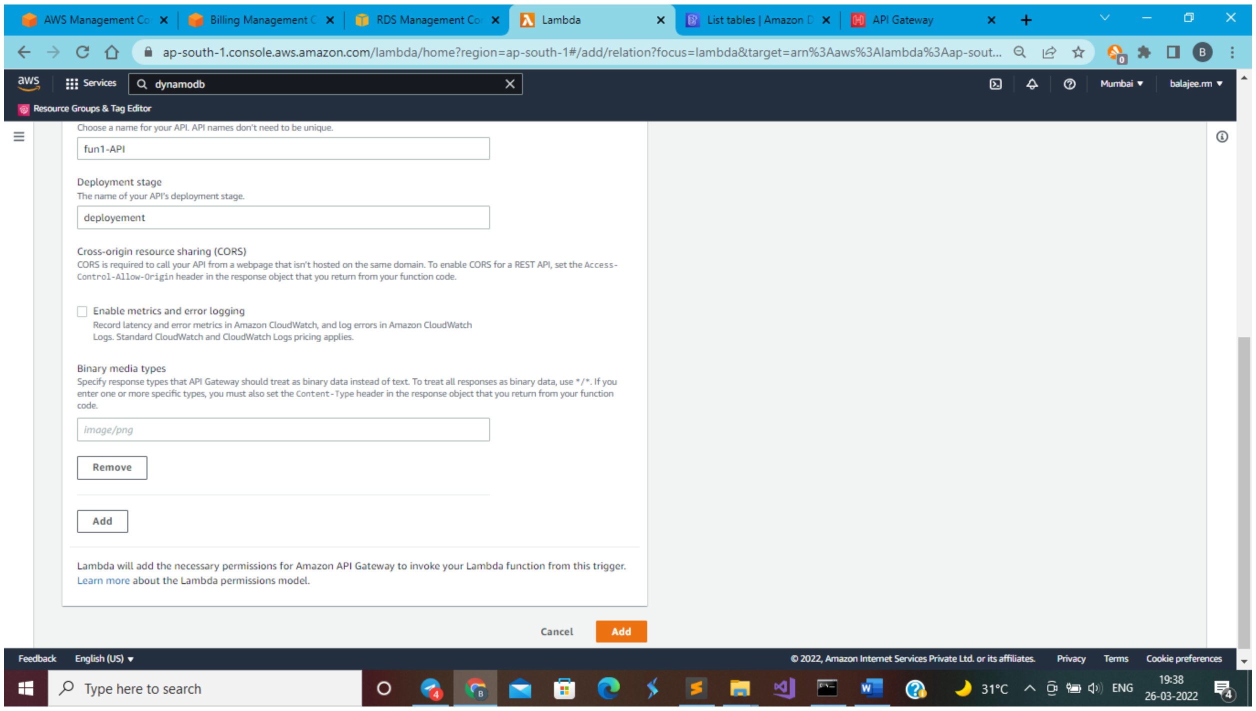
API type
HTTP API — Create an HTTP API.
REST API — Create a REST API. (selected)

Security
Configure the security mechanism for your API endpoint.
Open

Additional settings

API name
Choose a name for your API. API names don't need to be unique.
fun1-API

Deployment stage
The name of your API's deployment stage.
deployement


Choose a name for your API. API names don't need to be unique.
fun1-API

Deployment stage
The name of your API's deployment stage.
deployement

Cross-origin resource sharing (CORS)
CORS is required to call your API from a webpage that isn't hosted on the same domain. To enable CORS for a REST API, set the `Access-Control-Allow-Origin` header in the response object that you return from your function code.

Enable metrics and error logging
Record latency and error metrics in Amazon CloudWatch, and log errors in Amazon CloudWatch Logs. Standard CloudWatch and CloudWatch Logs pricing applies.

Binary media types
Specify response types that API Gateway should treat as binary data instead of text. To treat all responses as binary data, use `*/*`. If you enter one or more specific types, you must also set the `Content-Type` header in the response object that you return from your function code.
image/png

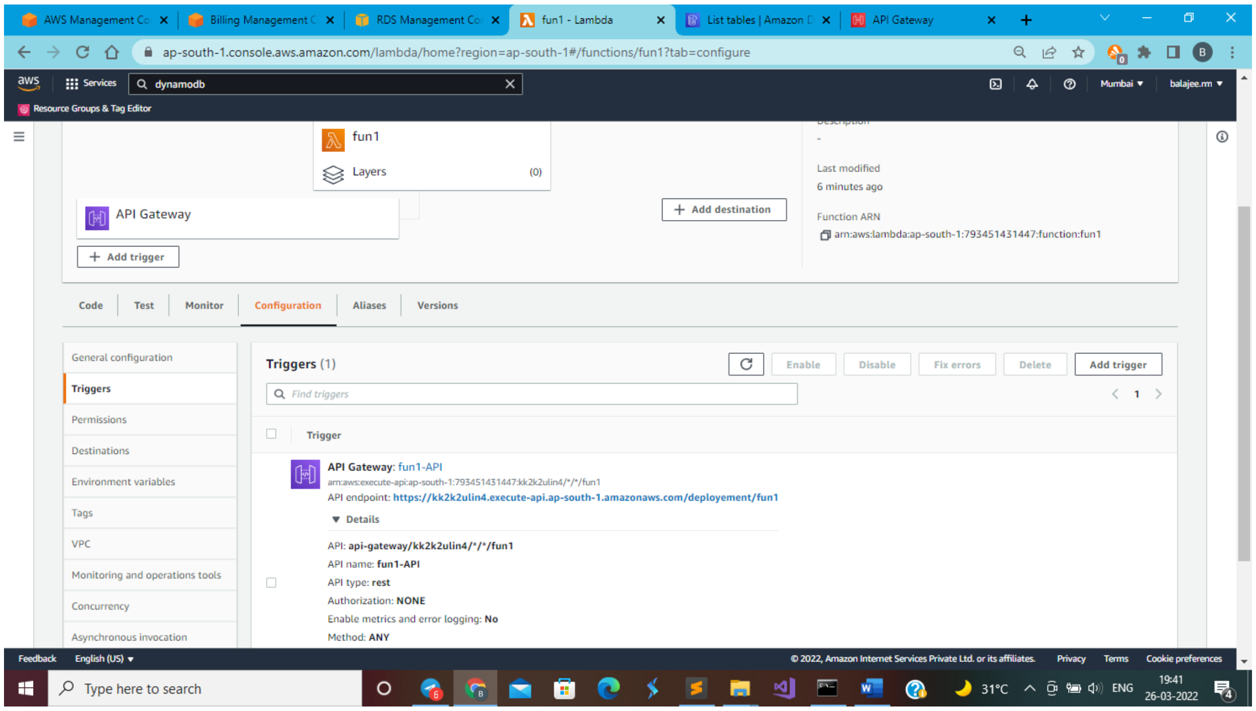
Remove
Add

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Cancel Add

Click add and add the trigger to the lambda function, so that when ever some thing is submitted or called in browser with specific API gateway url, then the lambda function will be triggered.

