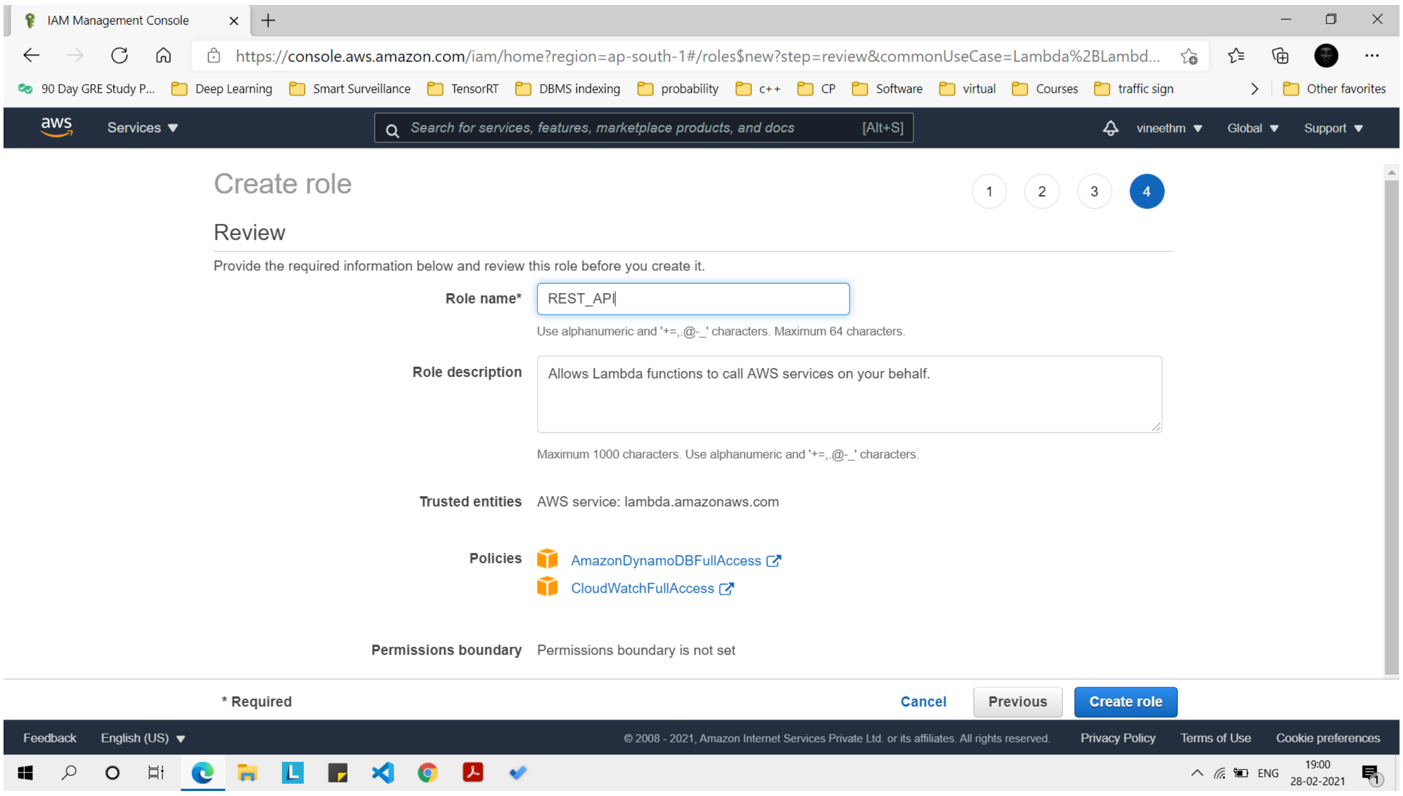
Lab Assignment – 06

AWS API Gateway, Lambda and Dynamo DB; REST API implementation.

API Gateway provides tools for creating and documenting web API's that route HTTP requests to Lambda functions. You can secure the access to your API with authentication and authorization controls. Your APIs can serve traffic over the internet or can be accessible only within your VPC.

Lab Objective: Using AWS API Gateway, connect it with AWS Lambda function and DynamoDB. Using the API Gateway, you can call the AWS lambda function while something is asked through the web browser and HTTP request hits to the DynamoDB table/ database.

Task-1: Create IAM role for Full DynamoDB access and CloudWatch service.


IAM Management Console
https://console.aws.amazon.com/iam/home?region=ap-south-1#/roles$new?step=review&commonUseCase=Lambda%2BLambd...
Services
vineethm
Global
Support
1
2
3
4

