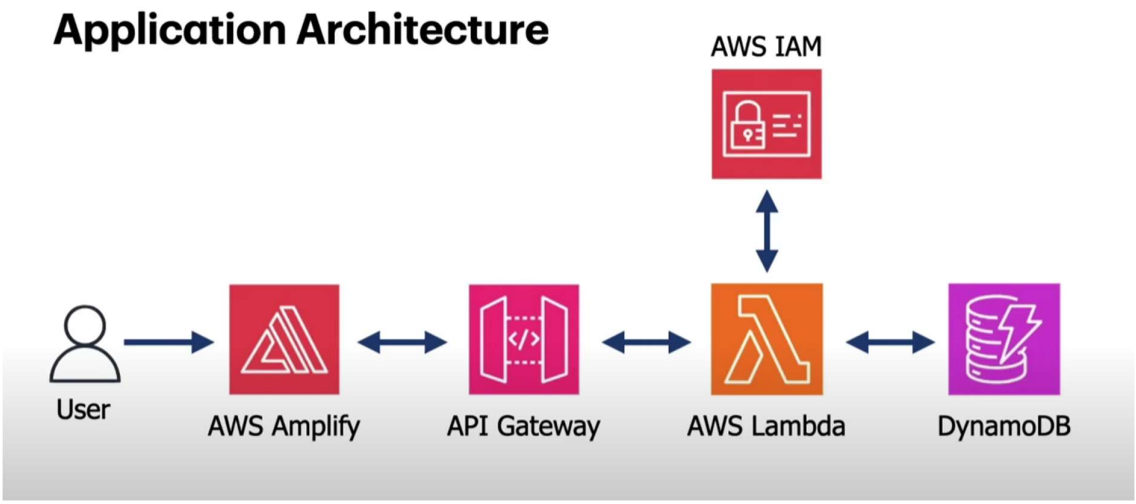
M. Sai Saranya

22BAI1471

Lab-05

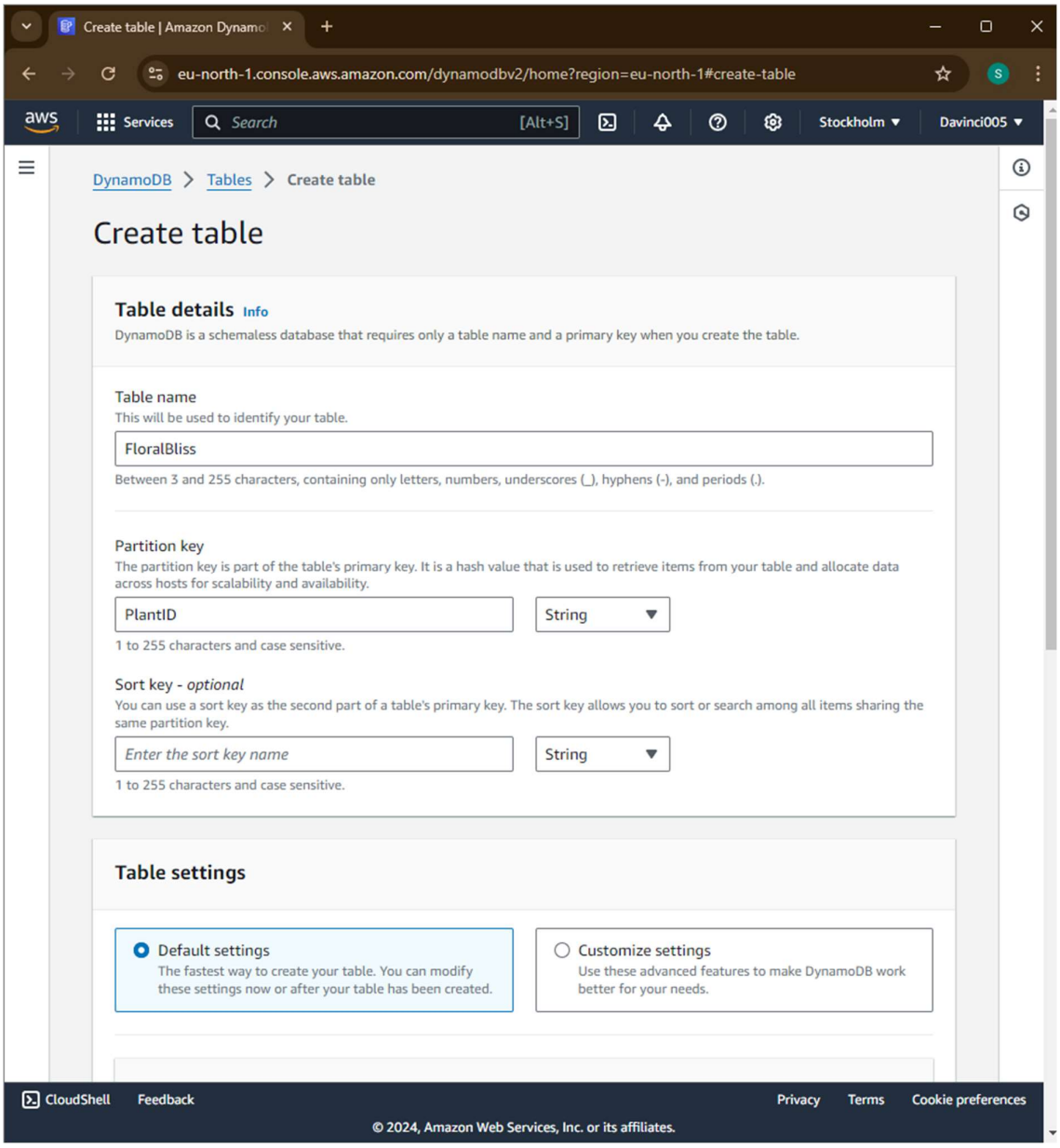
Creating and deploying a serverless application using DynamoDB

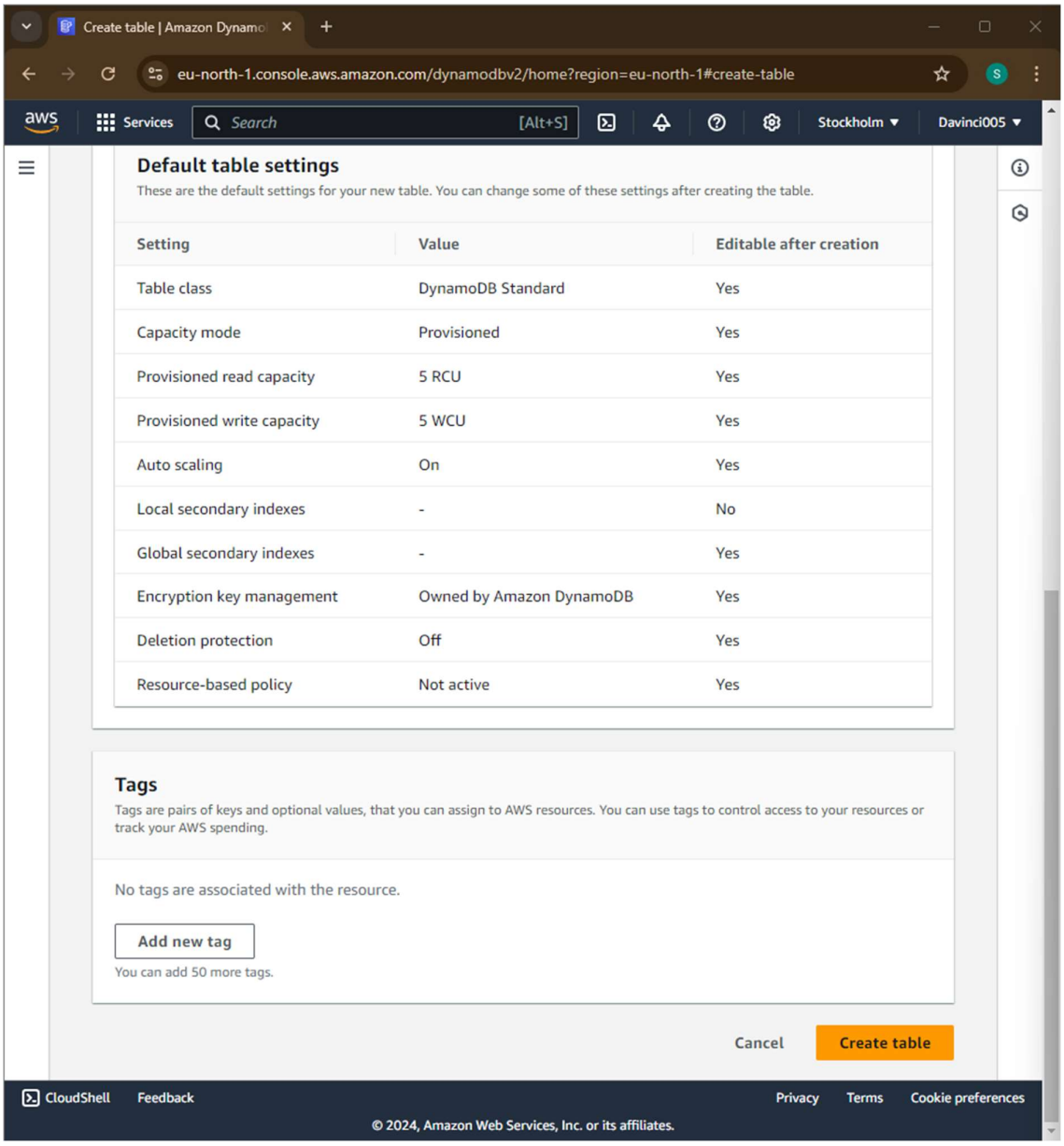


1. **DynamoDB (Database)**:
    - **Purpose**: Stores plant data (e.g., PlantID, Name, Price, etc.).
    - **Link**: The table is accessed by the Lambda function for reading (Scan) and writing (PutItem) operations.
2. **Lambda (Serverless Backend)**:
    - **Purpose**: Contains the logic to interact with DynamoDB (add plants, fetch plant details).
    - **Link**: Lambda reads from and writes to the DynamoDB table. It is invoked by API Gateway to perform these operations when triggered by HTTP requests.
3. **API Gateway (REST API)**:
    - **Purpose**: Provides a REST API to access Lambda functions through HTTP methods (POST, GET).
    - **Link**: Acts as an interface between the front-end (HTML/JavaScript) and the Lambda function, sending user requests to Lambda.
4. **IAM (Access Control)**:
    - **Purpose**: Manages permissions and access control for AWS resources.
    - **Link**: IAM roles and policies grant the Lambda function permissions to read and write data in DynamoDB.
5. **AWS Amplify (Web Hosting)**:
    - **Purpose**: Hosts and deploys the front-end (HTML/JavaScript) of the botanist shop app.
    - **Link**: Amplify serves the front-end which interacts with API Gateway to make requests that trigger the Lambda function.