The configuration of lambda trigger will look like this



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/fun1?tab=configure

Last modified
6 minutes ago

Function ARN
arn:aws:lambda:ap-south-1:793451431447:function:fun1

Triggers (1)

Trigger

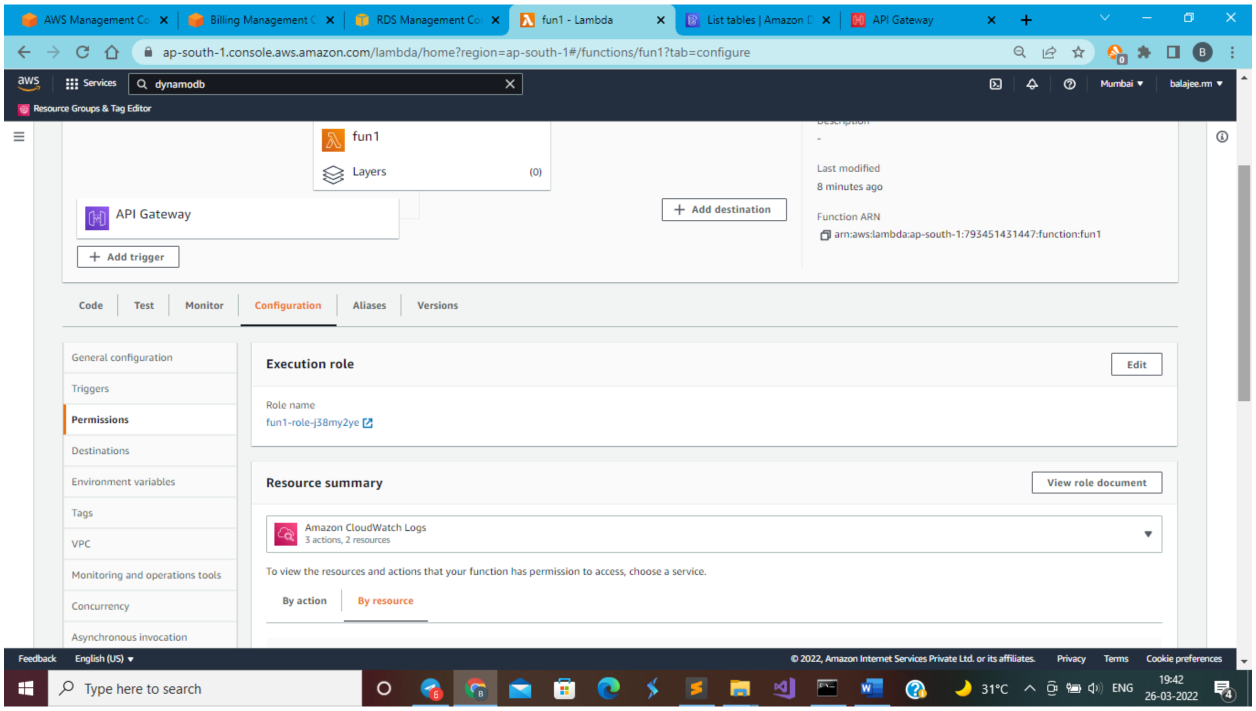
API Gateway: fun1-API
arn:aws:execute-api:ap-south-1:793451431447:kk2k2ulin4/*/*/fun1
API endpoint: https://kk2k2ulin4.execute-api.ap-south-1.amazonaws.com/deployement/fun1

▼ Details

API: api-gateway/kk2k2ulin4/*/*/fun1
API name: fun1-API
API type: rest
Authorization: NONE
Enable metrics and error logging: No
Method: ANY

If you see here, specific url with any method (restfull api methods) can trigger this lambda function.

Under “configurations” go to “permission” on the left and click on the role to open in the new window



Last modified
8 minutes ago

Function ARN
arn:aws:lambda:ap-south-1:793451431447:function:fun1

Execution role

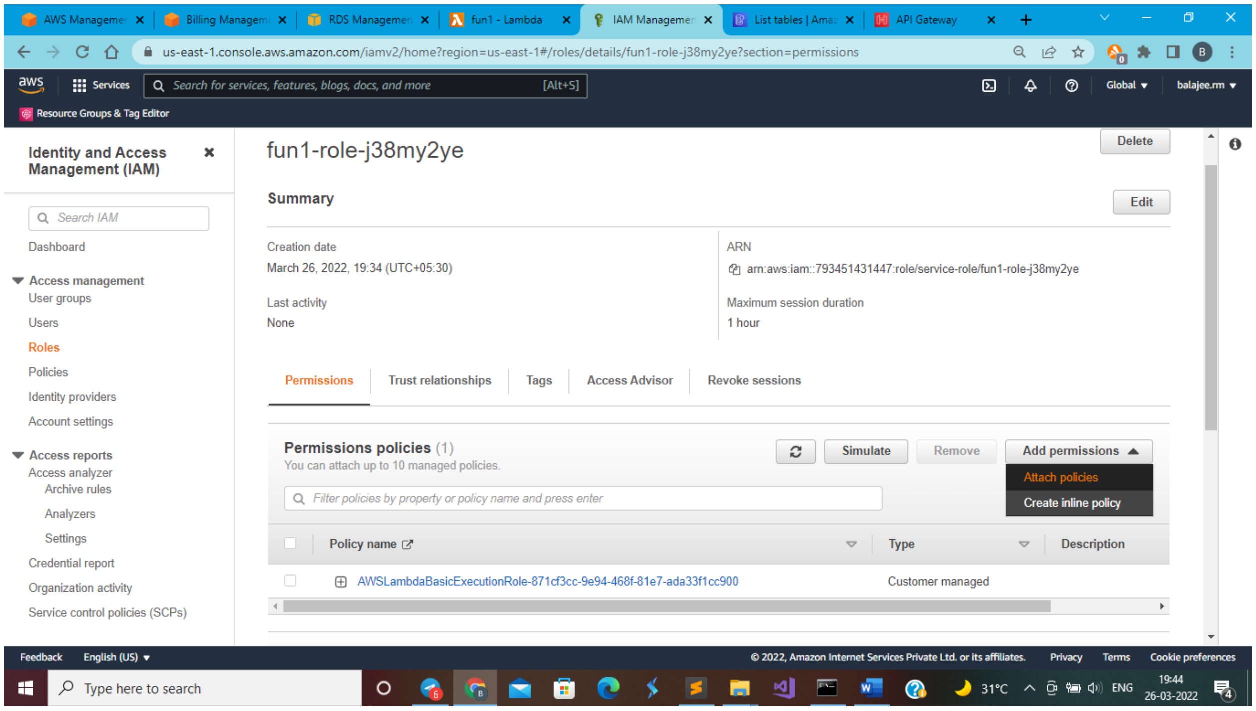
Role name
fun1-role-j38my2ye

Resource summary

Amazon CloudWatch Logs
3 actions, 2 resources

To view the resources and actions that your function has permission to access, choose a service.

Click on the “add permissions” and “attach policies” to permit lambda function to access dynamodb.



us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/fun1-role-j38my2ye?section=permissions

Identity and Access Management (IAM)

fun1-role-j38my2ye

Summary

Creation date
March 26, 2022, 19:34 (UTC+05:30)