Create role

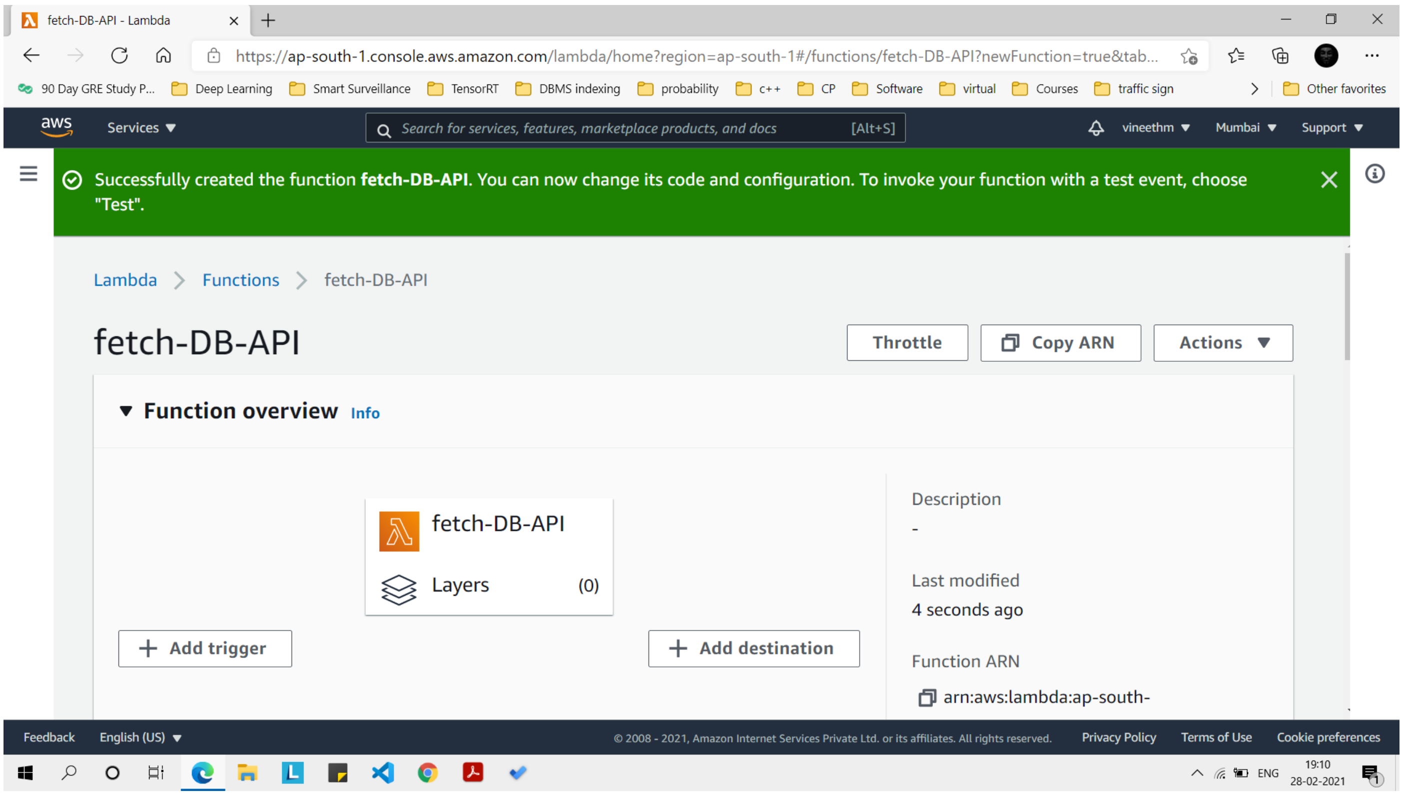
Review

Provide the required information below and review this role before you create it.

Role name*	REST_API
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	Allows Lambda functions to call AWS services on your behalf.
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	AWS service: lambda.amazonaws.com
Policies	AmazonDynamoDBFullAccess CloudWatchFullAccess
Permissions boundary	Permissions boundary is not set

* Required

Task-2: Create Lambda function and provide the above create role.