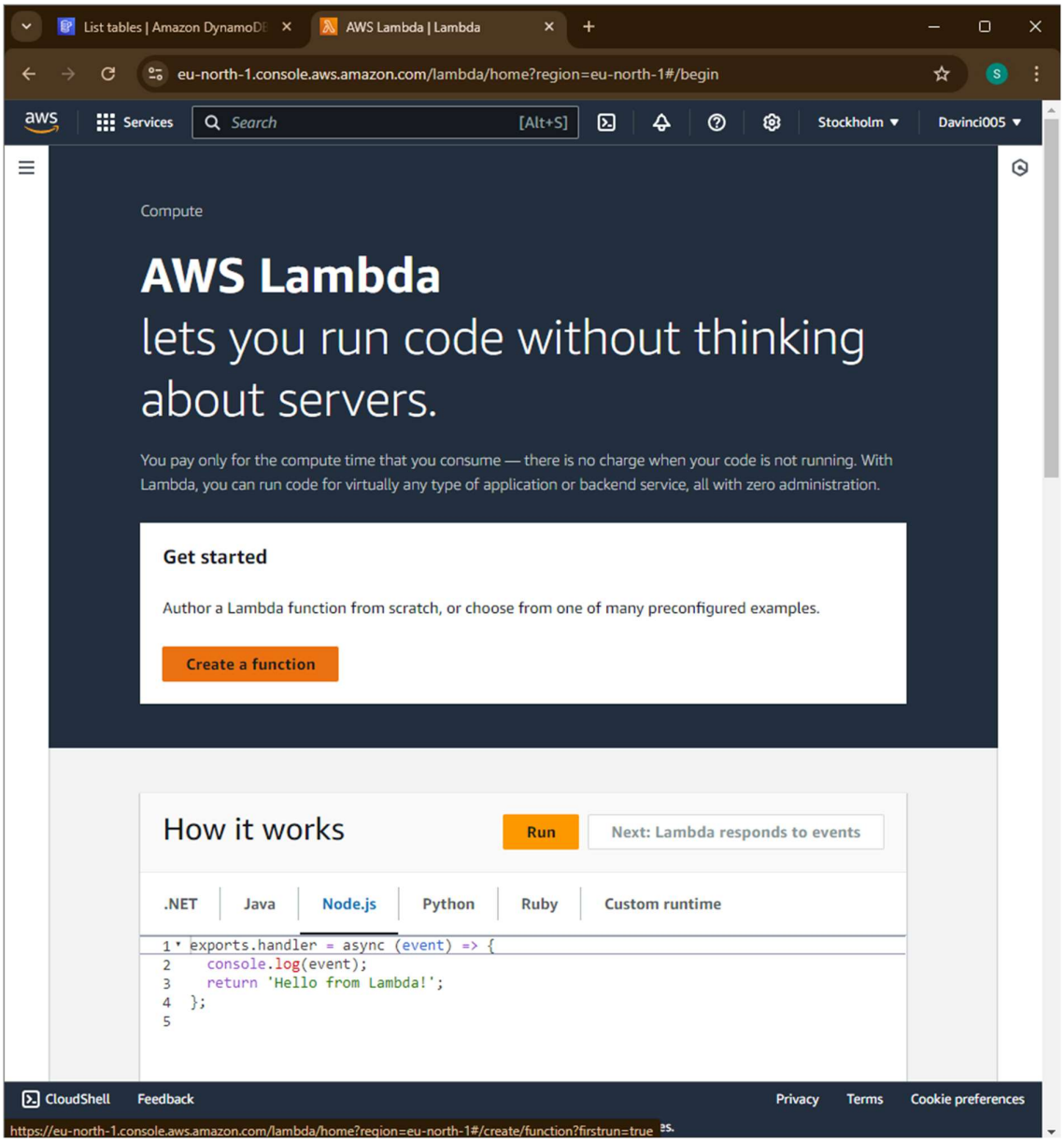
These services work together to create a full-stack application where users can interact with the front-end (Amplify), make requests via API Gateway, which trigger Lambda functions to interact with DynamoDB, all controlled by IAM permissions.

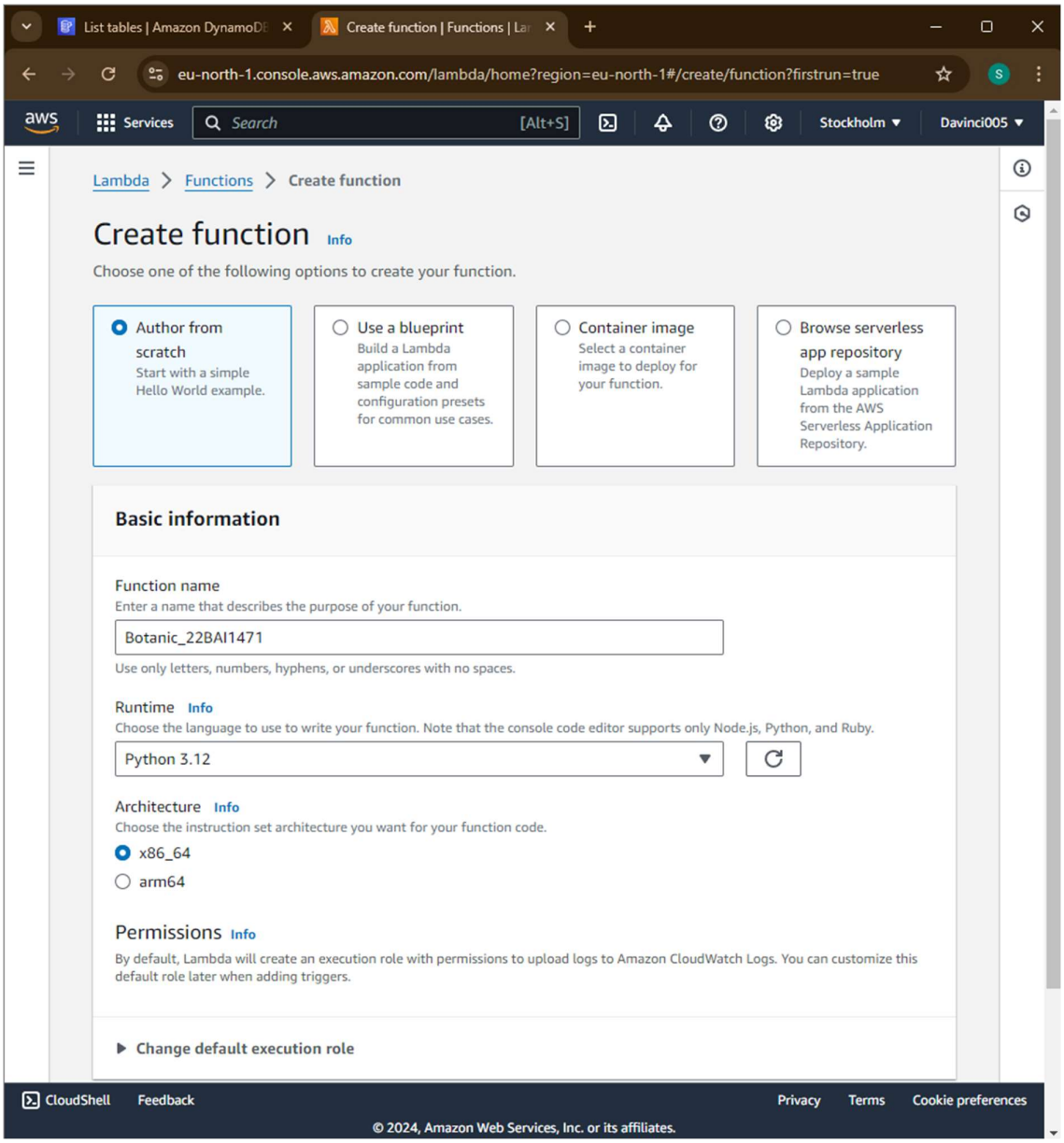
1) Create a table in DynamoDB and give a partition key that serves as the primary key for accessing the table items.