ARN
arn:aws:iam::793451431447:role/service-role/fun1-role-j38my2ye

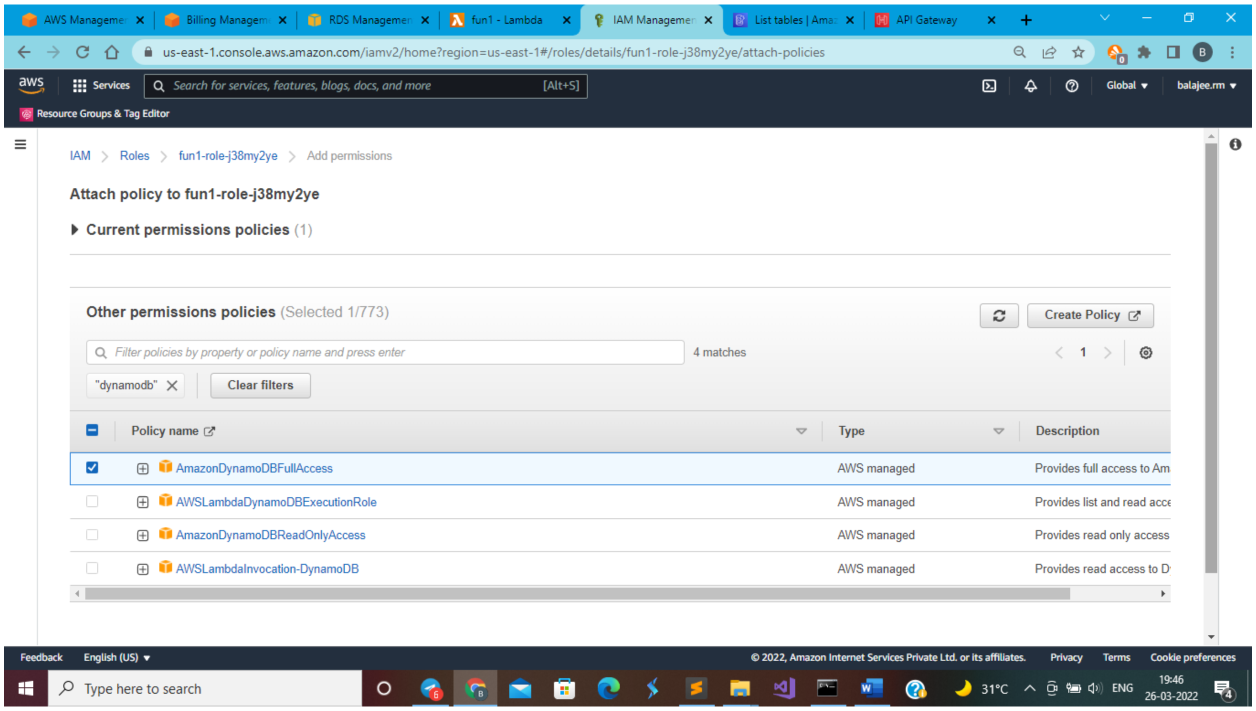
Last activity
None

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (1)
You can attach up to 10 managed policies.

Policy name	Type	Description
AWSLambdaBasicExecutionRole-871cf3cc-9e94-468f-81e7-ada33f1cc900	Customer managed	



us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/fun1-role-j38my2ye/attach-policies

IAM > Roles > fun1-role-j38my2ye > Add permissions

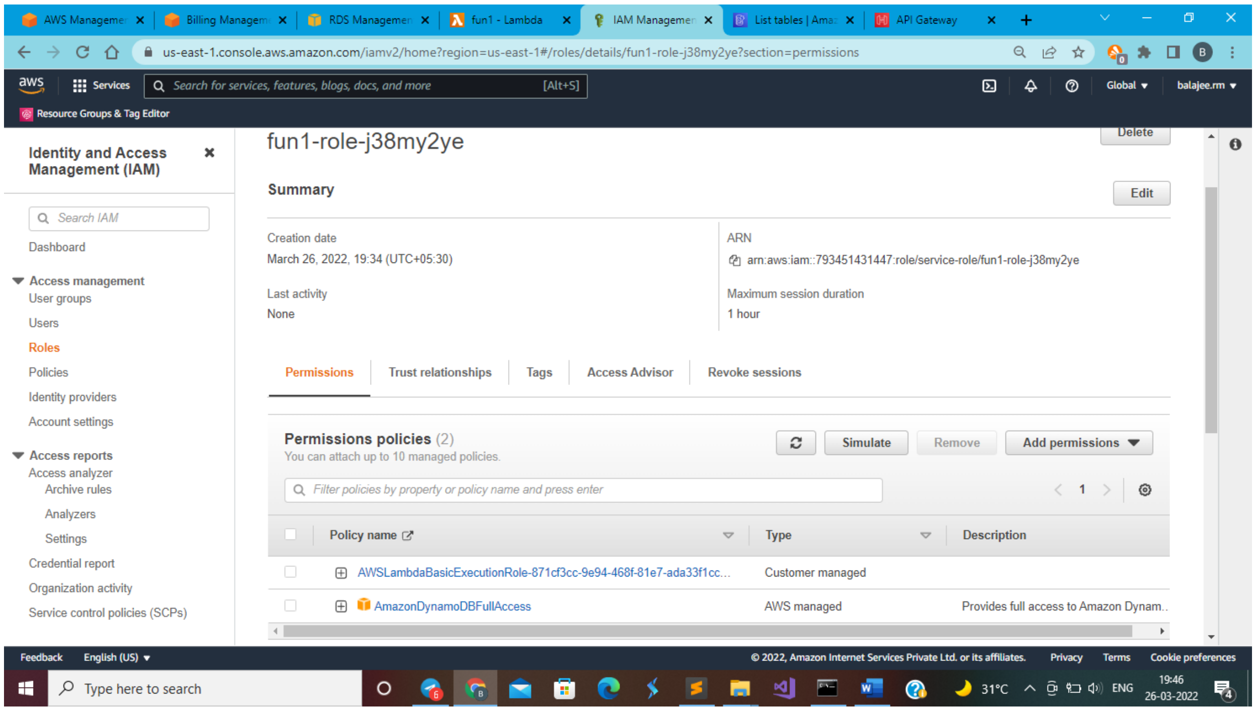
Attach policy to fun1-role-j38my2ye

Current permissions policies (1)

Other permissions policies (Selected 1/773)

"dynamodb" — 4 matches

	Policy name	Type	Description
✓	AmazonDynamoDBFullAccess	AWS managed	Provides full access to Am
	AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read acce
	AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access
	AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to D


us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/fun1-role-j38my2ye?section=permissions

Identity and Access Management (IAM)

fun1-role-j38my2ye

Summary

Creation date	ARN
March 26, 2022, 19:34 (UTC+05:30)	arn:aws:iam::793451431447:role/service-role/fun1-role-j38my2ye
Last activity	Maximum session duration
None	1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (2)
You can attach up to 10 managed policies.

Policy name	Type	Description
AWSLambdaBasicExecutionRole-871cf3cc-9e94-468f-81e7-ada33f1cc...	Customer managed	
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon Dynam...

© 2022, Amazon Internet Services Private Ltd. or its affiliates.

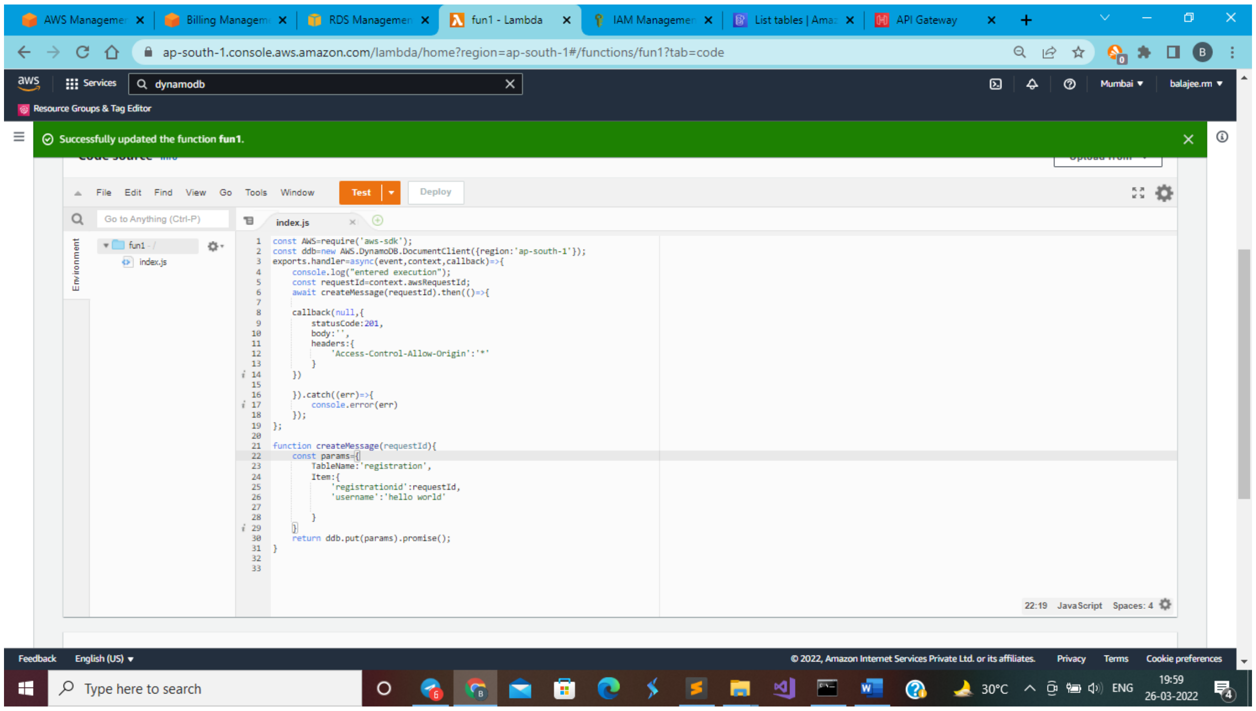
After adding come back to lambda and past the below code for writing in dynamodb