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

Lambda > Functions > fetch-DB-API

fetch-DB-API

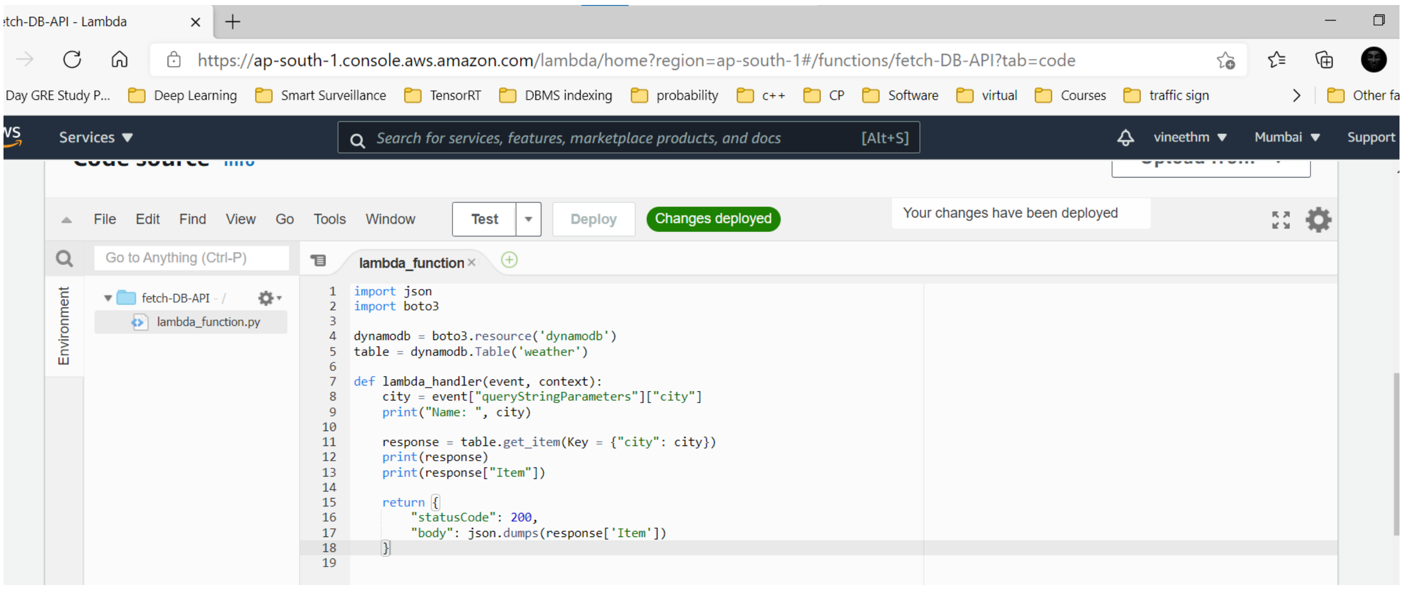
Function overview

Description
-

Last modified
4 seconds ago

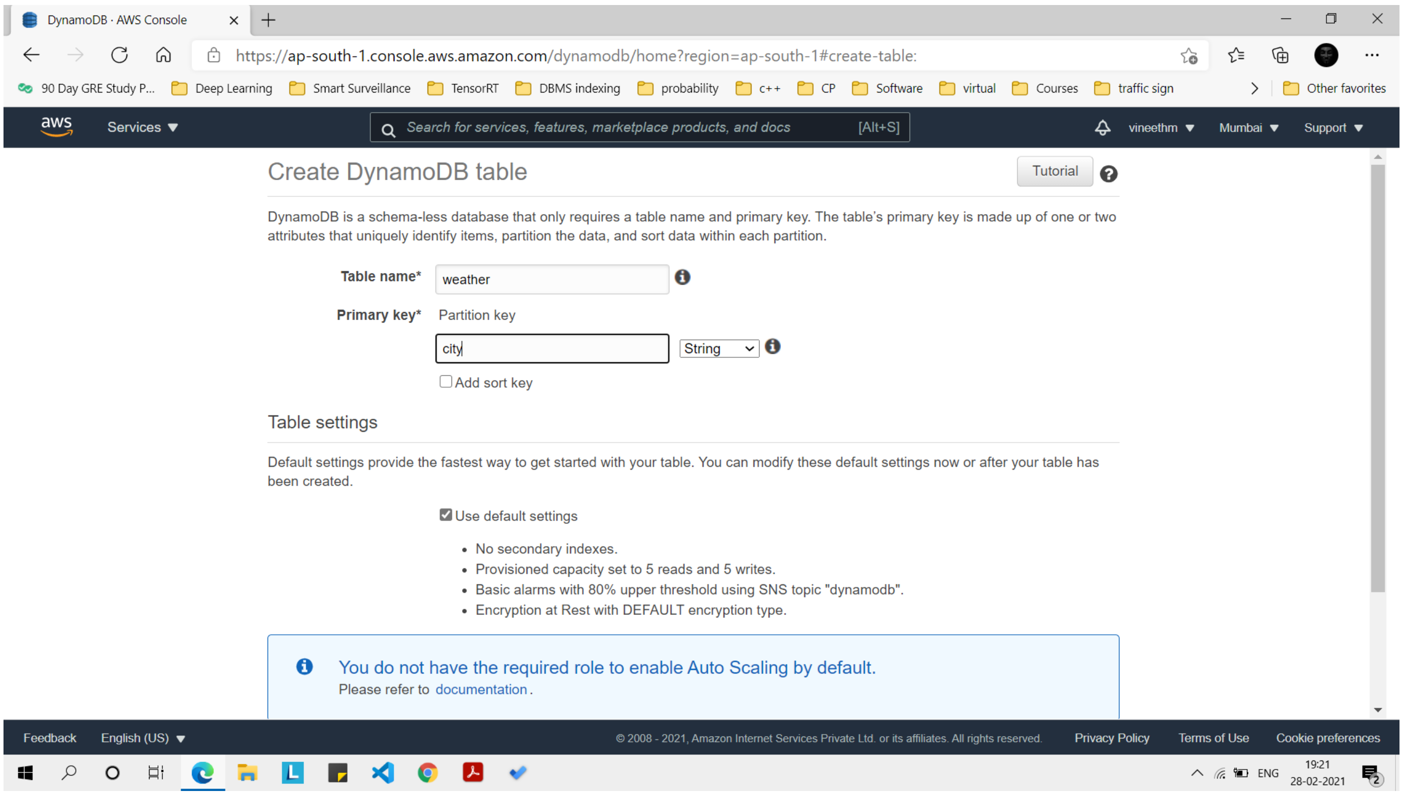
Function ARN
arn:aws:lambda:ap-south-

Task-3: Write the code in the Lambda function.



```
import json
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('weather')

def lambda_handler(event, context):
    city = event["queryStringParameters"]["city"]
    print("Name: ", city)

    response = table.get_item(Key = {"city": city})
    print(response)
    print(response["Item"])

    return {
        "statusCode": 200,
        "body": json.dumps(response['Item'])
    }
```

Task-4: Go to the DynamoDB database server and create a table named weather. Provide the key name as city.



Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*: weather

Primary key*: Partition key: city — String

☐ Add sort key

Table settings

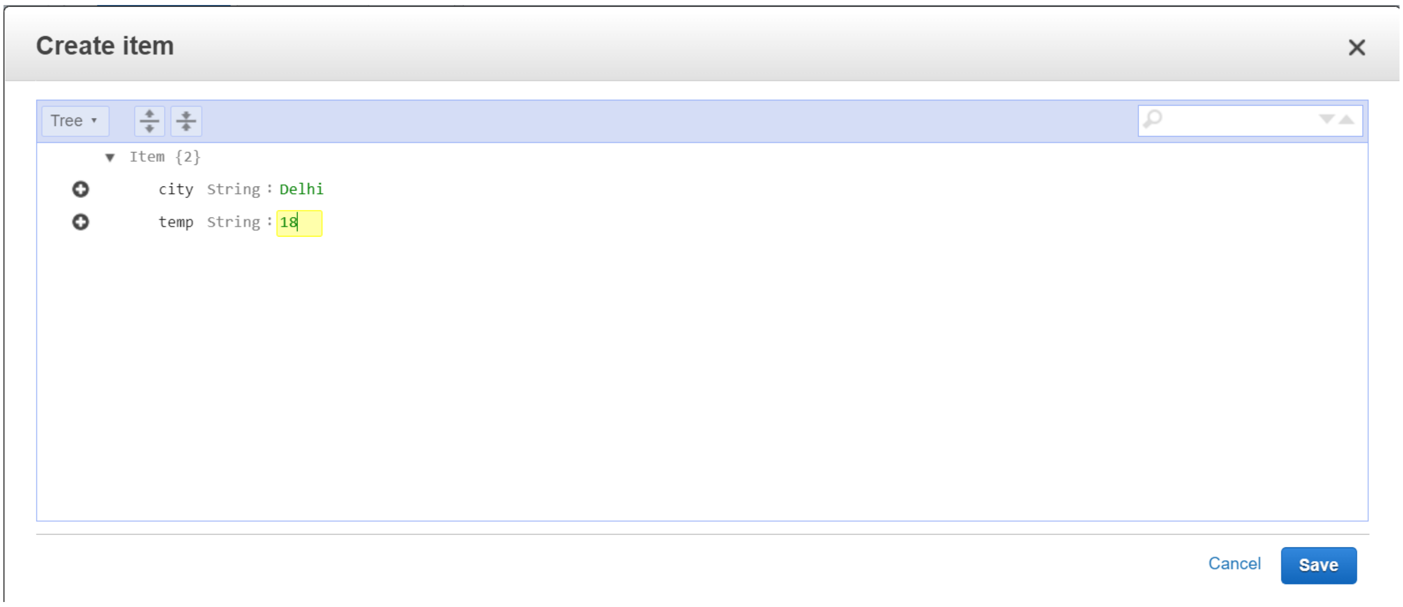
Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☑ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.

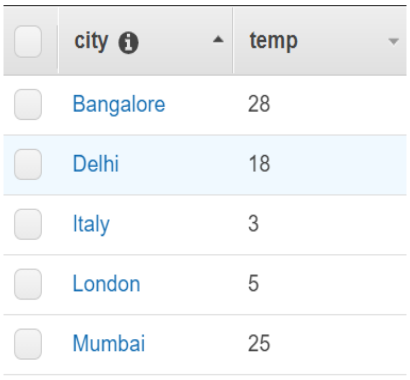
You do not have the required role to enable Auto Scaling by default.
Please refer to documentation.

Task-5: Create the Table with the values given.



Create item

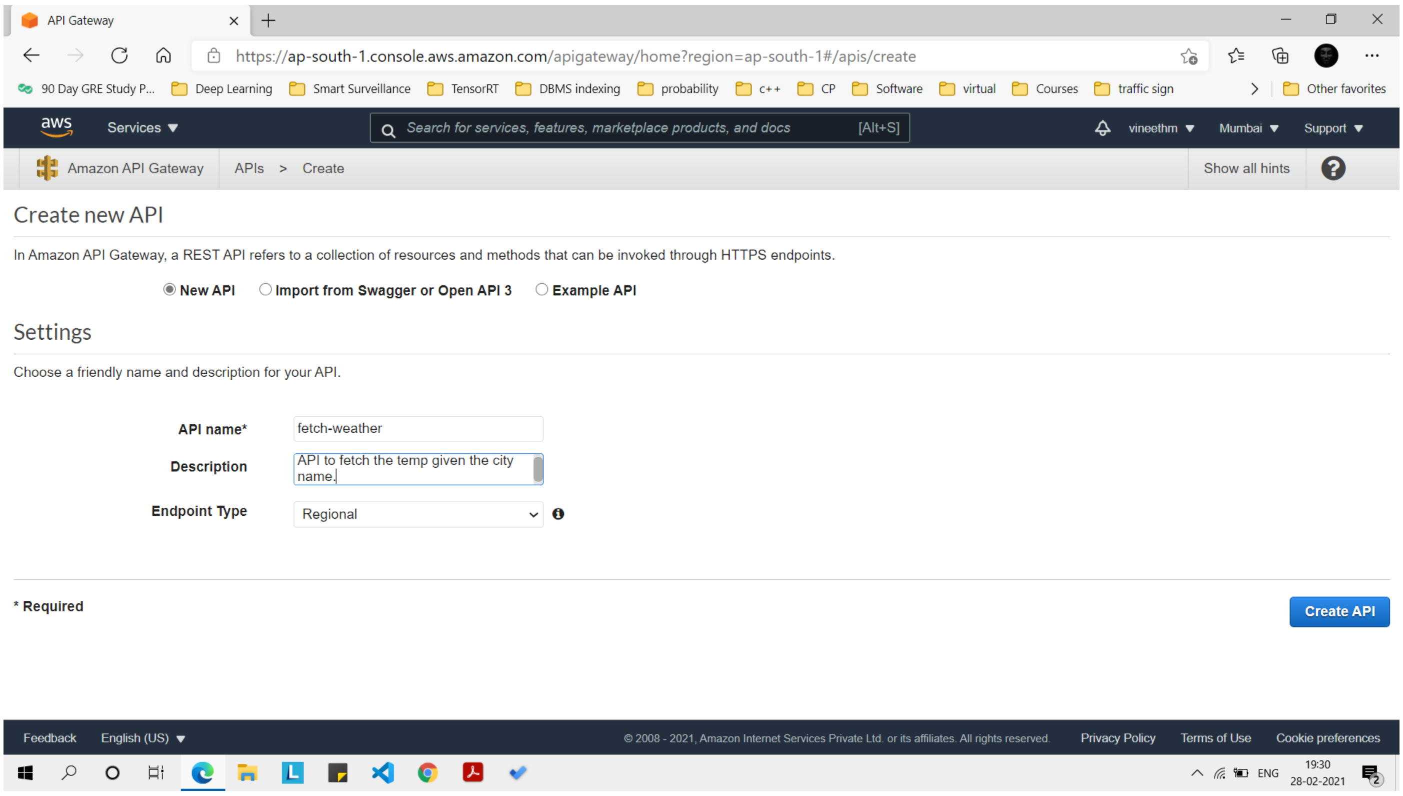
```
▼ Item {2}
    city  String : Delhi
    temp  String : 18
```