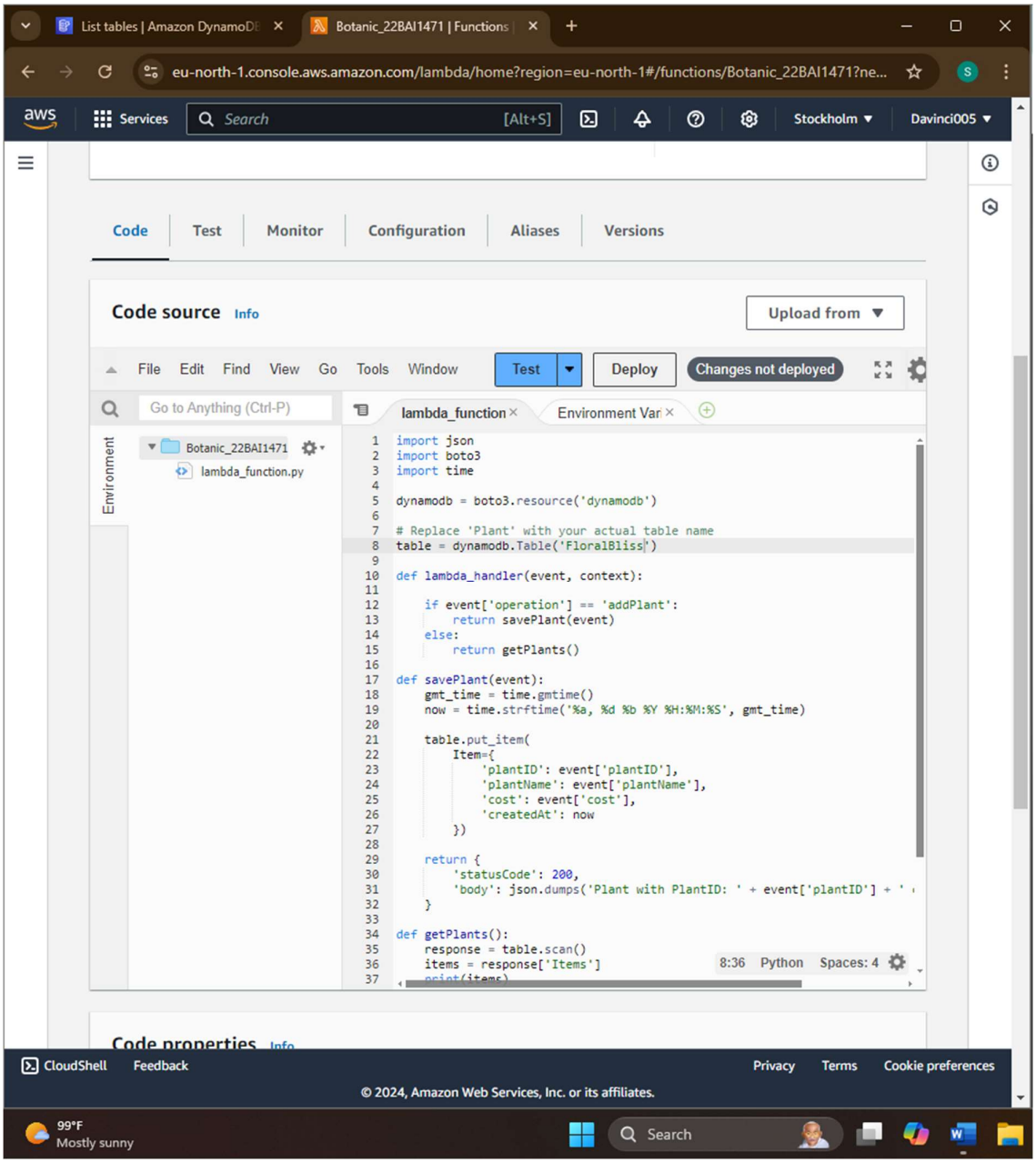




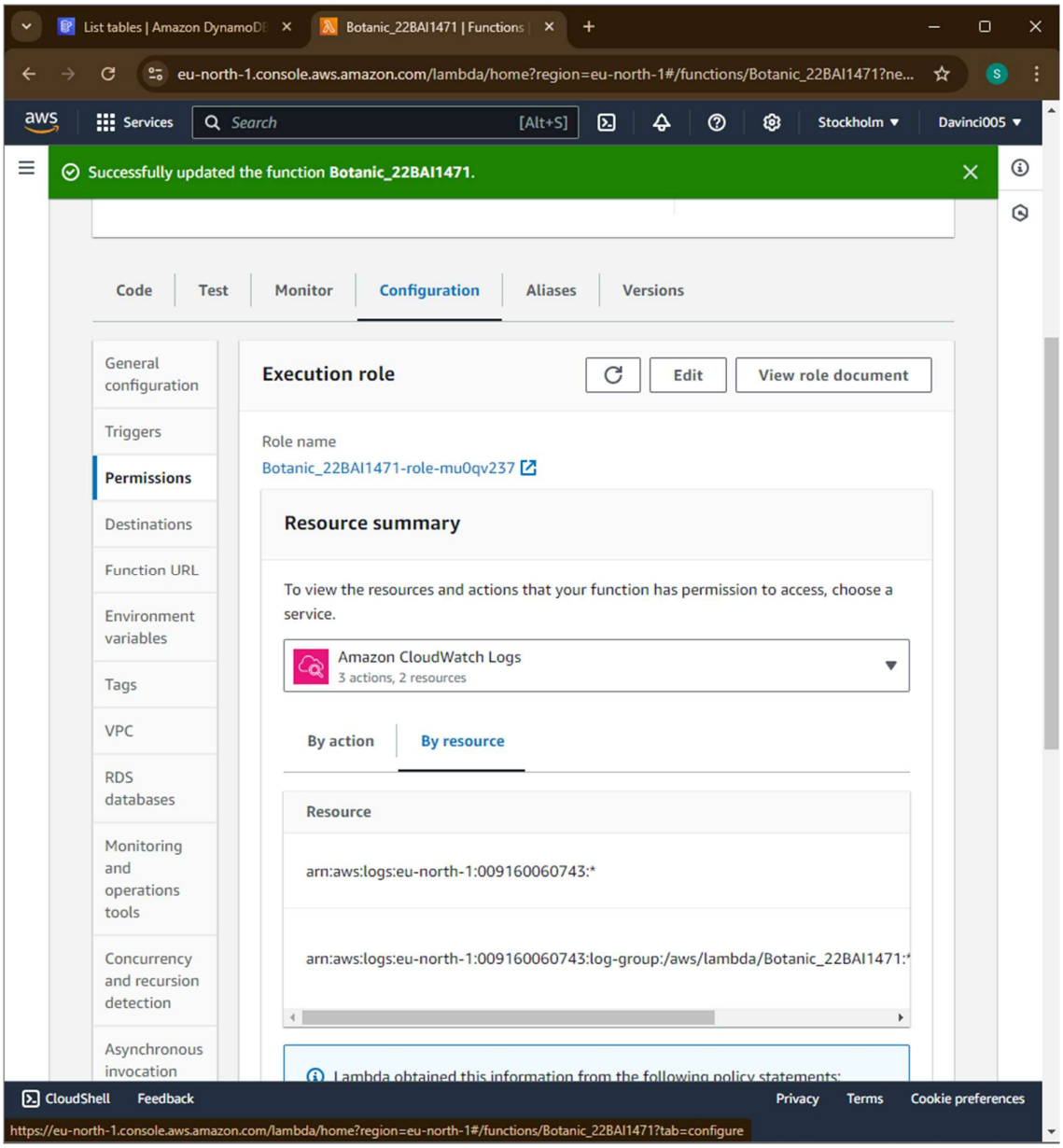
2) Create a lambda function by choosing the desired language and writing functionalities of how the app should work.

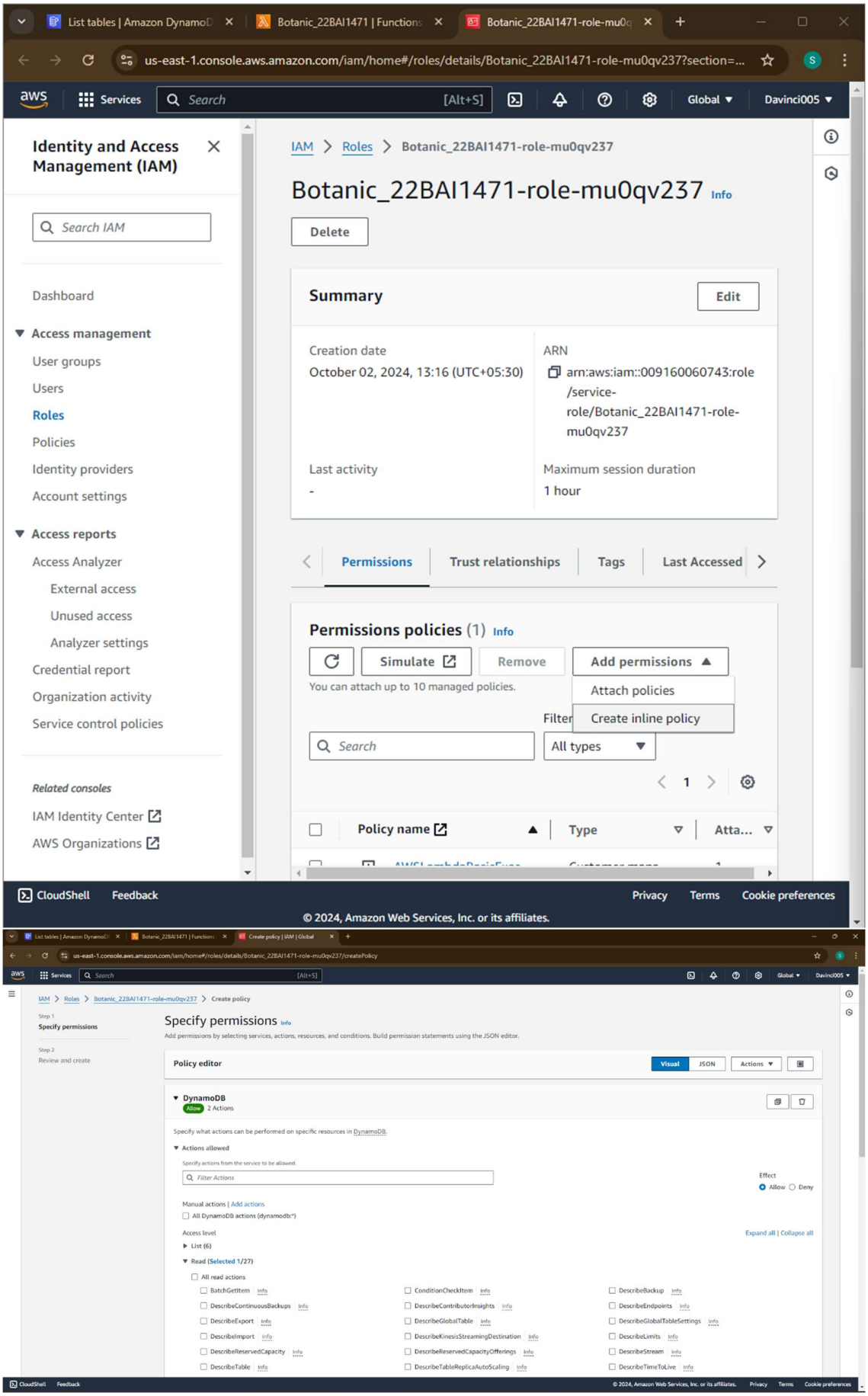

List tables | Amazon DynamoDB
AWS Lambda | Lambda
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/begin
aws
Services
Search
[Alt+S]
Stockholm
Davinci005
Compute
AWS Lambda
lets you run code without thinking about servers.
You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.
Get started
Author a Lambda function from scratch, or choose from one of many preconfigured examples.
Create a function
How it works
Run
Next: Lambda responds to events
.NET
Java
Node.js
Python
Ruby
Custom runtime
exports.handler = async (event) => {
console.log(event);
return 'Hello from Lambda!';
};
CloudShell
Feedback
Privacy
Terms
Cookie preferences
https://eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function?firstrun=true


List tables | Amazon DynamoDB
Create function | Functions | Lambda
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function?firstrun=true
aws
Services
Search
[Alt+S]
Stockholm
Davinci005
Lambda > Functions > 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.
Botanic_22BAI1471
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.
Python 3.12
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
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2024, Amazon Web Services, Inc. or its affiliates.

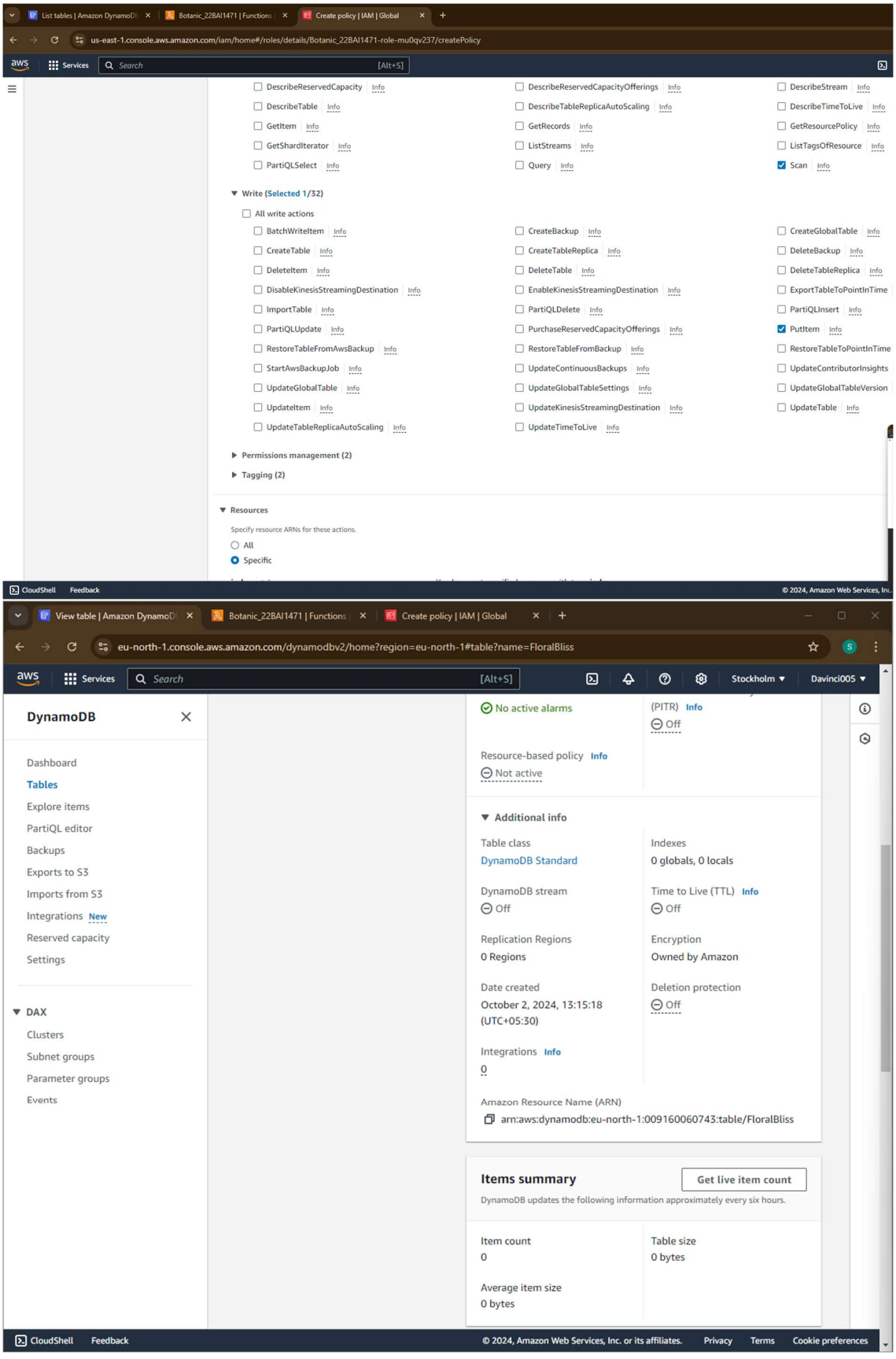


3) But this function is not ready to use, since we didn't give access to it like for iam user for execution role
4) Go to the permissions and create an inline policy where we can select the service, we can use like DynamoDB and also the operations that we need to perform mainly two-> to read (Scan) and to write (PutItem)