Remember to edit the “region” of dynamodb table and params with table name

Code

```
const AWS=require('aws-sdk');
const ddb=new AWS.DynamoDB.DocumentClient({region:'ap-south-1'});
exports.handler=async(event,context,callback)=>{
  console.log(“entered execution”);
  const requestId=context.awsRequestId;
  await createMessage(requestId).then(()=>{

  callback(null,{
    statusCode:201,
    body:'',
    headers:{
      'Access-Control-Allow-Origin':'*'
    }
  })

  }).catch((err)=>{
    console.error(err)
  });
};
```

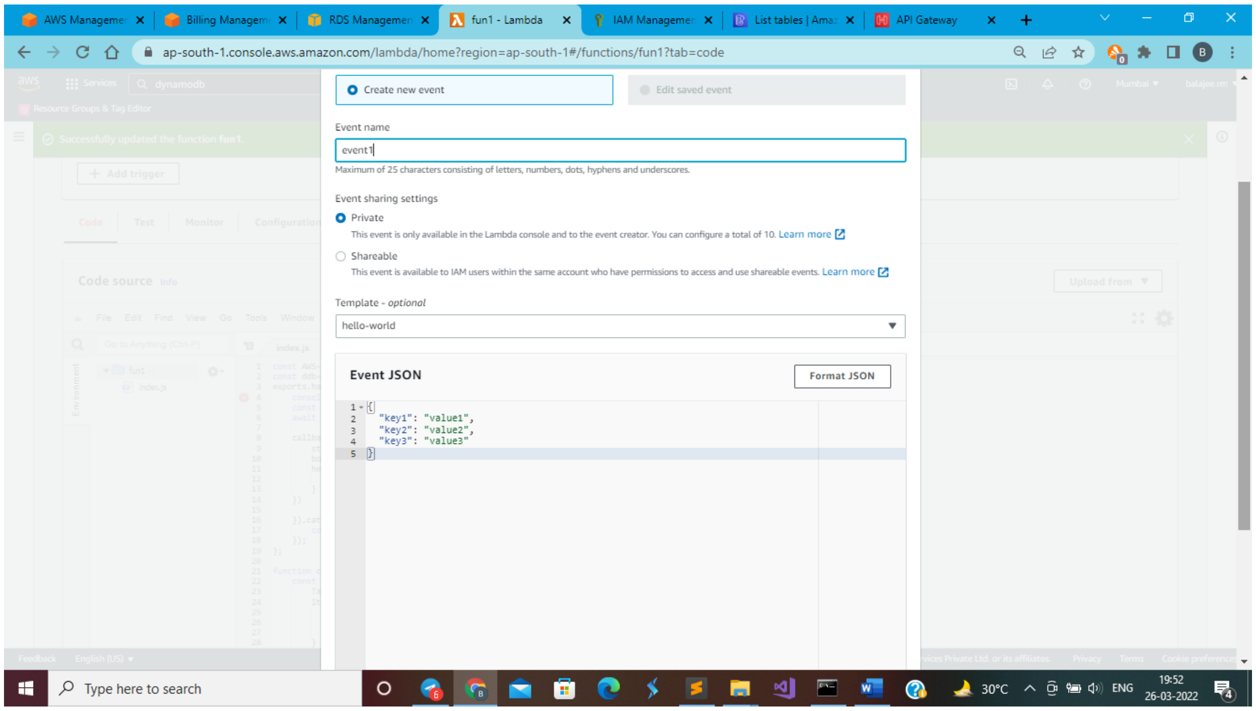
```
function createMessage(requestId){
  const params={
    TableName:'registration',
    Item:{
      'registrationid':requestId,
      'username':'hello world'


    }
  }
  return ddb.put(params).promise();
}
```



Successfully updated the function fun1.

```
const AWS=require('aws-sdk');
const ddb=new AWS.DynamoDB.DocumentClient({region:'ap-south-1'});
exports.handler=async(event,context,callback)=>{
    console.log("entered execution");
    const requestId=context.awsRequestId;
    await createMessage(requestId).then(()=>{

    callback(null,{
        statusCode:201,
        body:'',
        headers:{
            'Access-Control-Allow-Origin':'*'
        }
    })

    }).catch((err)=>{
        console.error(err)
    });
};

function createMessage(requestId){
    const params={
        TableName:'registration',
        Item:{
            'registrationid':requestId,
            'username':'hello world'

        }
    }
    return ddb.put(params).promise();
}
```

Click on "deploy" to save the changes and then click on "test" to provide the test event name.