city	temp
Bangalore	28
Delhi	18
Italy	3
London	5
Mumbai	25

Task-6: Create an Amazon API Gateway by selecting REST Architecture. API name could be anything as per your choice.

Note: Endpoint type should be regional.



https://ap-south-1.console.aws.amazon.com/apigateway/home?region=ap-south-1#/apis/create

Amazon API Gateway > APIs > Create

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

[✓ New API] Import from Swagger or Open API 3 Example API

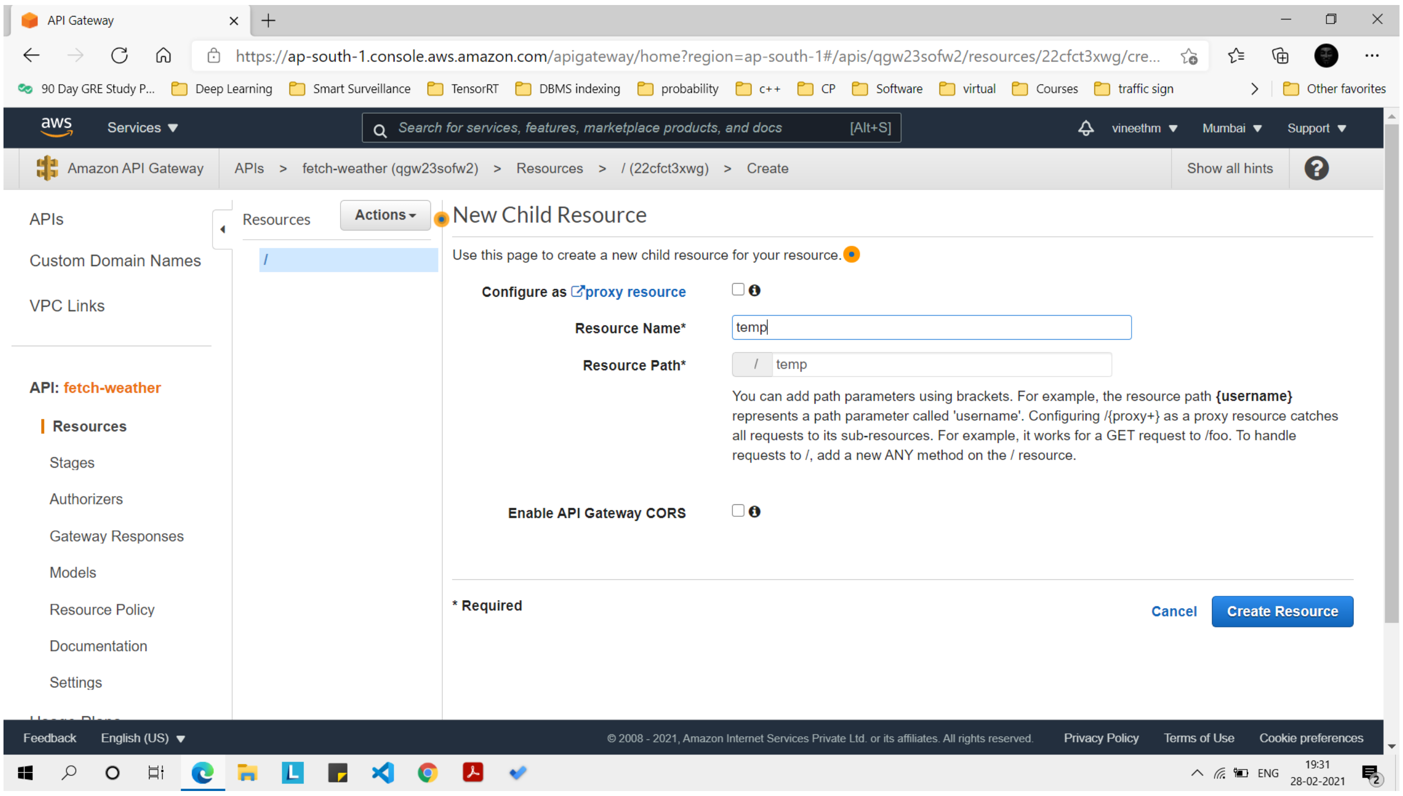
Settings

Choose a friendly name and description for your API.