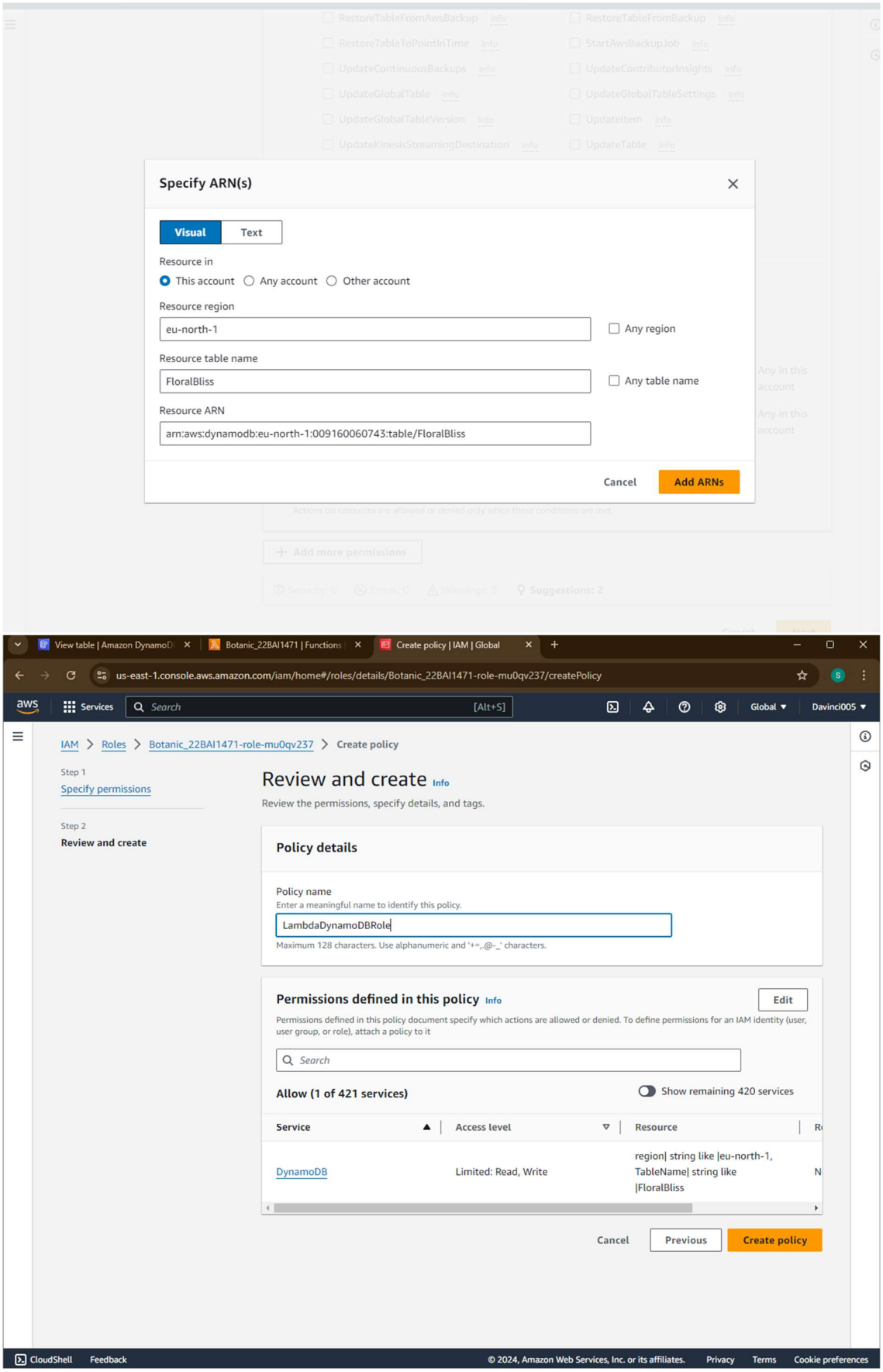

List tables | Amazon DynamoDB
Botanic_22BAI1471 | Functions
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/functions/Botanic_22BAI1471?ne...
aws
Services
Search
[Alt+S]
Stockholm
Davinci005
Successfully updated the function Botanic_22BAI1471.
Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools
Concurrency and recursion detection
Asynchronous invocation
Execution role
Edit
View role document
Role name
Botanic_22BAI1471-role-mu0qv237
Resource summary
To view the resources and actions that your function has permission to access, choose a service.
Amazon CloudWatch Logs
3 actions, 2 resources
By action
By resource
Resource
arn:aws:logs:eu-north-1:009160060743:*
arn:aws:logs:eu-north-1:009160060743:log-group:/aws/lambda/Botanic_22BAI1471:*
Lambda obtained this information from the following policy statements:
CloudShell
Feedback
Privacy
Terms
Cookie preferences
https://eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/functions/Botanic_22BAI1471?tab=configure

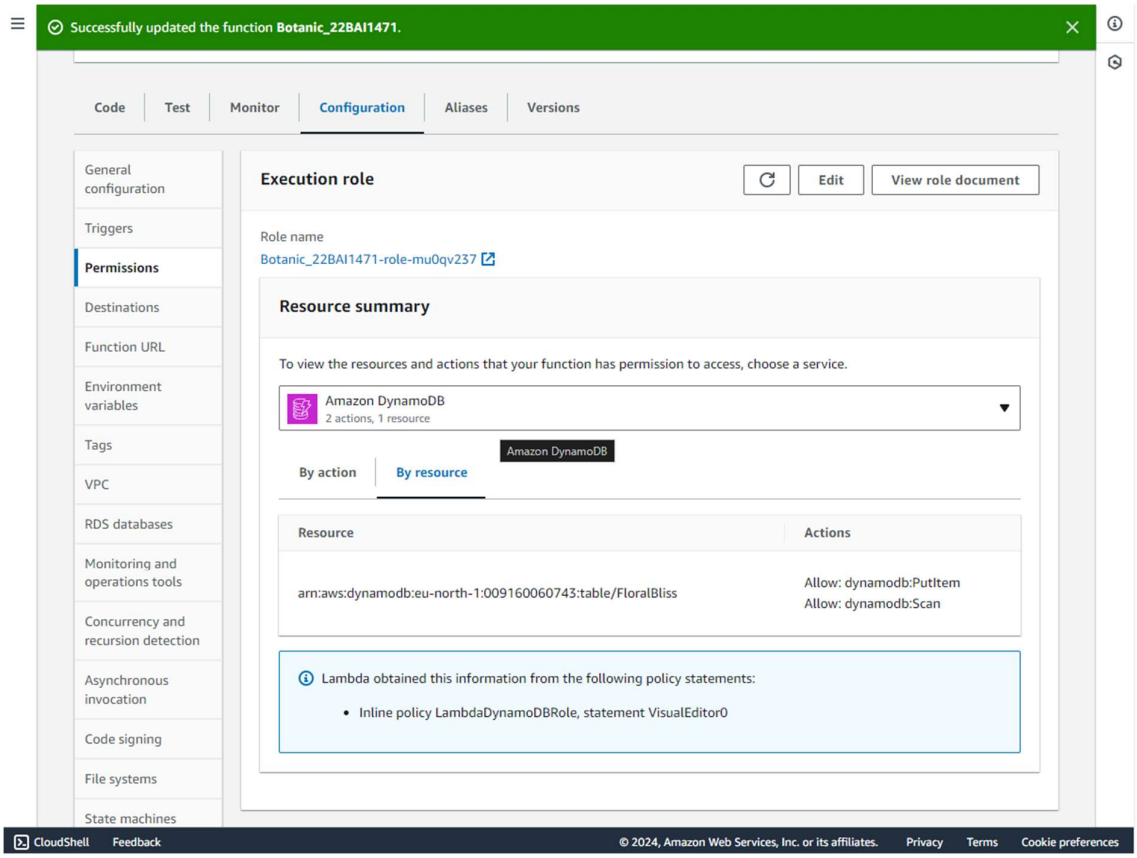

List tables | Amazon DynamoD
Botanic_22BAI1471 | Function
Botanic_22BAI1471-role-mu0q
us-east-1.console.aws.amazon.com/iam/home#/roles/details/Botanic_22BAI1471-role-mu0qv237?section=...
aws
Services
Search
[Alt+S]
Global
Davinci005
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies
Related consoles
IAM Identity Center
AWS Organizations
IAM > Roles > Botanic_22BAI1471-role-mu0qv237
Botanic_22BAI1471-role-mu0qv237 Info
Delete
Summary
Edit
Creation date
October 02, 2024, 13:16 (UTC+05:30)
ARN
arn:aws:iam::009160060743:role/service-role/Botanic_22BAI1471-role-mu0qv237
Last activity
-
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Permissions policies (1) Info
Simulate
Remove
Add permissions
You can attach up to 10 managed policies.
Attach policies
Create inline policy
Filter
Search
All types
1
Policy name
Type
Atta...
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2024, Amazon Web Services, Inc. or its affiliates.

us-east-1.console.aws.amazon.com/iam/home#/roles/details/Botanic_22BAI1471-role-mu0qv237/createPolicy
IAM > Roles > Botanic_22BAI1471-role-mu0qv237 > Create policy
Step 1
Specify permissions
Step 2
Review and create
Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.
Policy editor
Visual
JSON
Actions
DynamoDB
Allow 2 Actions
Specify what actions can be performed on specific resources in DynamoDB.
Actions allowed
Specify actions from the service to be allowed.
Filter Actions
Effect
Allow
Deny
Manual actions | Add actions
All DynamoDB actions (dynamodb:*)
Access level
Expand all | Collapse all
List (6)
Read (Selected 1/27)
All read actions
BatchGetItem
ConditionCheckItem
DescribeBackup
DescribeContinuousBackups
DescribeContributorInsights
DescribeEndpoints
DescribeExport
DescribeGlobalTable
DescribeGlobalTableSettings
DescribeImport
DescribeKinesisStreamingDestination
DescribeLimits
DescribeReservedCapacity
DescribeReservedCapacityOfferings
DescribeStream
DescribeTable
DescribeTableReplicaAutoScaling
DescribeTimeToLive
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