Create new event

Edit saved event

Event name

event1

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

● Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

○ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

Template - *optional*

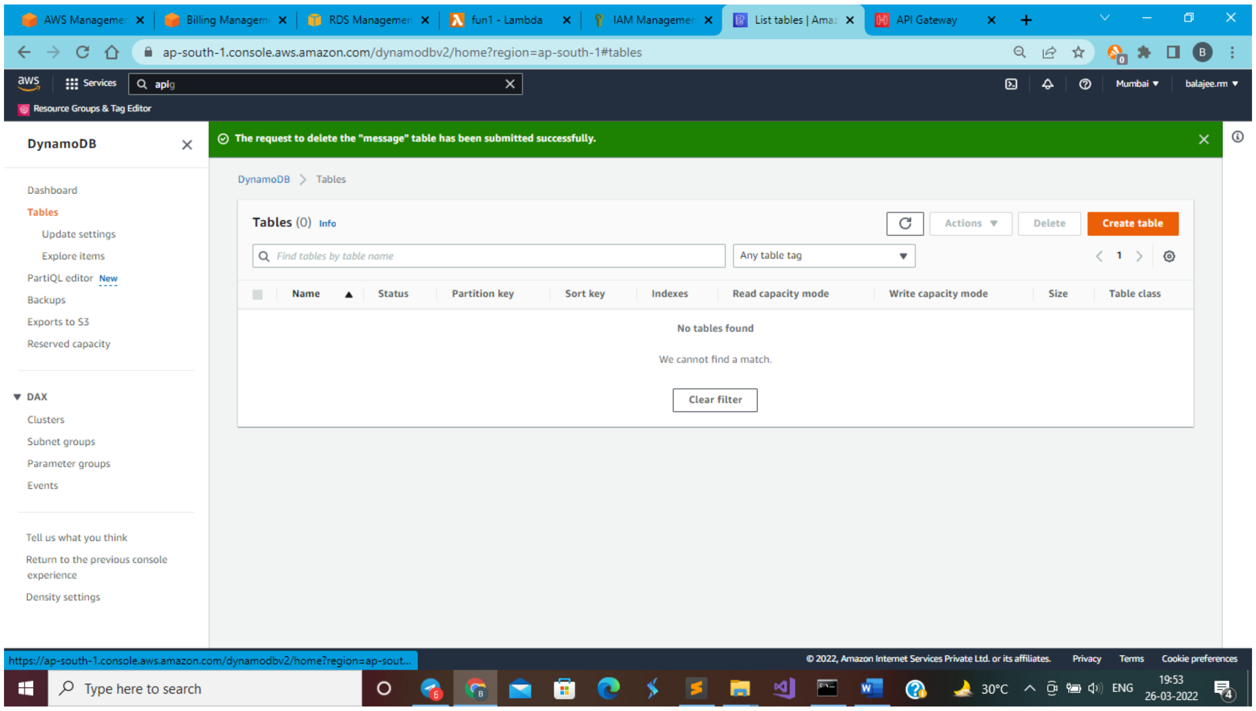
hello-world

Event JSON

Format JSON

```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

Go to dynamodb and click on "create table"



The request to delete the "message" table has been submitted successfully.

DynamoDB > Tables

Tables (0) Info

Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode	Size	Table class

No tables found

We cannot find a match.

Clear filter

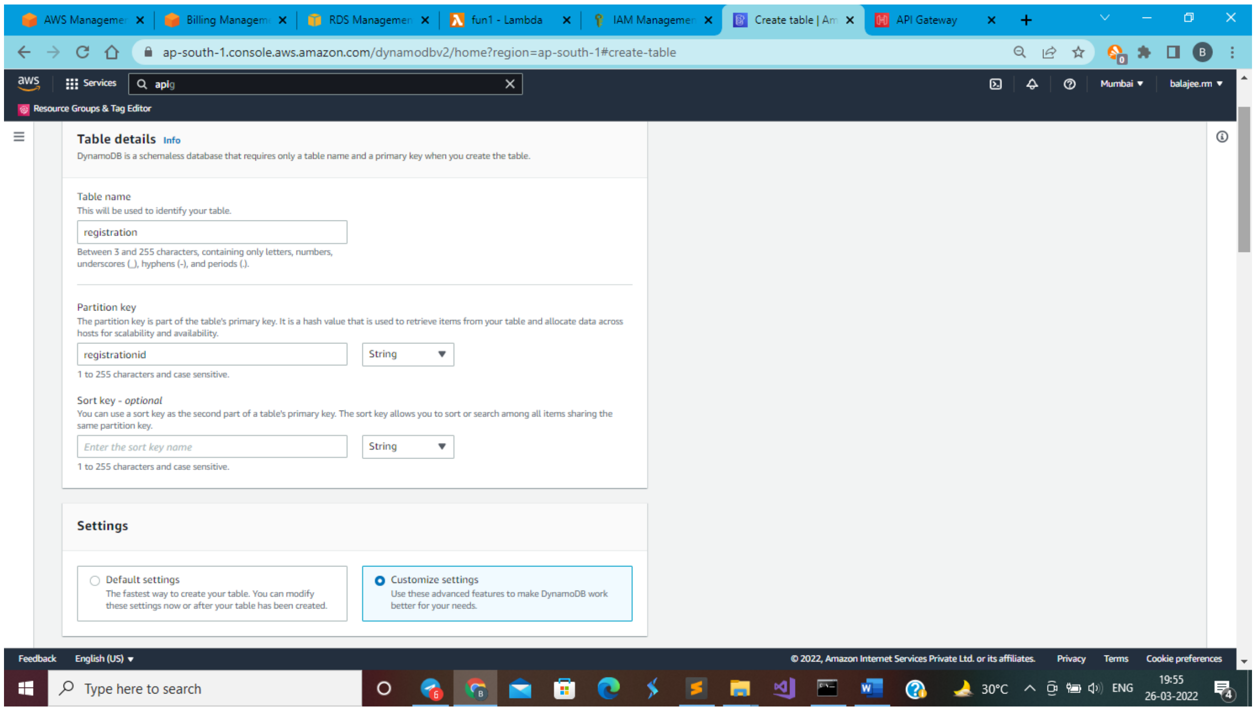

ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#create-table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
registration
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
registrationid — String
1 to 255 characters and case sensitive.

Sort key - *optional*
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
Enter the sort key name — String
1 to 255 characters and case sensitive.

Settings

[ ] Default settings
The fastest way to create your table. You can modify these settings now or after your table has been created.

[✓ Customize settings]