API name*	fetch-weather
Description	API to fetch the temp given the city name.
Endpoint Type	Regional

* Required

Task-7: Create the API resource.



New Child Resource

Use this page to create a new child resource for your resource.

Configure as proxy resource ☐

Resource Name* temp

Resource Path* / temp

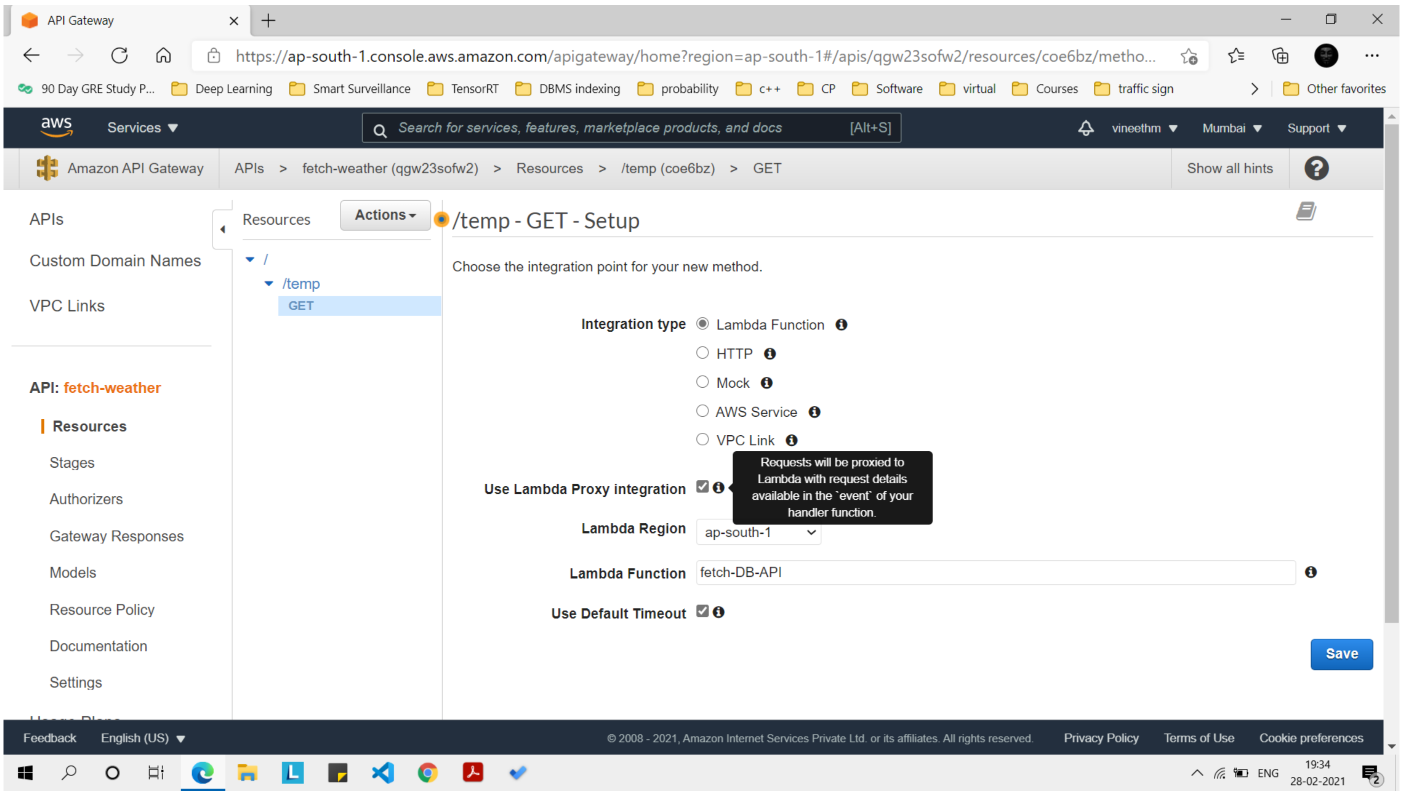
You can add path parameters using brackets. For example, the resource path {username} represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS ☐

* Required

https://ap-south-1.console.aws.amazon.com/apigateway/home?region=ap-south-1#/apis/qgw23sofw2/resources/22cfct3xwg/cre...
Amazon API Gateway
APIs > fetch-weather (qgw23sofw2) > Resources > / (22cfct3xwg) > Create
Cancel
Create Resource

Task-8: Create a GET method and select lambda function as integration type and select the “Use Lambda Proxy integration” option and then provide the Lambda function name, created above.



/temp - GET - Setup

Choose the integration point for your new method.

Integration type [✓ Lambda Function] HTTP Mock AWS Service VPC Link

Use Lambda Proxy integration ✓

Requests will be proxied to Lambda with request details available in the `event` of your handler function.