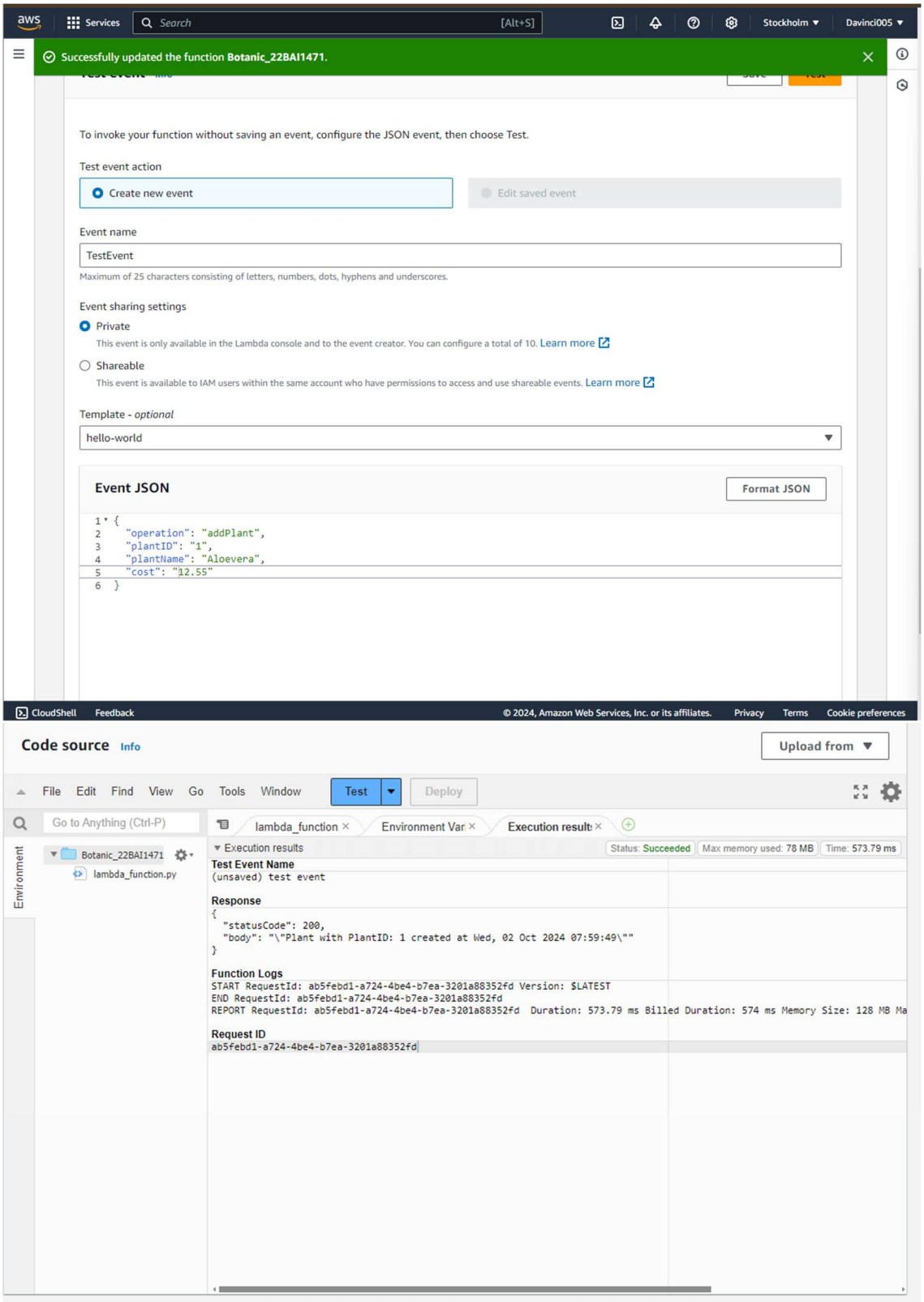


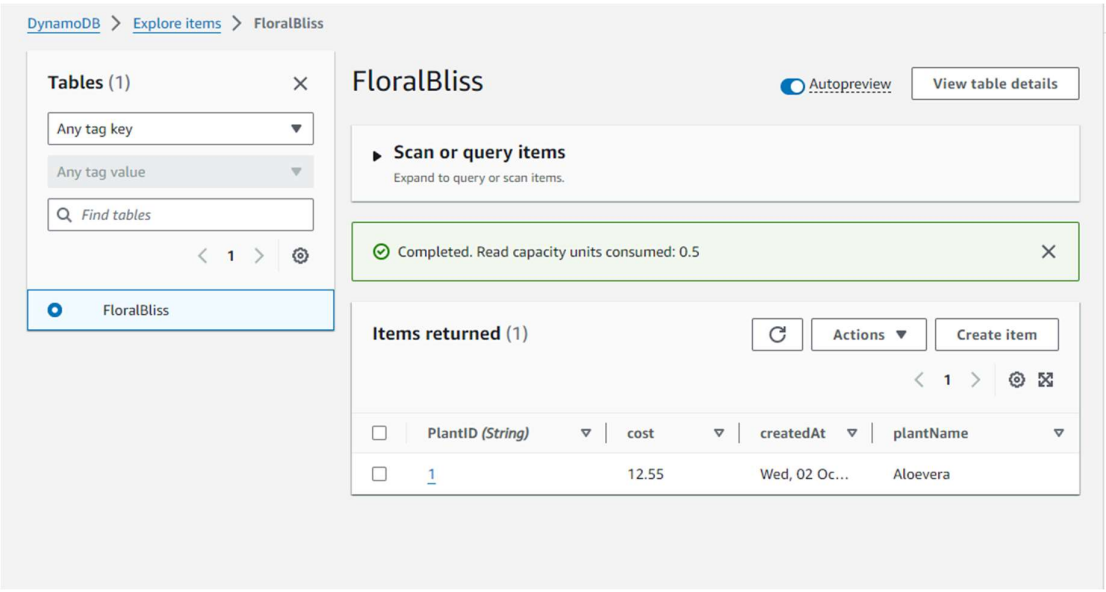
5) Copy the ARN and paste it into the option named add resources in permissions of the user role that we need to select resource region, table name, and ARN


Specify ARN(s)
Visual
Text
Resource in
This account
Any account
Other account
Resource region
eu-north-1
Any region
Resource table name
FloralBliss
Any table name
Resource ARN
arn:aws:dynamodb:eu-north-1:009160060743:table/FloralBliss
Cancel
Add ARNs
Add more permissions
View table | Amazon DynamoD
Botanic_22BAI1471 | Functions
Create policy | IAM | Global
us-east-1.console.aws.amazon.com/iam/home#/roles/details/Botanic_22BAI1471-role-mu0qv237/createPolicy
Services
Search
[Alt+S]
Global
Davinci005
IAM
Roles
Botanic_22BAI1471-role-mu0qv237
Create policy
Step 1
Specify permissions
Step 2
Review and create
Review and create
Info
Review the permissions, specify details, and tags.
Policy details
Policy name
Enter a meaningful name to identify this policy.
LambdaDynamoDBRole
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.
Permissions defined in this policy
Info
Edit
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it
Search
Allow (1 of 421 services)
Show remaining 420 services
Service
Access level
Resource
DynamoDB
Limited: Read, Write
region| string like |eu-north-1, TableName| string like |FloralBliss
Cancel
Previous
Create policy
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

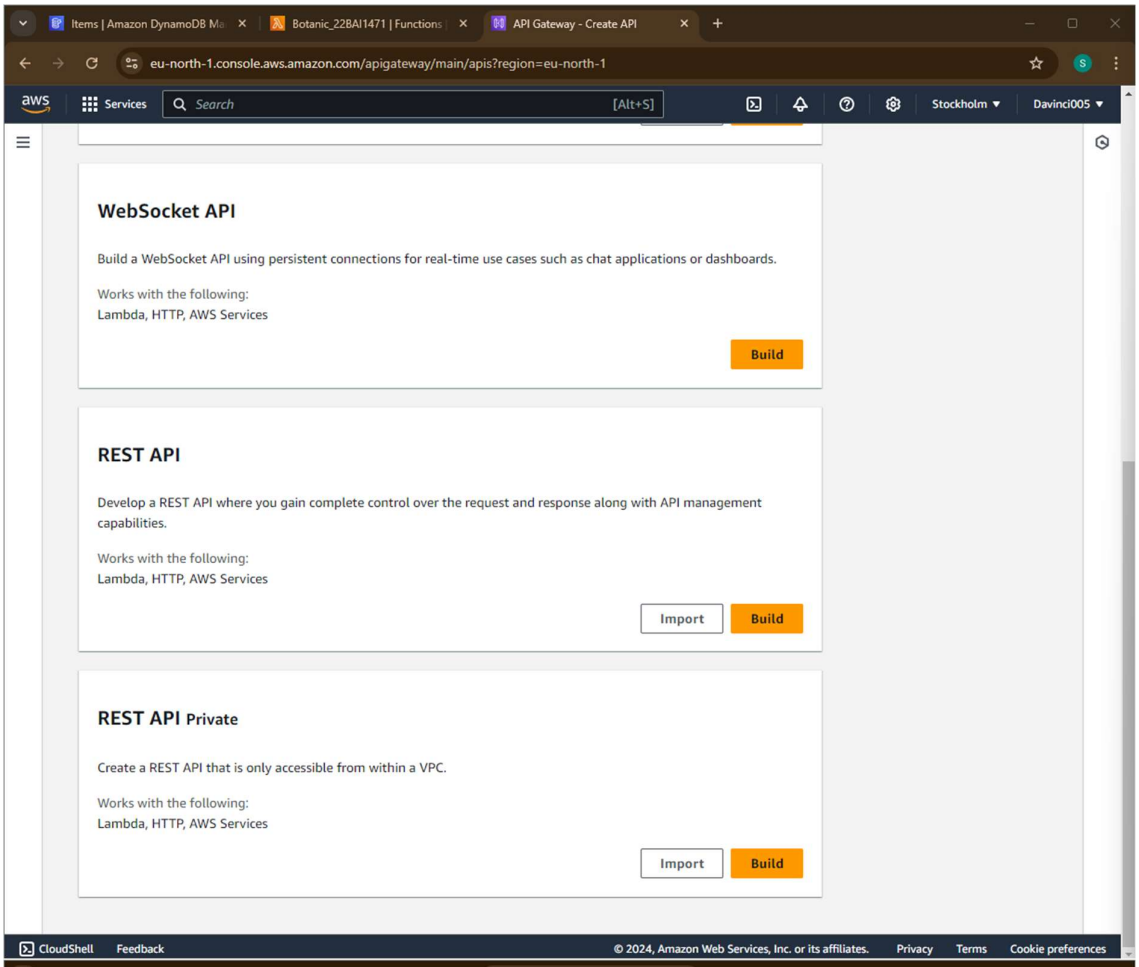


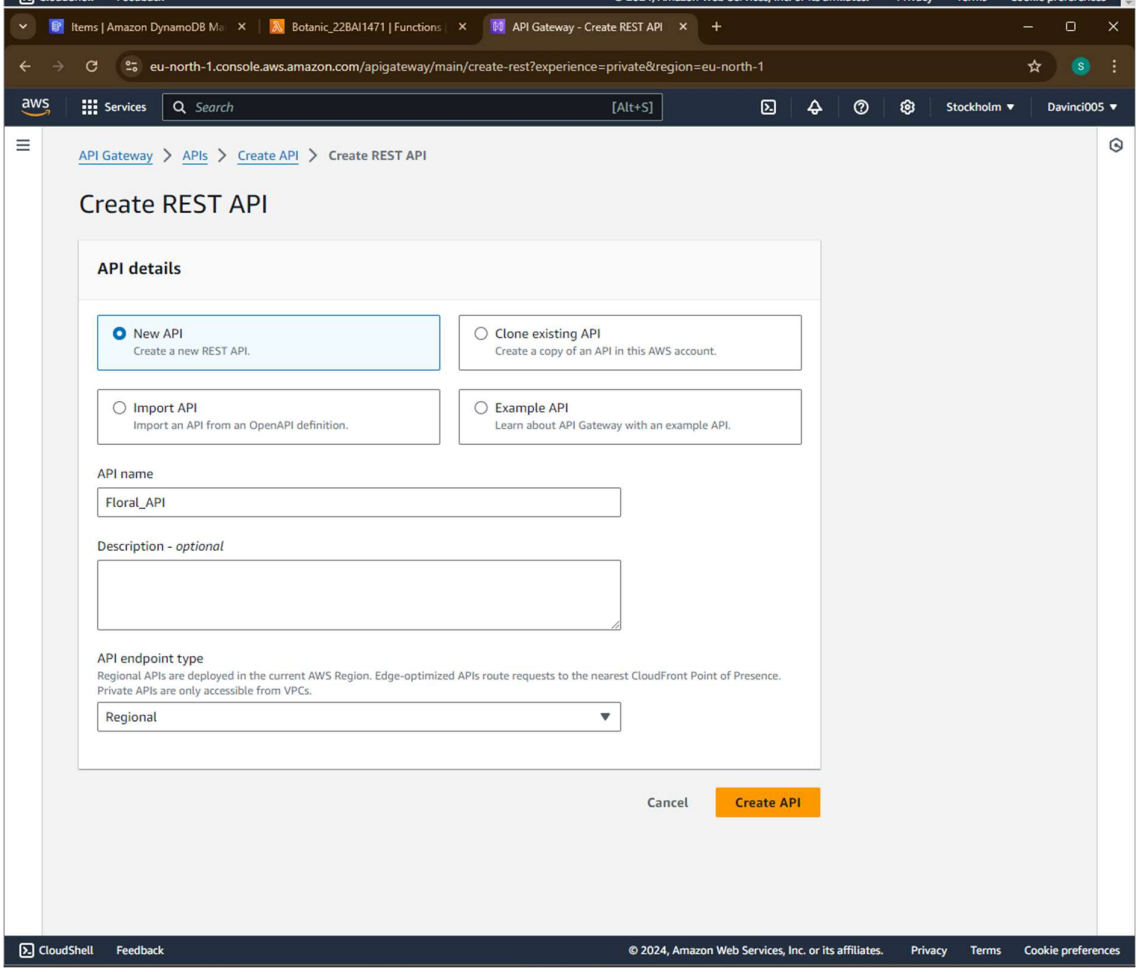
6) Now we can test the function using a test event by providing a Json format of data which can be inserted into the table


aws
Services
Search
[Alt+S]
Stockholm
Davinci005
Successfully updated the function Botanic_22BAI1471.
To invoke your function without saving an event, configure the JSON event, then choose Test.
Test event action
Create new event
Edit saved event
Event name
TestEvent
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
{
"operation": "addPlant",
"plantID": "1",
"plantName": "Aloevera",
"cost": "12.55"
}
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Code source
Info
Upload from
File
Edit
Find
View
Go
Tools
Window
Test
Deploy
Go to Anything (Ctrl-P)
Environment
Botanic_22BAI1471
lambda_function.py
lambda_function
Environment Var
Execution result
Execution results
Status: Succeeded
Max memory used: 78 MB
Time: 573.79 ms
Test Event Name
(unsaved) test event
Response
{
"statusCode": 200,
"body": "\"Plant with PlantID: 1 created at Wed, 02 Oct 2024 07:59:49\""
}
Function Logs
START RequestId: ab5febd1-a724-4be4-b7ea-3201a88352fd Version: $LATEST
END RequestId: ab5febd1-a724-4be4-b7ea-3201a88352fd
REPORT RequestId: ab5febd1-a724-4be4-b7ea-3201a88352fd Duration: 573.79 ms Billed Duration: 574 ms Memory Size: 128 MB Ma
Request ID
ab5febd1-a724-4be4-b7ea-3201a88352fd