Use these advanced features to make DynamoDB work better for your needs.

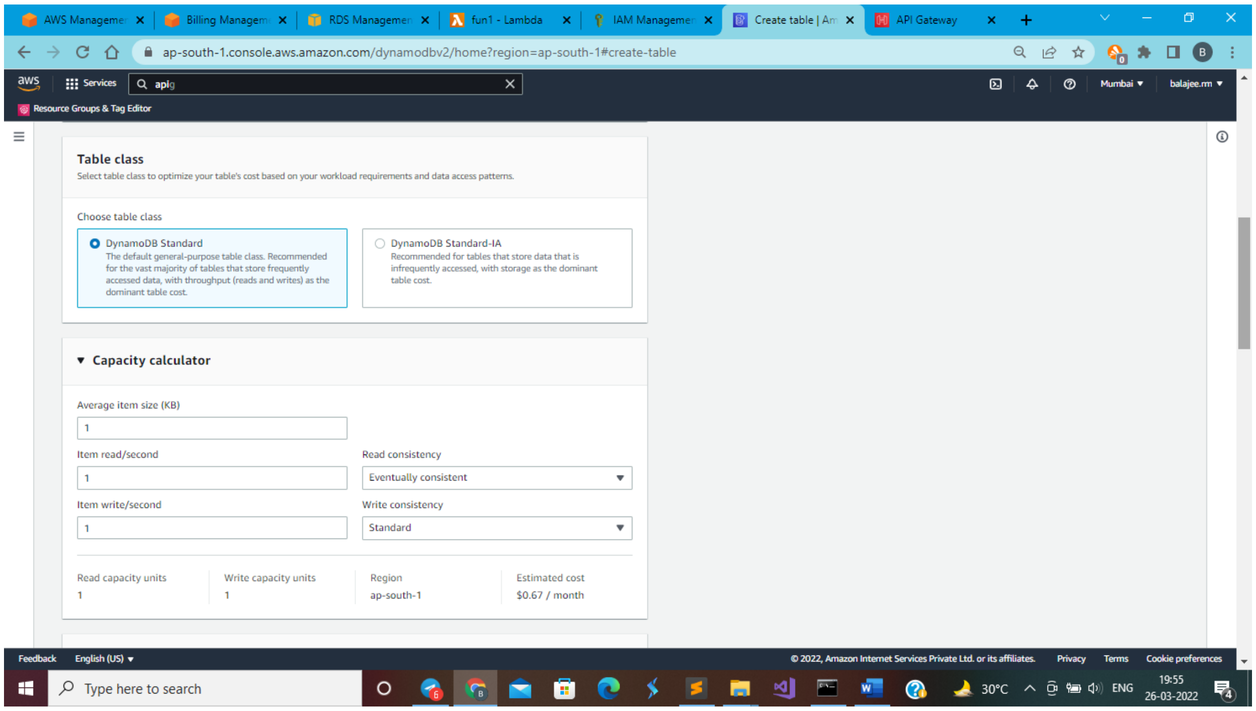

ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#create-table

Table class

Select table class to optimize your table's cost based on your workload requirements and data access patterns.

Choose table class

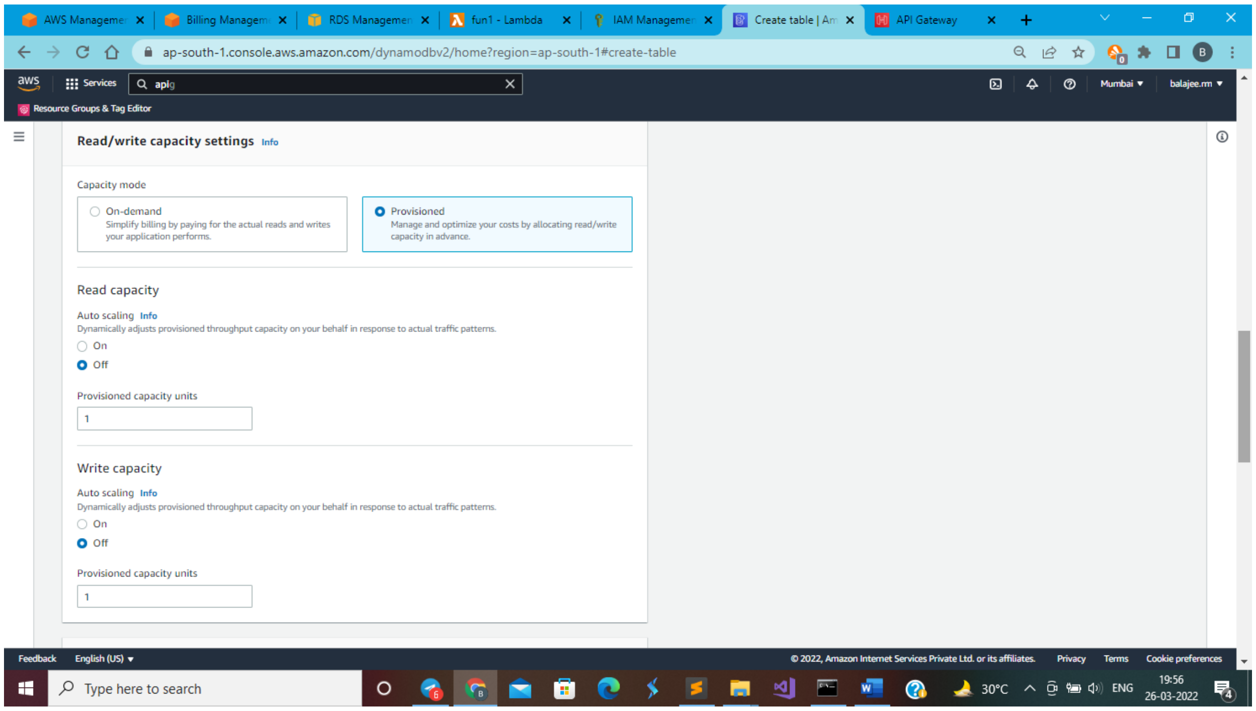
[✓ DynamoDB Standard]
The default general-purpose table class. Recommended for the vast majority of tables that store frequently accessed data, with throughput (reads and writes) as the dominant table cost.

[ ] DynamoDB Standard-IA
Recommended for tables that store data that is infrequently accessed, with storage as the dominant table cost.

Capacity calculator

Average item size (KB): 1
Item read/second: 1
Read consistency: Eventually consistent
Item write/second: 1
Write consistency: Standard

Read capacity units	Write capacity units	Region	Estimated cost
1	1	ap-south-1	$0.67 / month


Read/write capacity settings Info

Capacity mode

On-demand
Simplify billing by paying for the actual reads and writes your application performs.

[✓ Provisioned]
Manage and optimize your costs by allocating read/write capacity in advance.

Read capacity

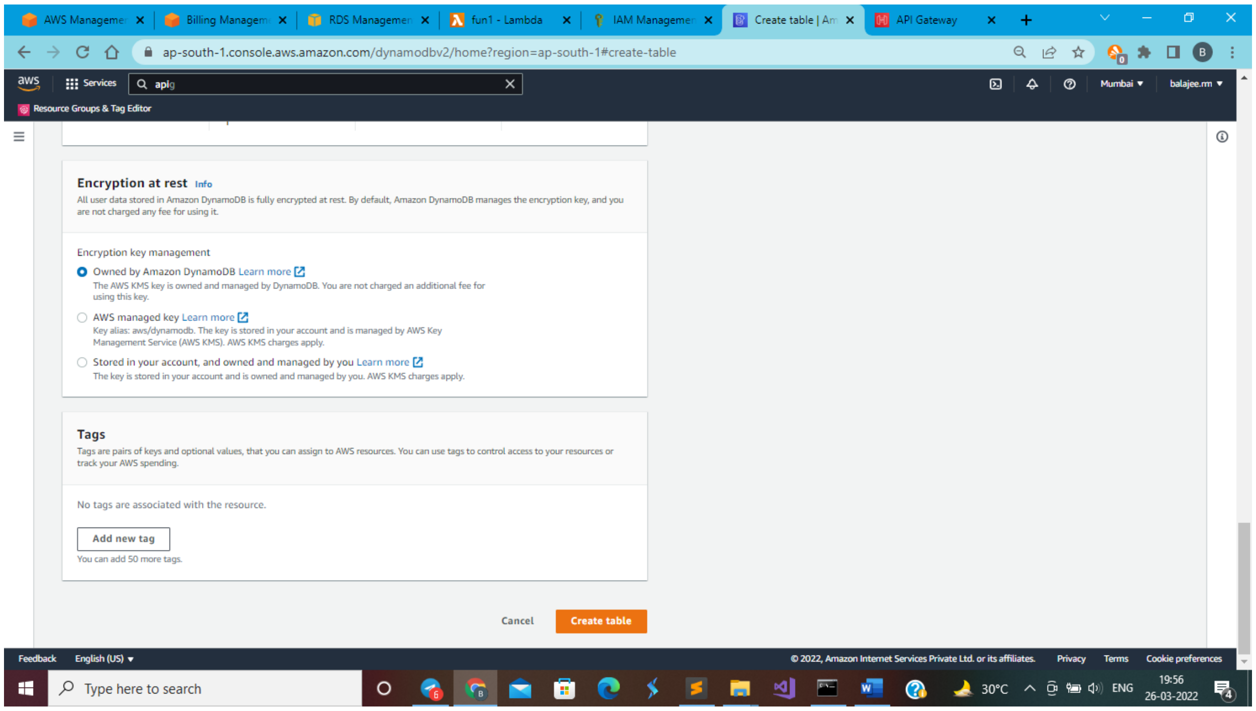
Auto scaling Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.
On [✓ Off]