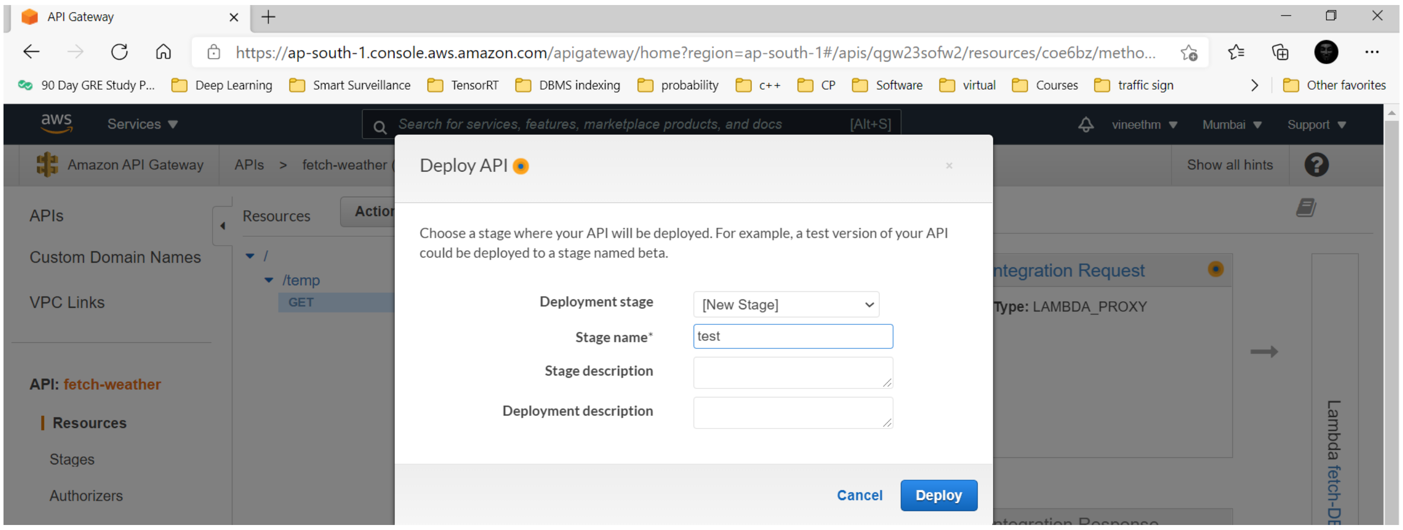
Lambda Region ap-south-1

Lambda Function fetch-DB-API

Use Default Timeout ✓

https://ap-south-1.console.aws.amazon.com/apigateway/home?region=ap-south-1#/apis/qgw23sofw2/resources/coe6bz/metho...
APIs > fetch-weather (qgw23sofw2) > Resources > /temp (coe6bz) > GET
/
/temp
GET
Save

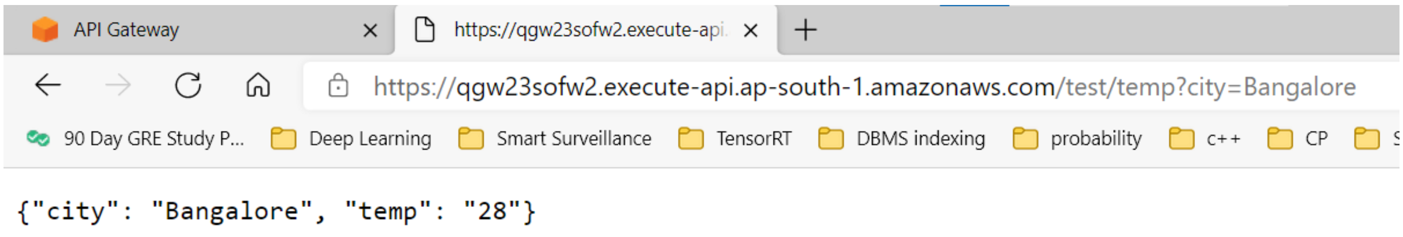
Task-9: Deploy the API.



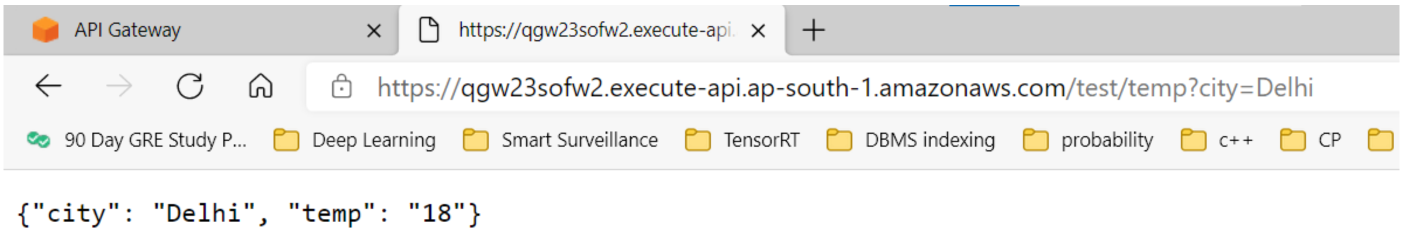
Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage: [New Stage]
Stage name*: test
Stage description:
Deployment description:

Task-10: Test the working of the deployed API by using the invoke url of the API.