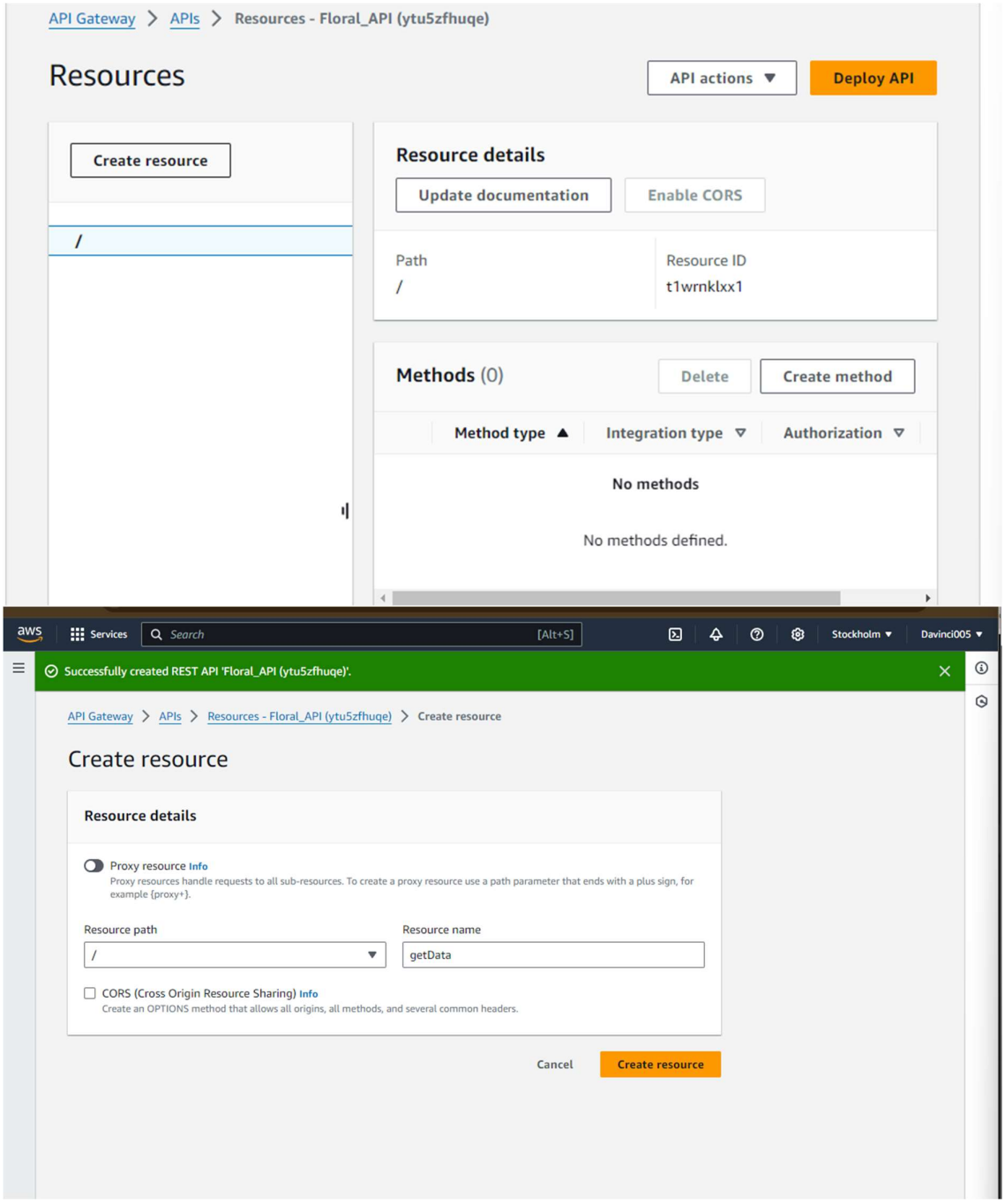


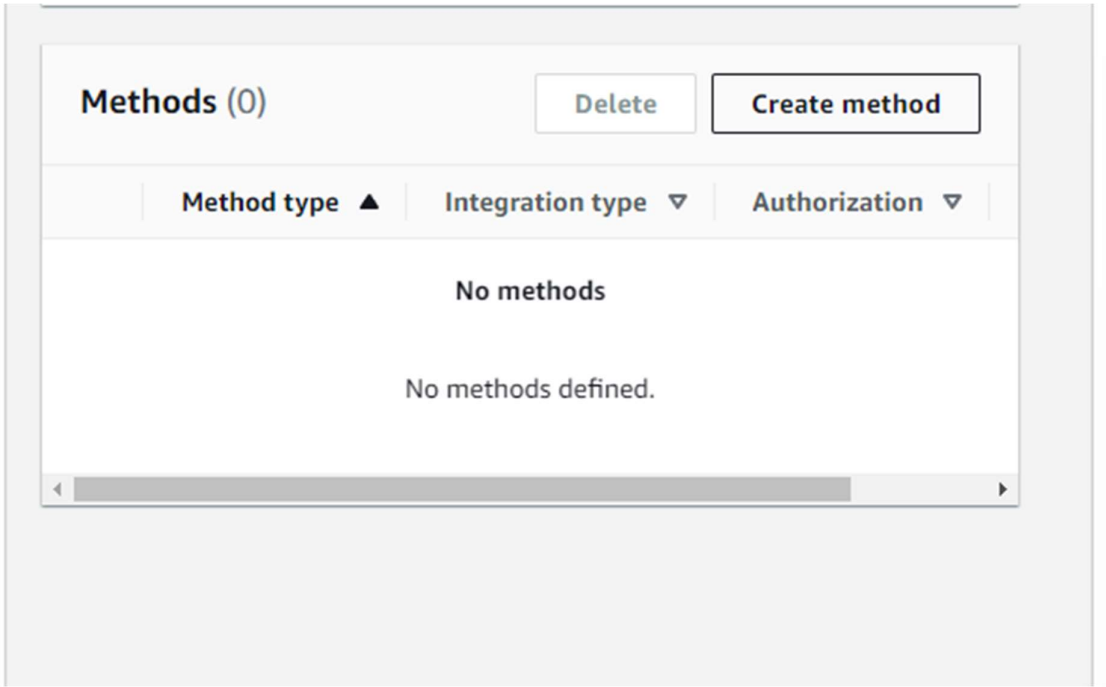
7) Now we need to create a rest API to access the table via function

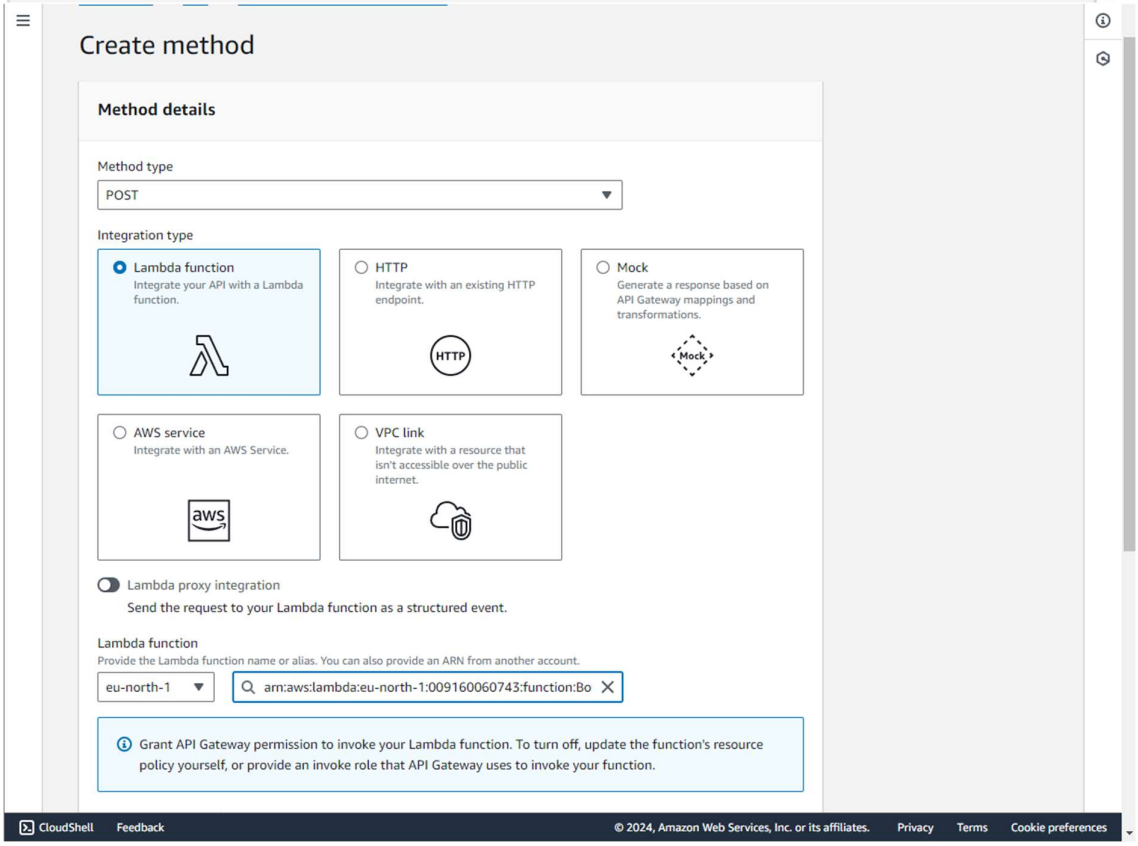

Items | Amazon DynamoDB Ma
Botanic_22BAI1471 | Functions
API Gateway - Create API
eu-north-1.console.aws.amazon.com/apigateway/main/apis?region=eu-north-1
aws
Services
Search
[Alt+S]
Stockholm
Davinci005
WebSocket API
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.
Works with the following:
Lambda, HTTP, AWS Services
Build
REST API
Develop a REST API where you gain complete control over the request and response along with API management capabilities.
Works with the following:
Lambda, HTTP, AWS Services
Import
Build
REST API Private
Create a REST API that is only accessible from within a VPC.
Works with the following:
Lambda, HTTP, AWS Services
Import
Build
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Items | Amazon DynamoDB Ma
Botanic_22BAI1471 | Functions
API Gateway - Create REST API
eu-north-1.console.aws.amazon.com/apigateway/main/create-rest?experience=private&region=eu-north-1
aws
Services
Search
[Alt+S]
Stockholm
Davinci005
API Gateway
APIs
Create API
Create REST API
Create REST API
API details
New API
Create a new REST API.
Clone existing API
Create a copy of an API in this AWS account.
Import API
Import an API from an OpenAPI definition.
Example API
Learn about API Gateway with an example API.
API name
Floral_API
Description - optional
API endpoint type
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.
Regional
Cancel
Create API
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