Provisioned capacity units: 1

Write capacity

Auto scaling Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.
On [✓ Off]

Provisioned capacity units: 1


Encryption at rest Info
All user data stored in Amazon DynamoDB is fully encrypted at rest. By default, Amazon DynamoDB manages the encryption key, and you are not charged any fee for using it.

Encryption key management

[✓ Owned by Amazon DynamoDB] Learn more
The AWS KMS key is owned and managed by DynamoDB. You are not charged an additional fee for using this key.

AWS managed key Learn more
Key alias: aws/dynamodb. The key is stored in your account and is managed by AWS Key Management Service (AWS KMS). AWS KMS charges apply.

Stored in your account, and owned and managed by you Learn more
The key is stored in your account and is owned and managed by you. AWS KMS charges apply.

Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

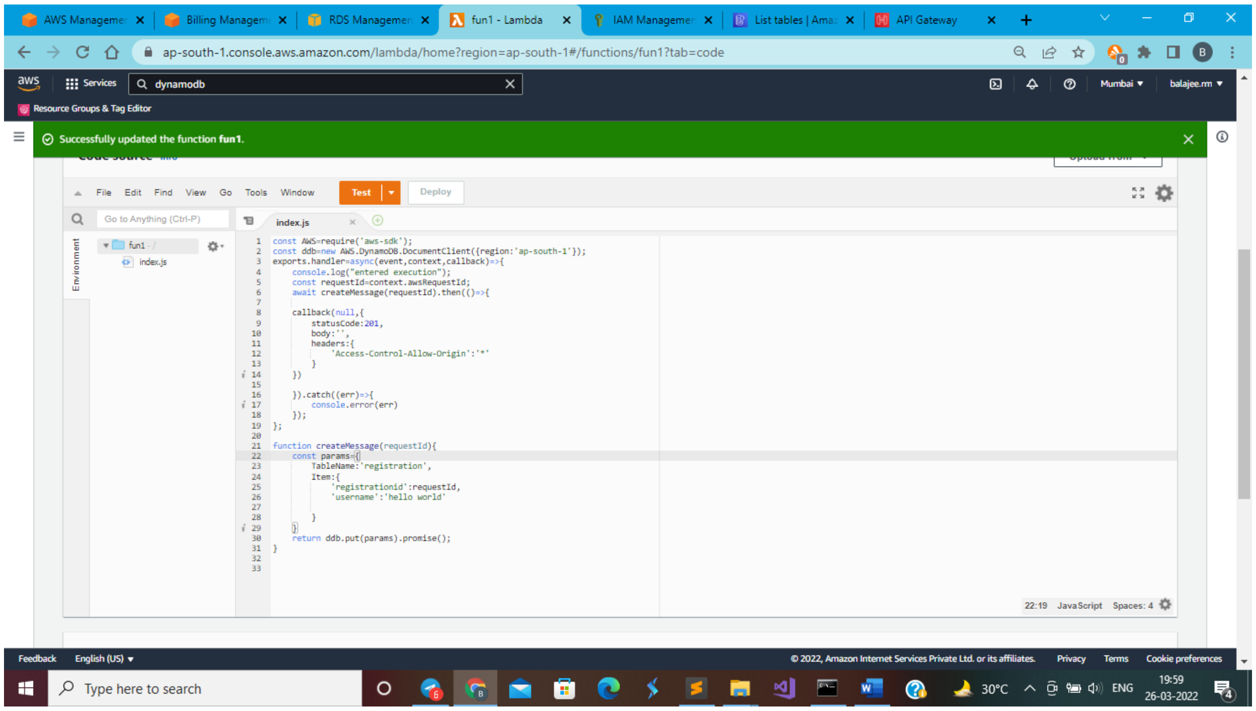
No tags are associated with the resource.

Add new tag
You can add 50 more tags.

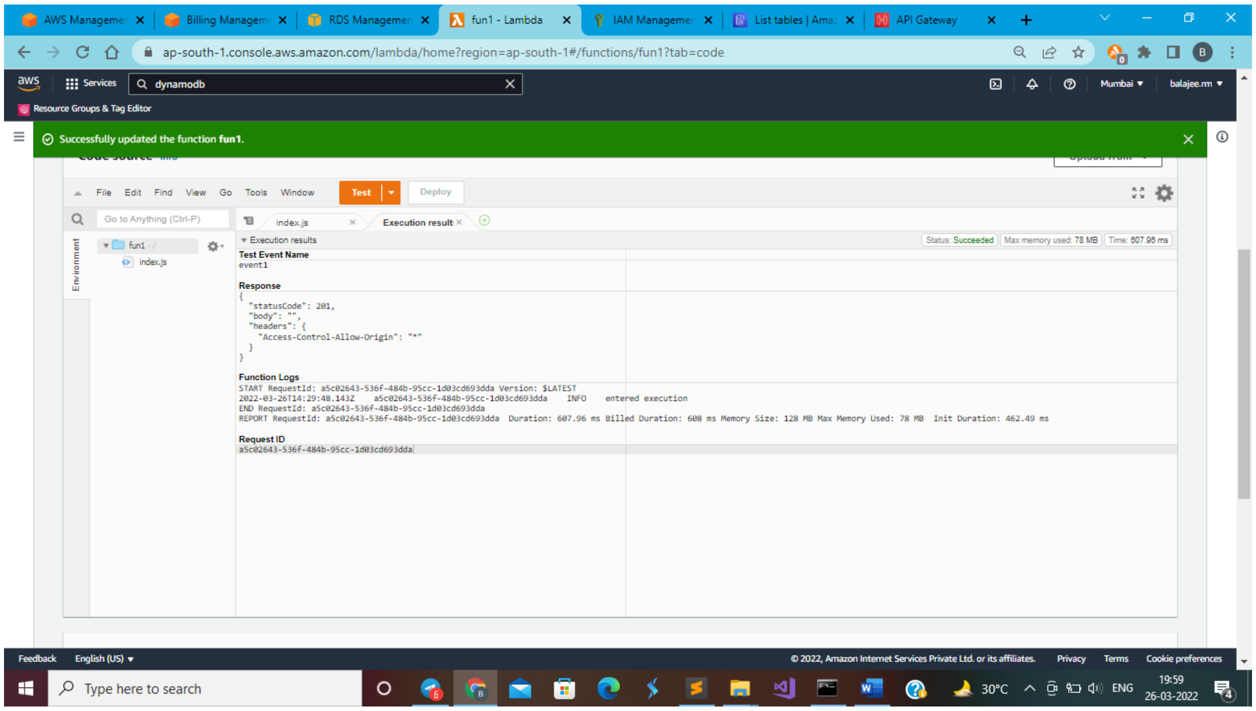
Cancel | Create table

Click on “create table” and create it

Now go to lambda and run the function by manually clicking the “test” button, so that, it will create a sample data in dynamodb.


Successfully updated the function **fun1**.