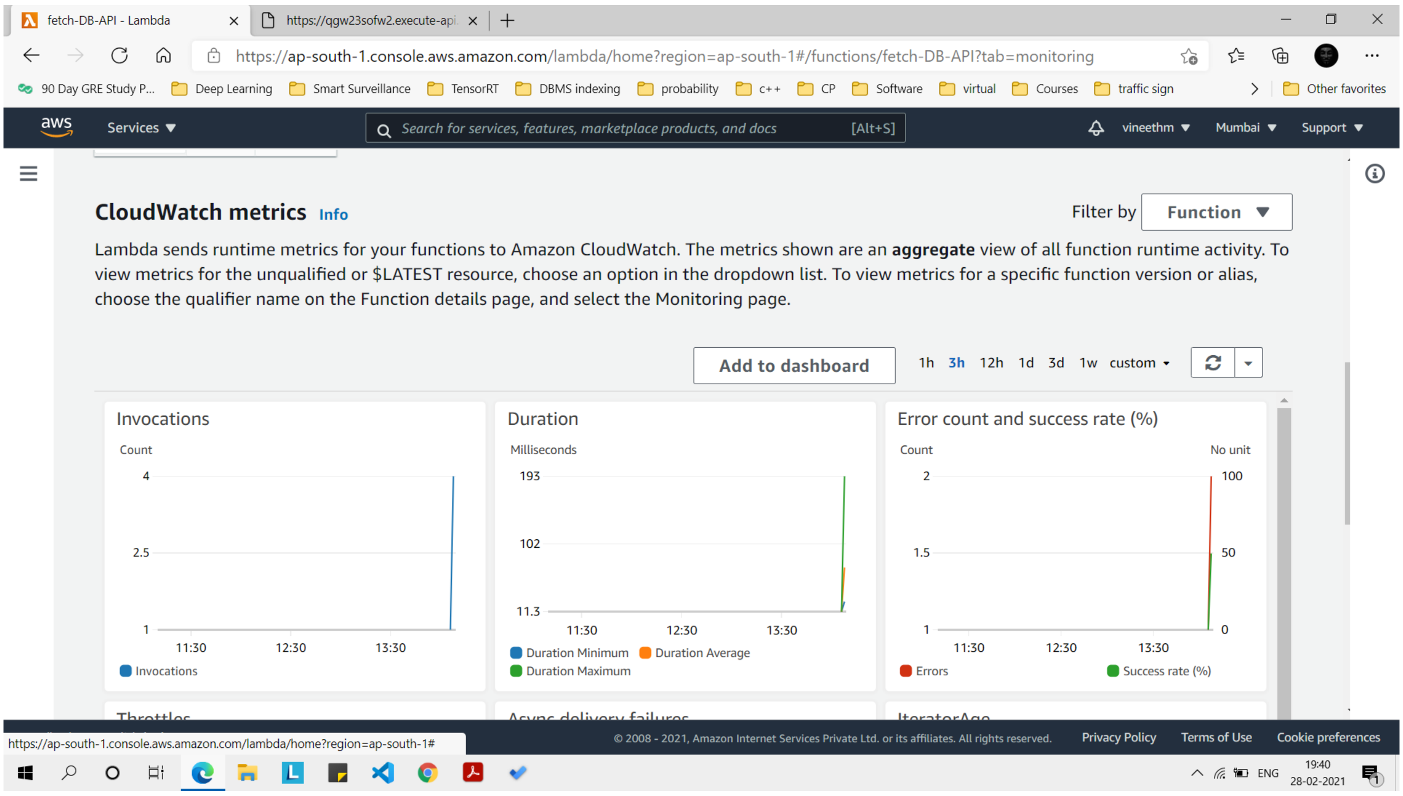

API Gateway
https://qgw23sofw2.execute-api.ap-south-1.amazonaws.com/test/temp?city=Bangalore

```
{"city": "Bangalore", "temp": "28"}
```


API Gateway
https://qgw23sofw2.execute-api.ap-south-1.amazonaws.com/test/temp?city=Delhi

```
{"city": "Delhi", "temp": "18"}
```

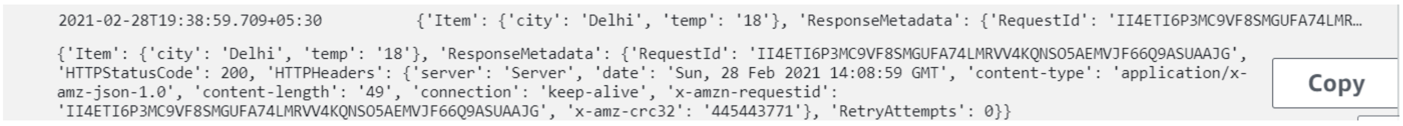
CloudWatch Logs:


fetch-DB-API - Lambda
https://qgw23sofw2.execute-api
https://ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/fetch-DB-API?tab=monitoring
Services
Search for services, features, marketplace products, and docs [Alt+S]
vineethm
Mumbai
Support

CloudWatch metrics Info

Filter by Function

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an **aggregate** view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose an option in the dropdown list. To view metrics for a specific function version or alias, choose the qualifier name on the Function details page, and select the Monitoring page.



```
2021-02-28T19:38:59.709+05:30          {'Item': {'city': 'Delhi', 'temp': '18'}, 'ResponseMetadata': {'RequestId': 'II4ETI6P3MC9VF8SMGUFA74LMR…

{'Item': {'city': 'Delhi', 'temp': '18'}, 'ResponseMetadata': {'RequestId': 'II4ETI6P3MC9VF8SMGUFA74LMRVV4KQNSO5AEMVJF66Q9ASUAAJG',
'HTTPStatusCode': 200, 'HTTPHeaders': {'server': 'Server', 'date': 'Sun, 28 Feb 2021 14:08:59 GMT', 'content-type': 'application/x-
amz-json-1.0', 'content-length': '49', 'connection': 'keep-alive', 'x-amzn-requestid':
'II4ETI6P3MC9VF8SMGUFA74LMRVV4KQNSO5AEMVJF66Q9ASUAAJG', 'x-amz-crc32': '445443771'}, 'RetryAttempts': 0}}
```