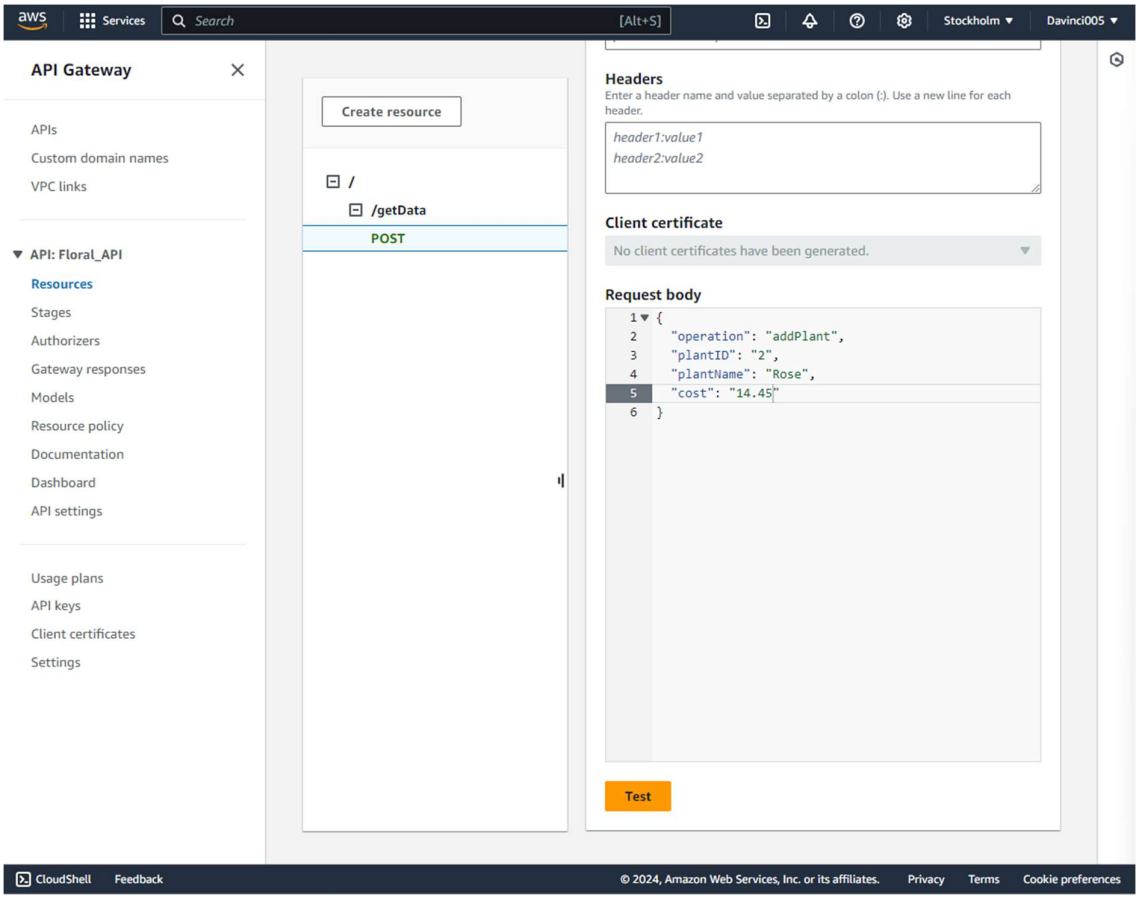
8) Create a resource named getData and add post and get methods to the api

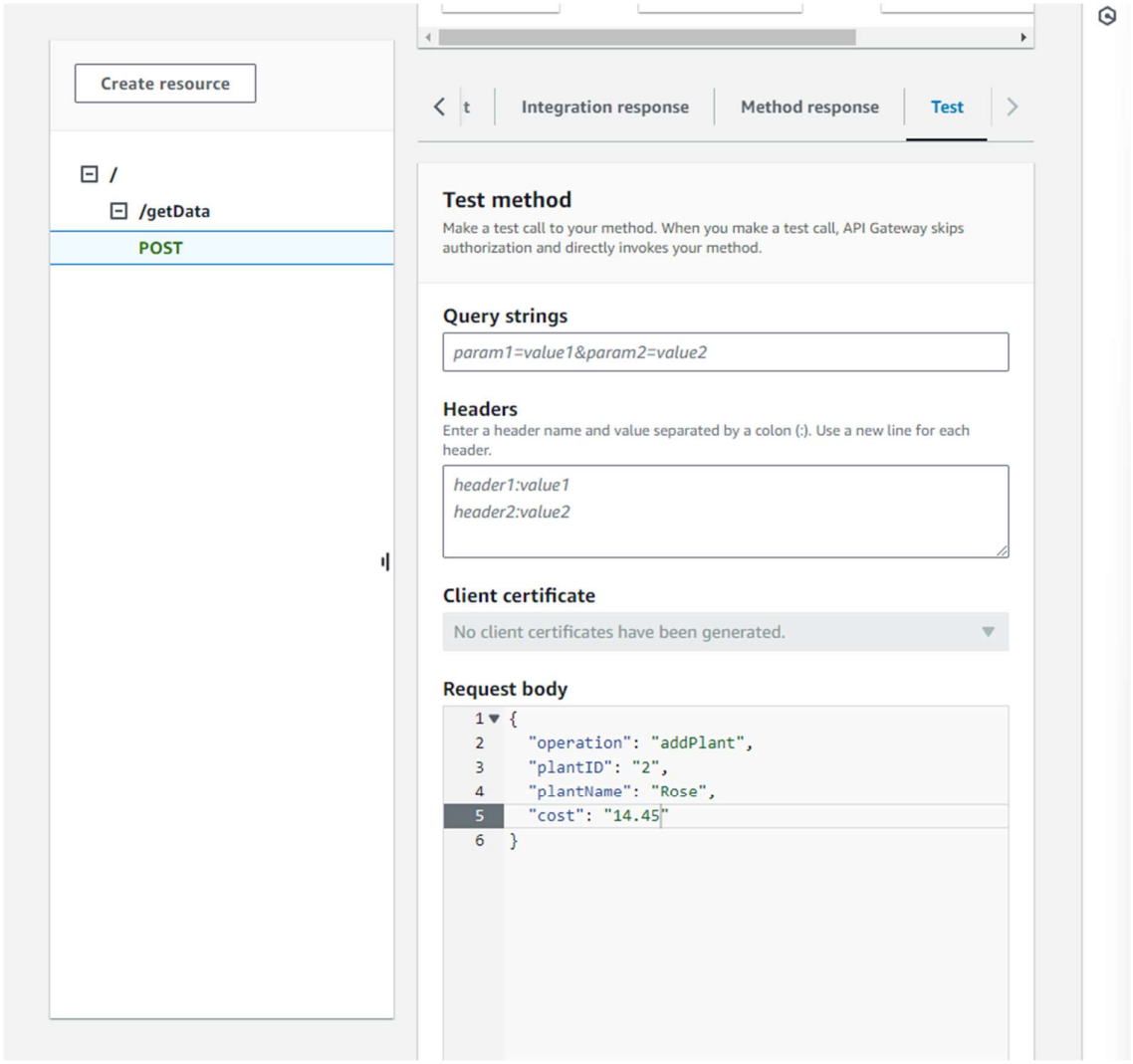






9) We can test whether the api is working or not using Request body like that of test event we did earlier


aws
Services
Search
[Alt+S]
Stockholm
Davinci005
API Gateway
APIs
Custom domain names
VPC links
API: Floral_API
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
API keys
Client certificates
Settings
Create resource
/
/getData
POST
Headers
Enter a header name and value separated by a colon (:). Use a new line for each header.
header1:value1
header2:value2
Client certificate
No client certificates have been generated.
Request body
{
"operation": "addPlant",
"plantID": "2",
"plantName": "Rose",
"cost": "14.45"
}
Test
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Create resource
/
/getData
POST
t
Integration response
Method response
Test
Test method
Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.
Query strings
param1=value1&param2=value2
Headers
Enter a header name and value separated by a colon (:). Use a new line for each header.
header1:value1
header2:value2
Client certificate
No client certificates have been generated.
Request body
{
"operation": "addPlant",
"plantID": "2",
"plantName": "Rose",
"cost": "14.45"
}

/getData - POST method test results

Request

/getData

Latency ms

543

Status

200

Response body

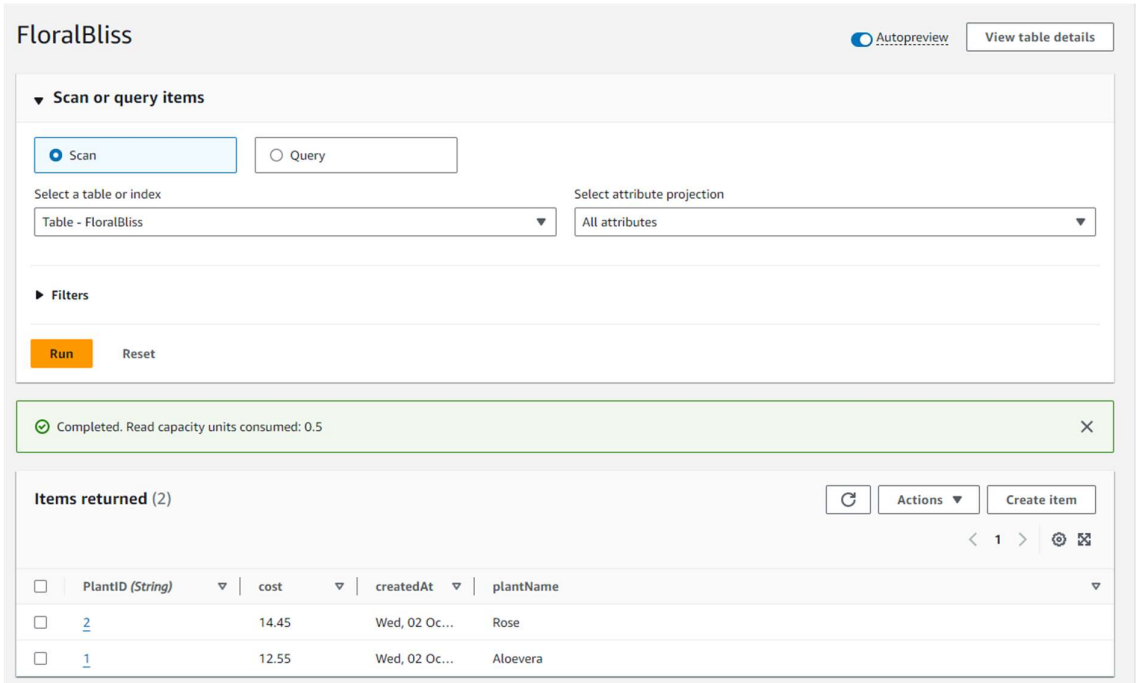

```
{"statusCode": 200, "body": "\"Plant with
PlantID: 2 created at Wed, 02 Oct 2024
08:05:26\""}
```