```
const AWS=require('aws-sdk');
const ddb=new AWS.DynamoDB.DocumentClient({region:'ap-south-1'});
exports.handler=async(event,context,callback)=>{
    console.log("entered execution");
    const requestId=context.awsRequestId;
    await createMessage(requestId).then(()=>{

    callback(null,{
        statusCode:201,
        body:'',
        headers:{
            'Access-Control-Allow-Origin':'*'
        }
    })

    }).catch((err)=>{
        console.error(err)
    });
};

function createMessage(requestId){
    const params={
        TableName:'registration',
        Item:{
            'registrationid':requestId,
            'username':'hello world'

        }
    }
    return ddb.put(params).promise();
}
```


Successfully updated the function **fun1**.

Execution results — Status: Succeeded | Max memory used: 78 MB | Time: 607.96 ms

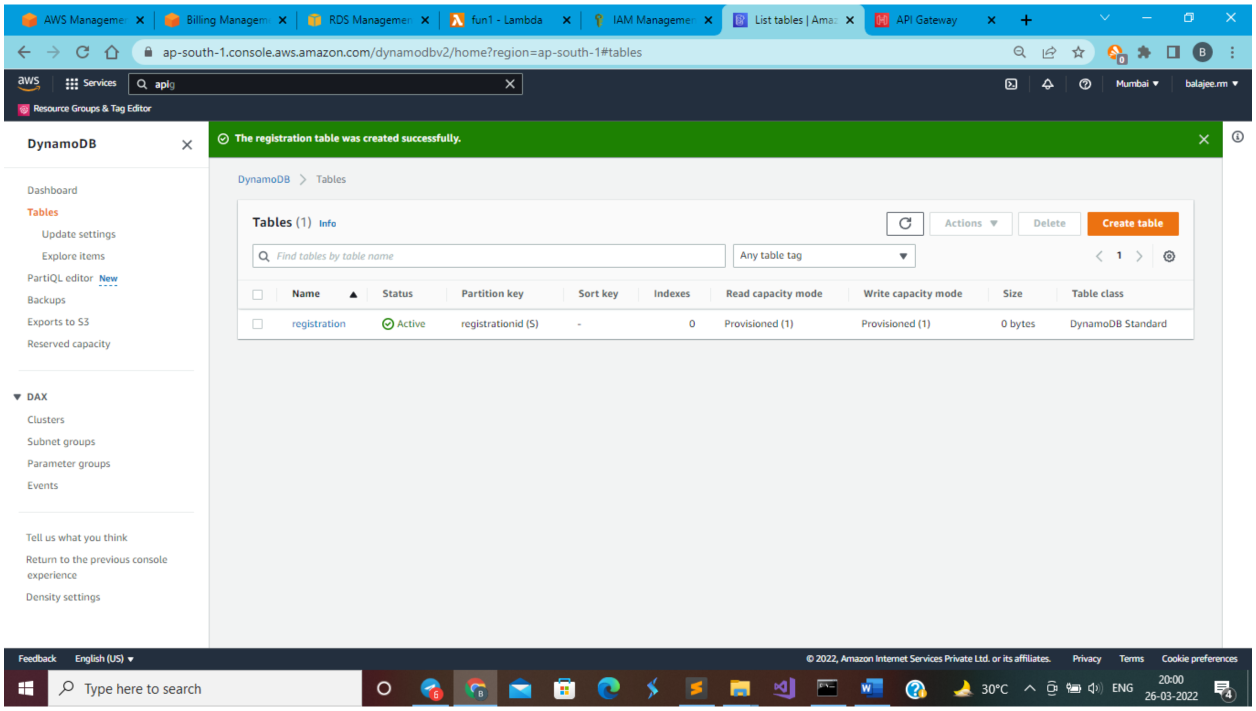
Test Event Name
```
event1
```

Response
```
{
  "statusCode": 201,
  "body": "",
  "headers": {
    "Access-Control-Allow-Origin": "*"
  }
}
```

Function Logs
```
START RequestId: a5c02643-536f-484b-95cc-1d03cd693dda Version: $LATEST
2022-03-26T14:29:48.143Z	a5c02643-536f-484b-95cc-1d03cd693dda	INFO	entered execution
END RequestId: a5c02643-536f-484b-95cc-1d03cd693dda
REPORT RequestId: a5c02643-536f-484b-95cc-1d03cd693dda	Duration: 607.96 ms	Billed Duration: 608 ms	Memory Size: 128 MB	Max Memory Used: 78 MB	Init Duration: 462.49 ms
```

Request ID
```
a5c02643-536f-484b-95cc-1d03cd693dda
```

Go to dynamodb and check the result



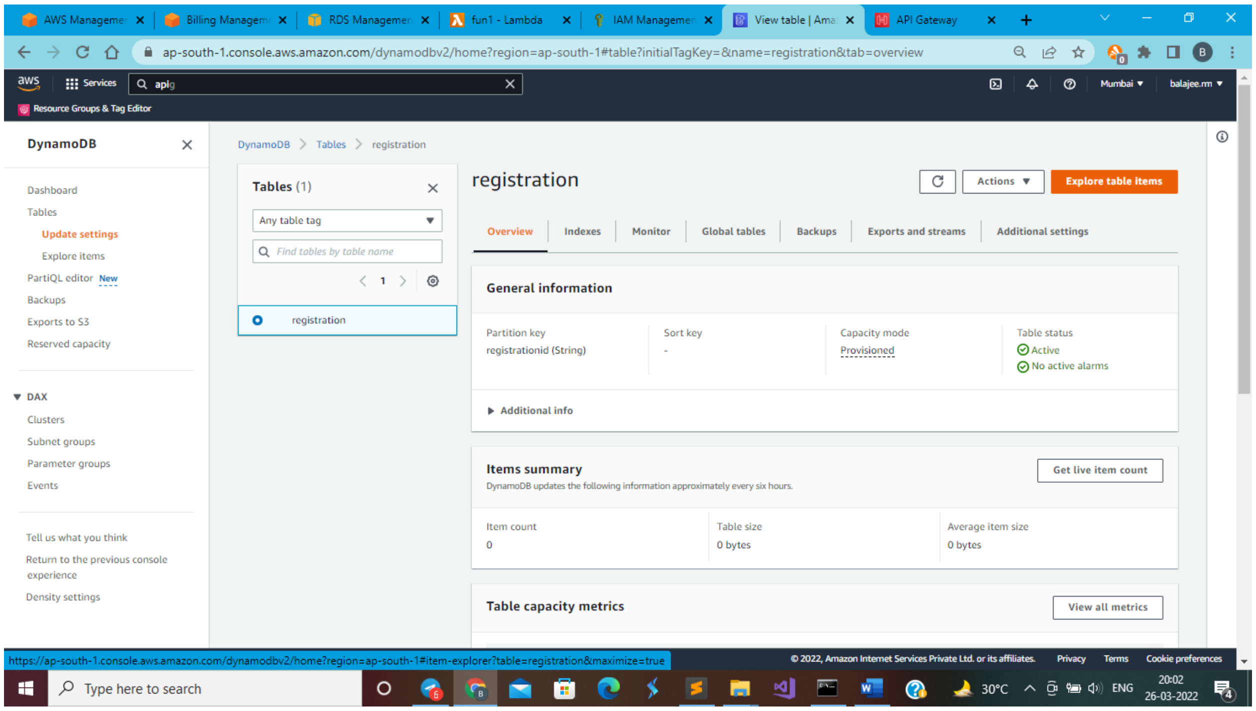
The registration table was created successfully.

DynamoDB > Tables

Tables (1) Info

	Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode	Size	Table class
☐	registration	✓ Active	registrationid (S)	-	0	Provisioned (1)	Provisioned (1)	0 bytes	DynamoDB Standard

Click on the table created



DynamoDB > Tables > registration

registration

General information

Partition key	Sort key	Capacity mode	Table status
registrationid (String)	-	Provisioned	✓ Active ✓ No active alarms

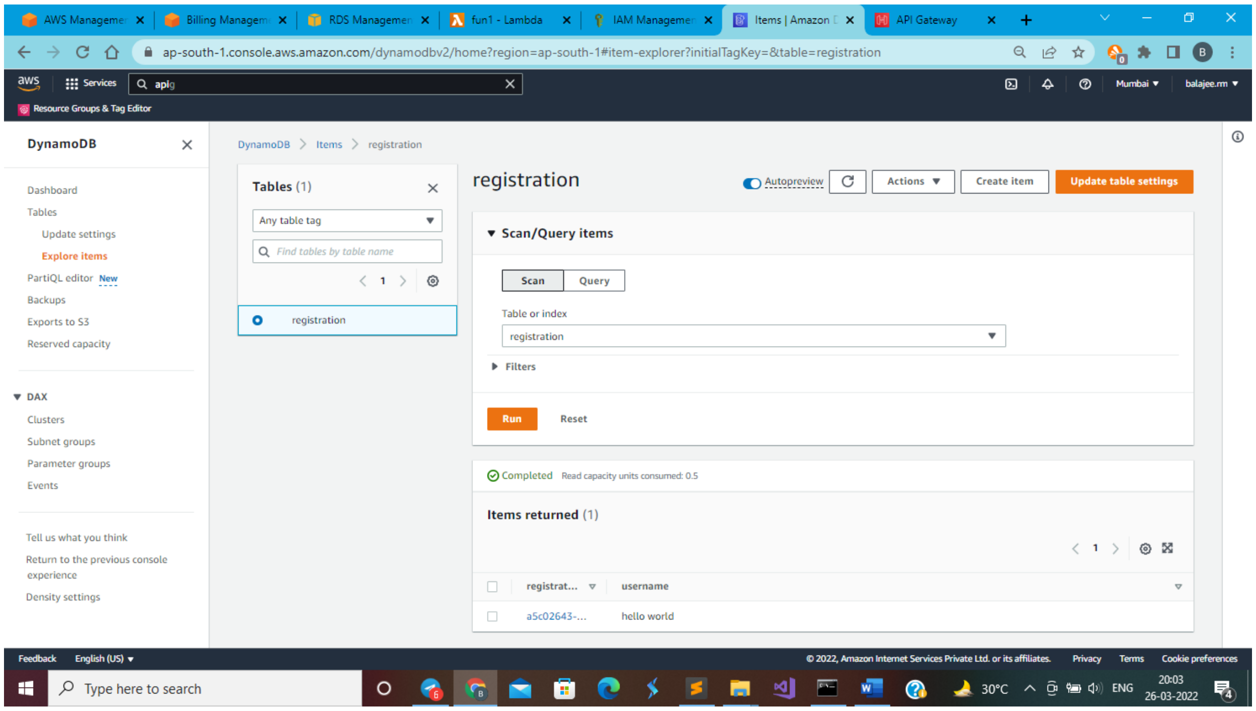
Items summary

DynamoDB updates the following information approximately every six hours.

Item count	Table size	Average item size
0	0 bytes	0 bytes

Table capacity metrics

Click on explore table items



registration

Table or index: registration

Completed Read capacity units consumed: 0.5

Items returned (1)

	registrat...	username
	a5c02643-...	hello world

The created record is shown above

Now go to lambda to edit the code to receive the post method “parameter” values

Code

```
const AWS=require('aws-sdk');
const querystring = require('querystring')
const ddb=new AWS.DynamoDB.DocumentClient({region:'ap-south-1'});
exports.handler=async(event,context,callback)=>{
  console.log("entered execution");

  if (event.body !== null && event.body !== undefined) {
    console.log("parameter came");
    var data = JSON.parse(event.body);
    console.log(data.id);
    console.log(data.name);

    const requestId=context.awsRequestId;
    await createMessage(requestId, data.id, data.name).then(()=>{

    callback(null,{
      statusCode:201,
      body:'',
      headers:{
        'Access-Control-Allow-Origin':'*'
```

```
            }
        })

        }).catch((err)=>{
            console.error(err)
        });
    }

};

function createMessage(requestId, id, name){
    const params={
        TableName:'registration',
        Item:{
            'registrationid': requestId,
            'userid': id,
            'username': name

        }
    }
    return ddb.put(params).promise();
}
```

Check it in dynamodb after making the axios request form react app with parameter (form) values to a specific api gateway url.