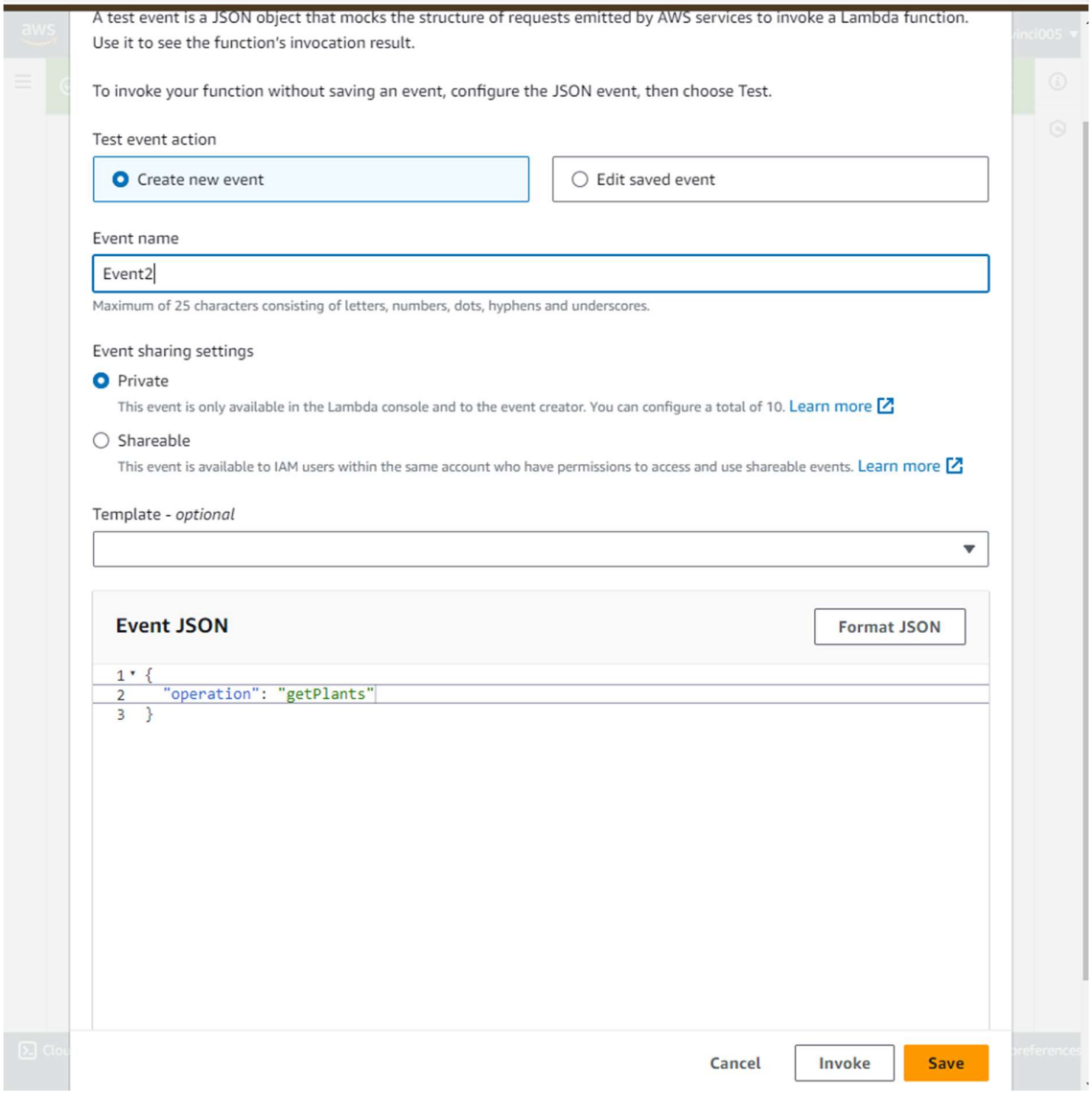

Response headers

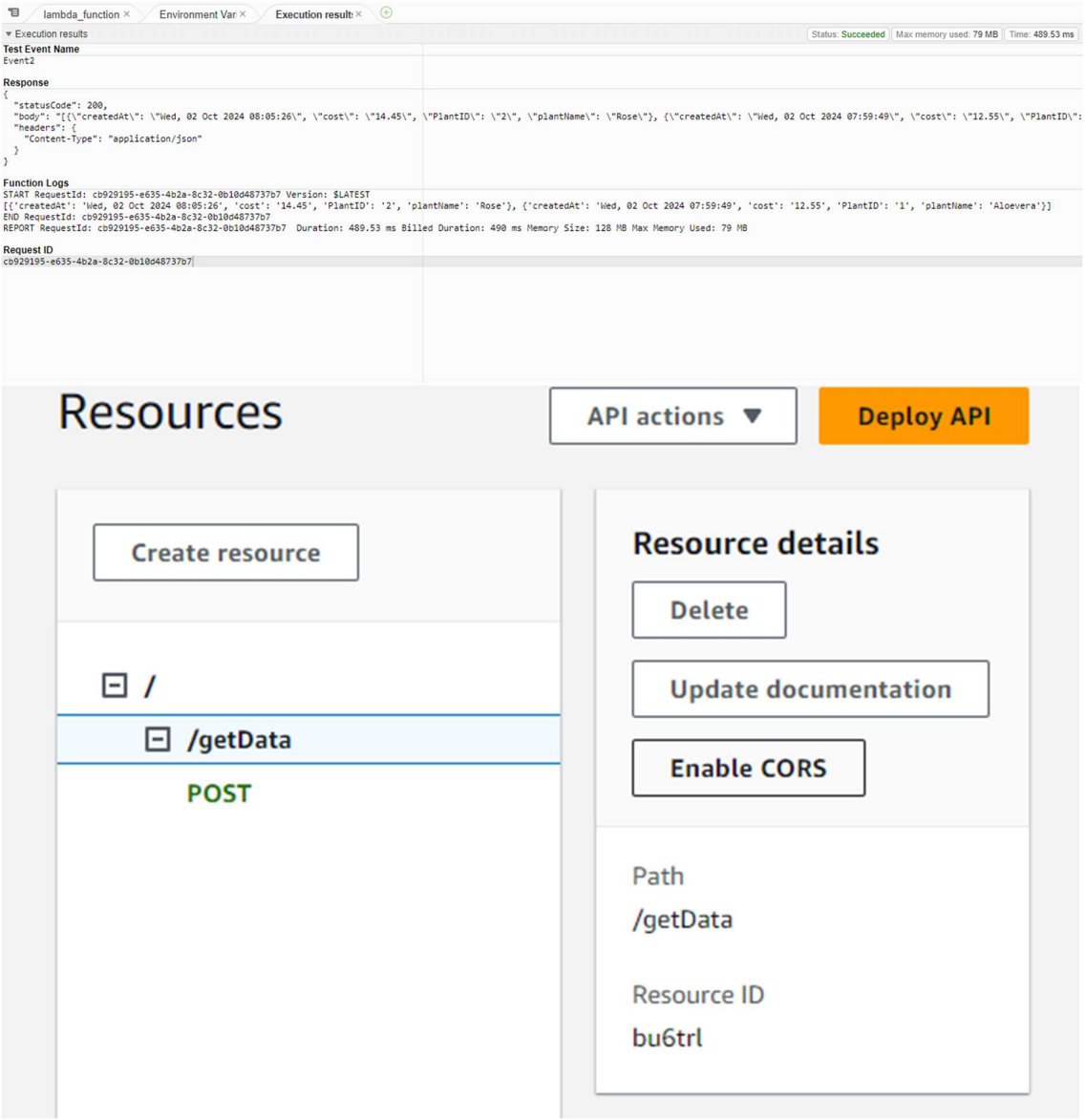

```
{
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-66fcfec6-
10b4dc60687fdacfc4575e43;Parent=382291edcb1e301d;
Sampled=0;Lineage=1:5abc95e2:0"
}
```


Logs

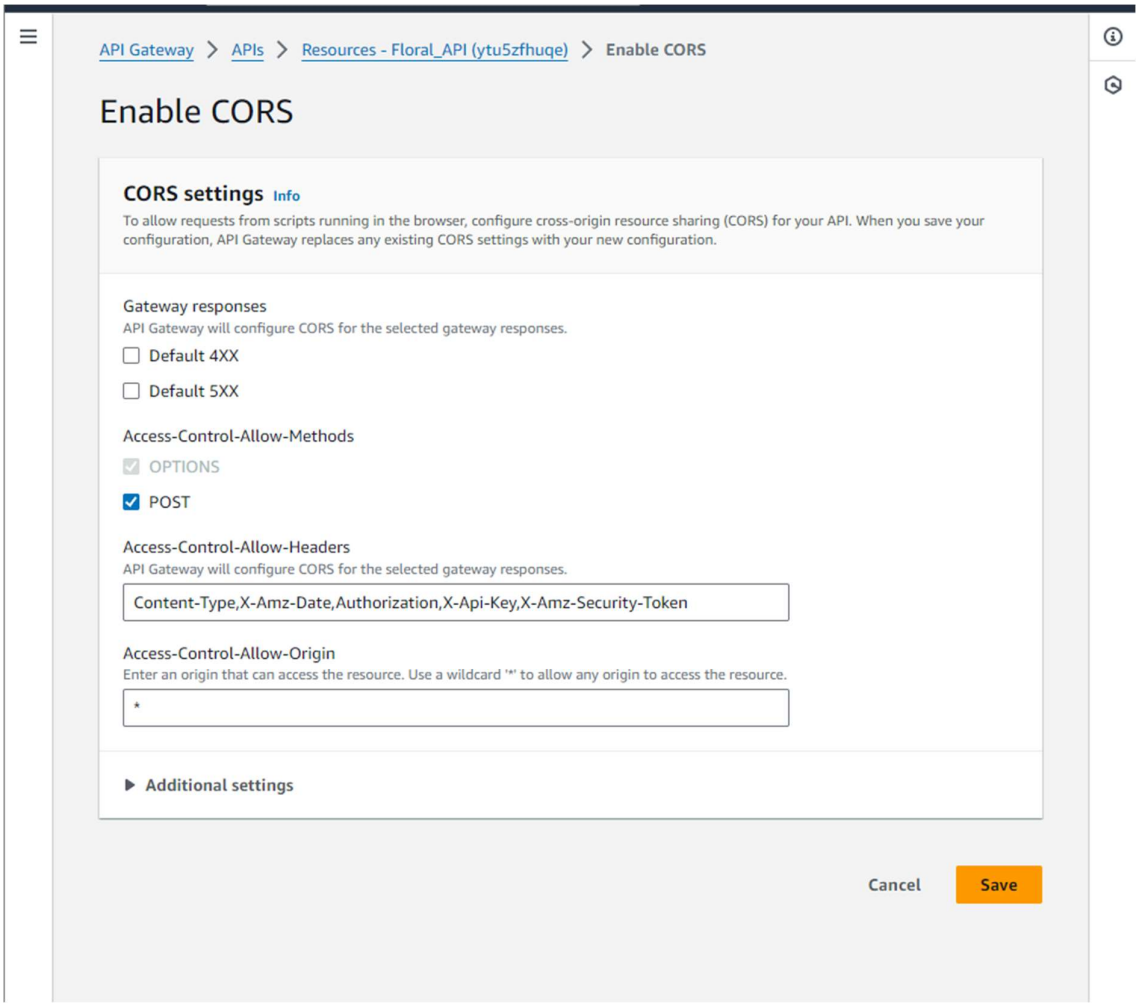
```
Execution log for request 8e5b3ec5-e1a2-4bde-
be7f-4622672fea6b
Wed Oct 02 08:05:26 UTC 2024 : Starting execution
for request: 8e5b3ec5-e1a2-4bde-be7f-4622672fea6b
Wed Oct 02 08:05:26 UTC 2024 : HTTP Method: POST,
Resource Path: /getData
Wed Oct 02 08:05:26 UTC 2024 : Method request
```


FloralBliss
Autopreview
View table details
Scan or query items
Scan
Query
Select a table or index
Table - FloralBliss
Select attribute projection
All attributes
Filters
Run
Reset
Completed. Read capacity units consumed: 0.5
Items returned (2)
Actions
Create item
1
PlantID (String)
cost
createdAt
plantName
2
14.45
Wed, 02 Oc...
Rose
1
12.55
Wed, 02 Oc...
Aloevera

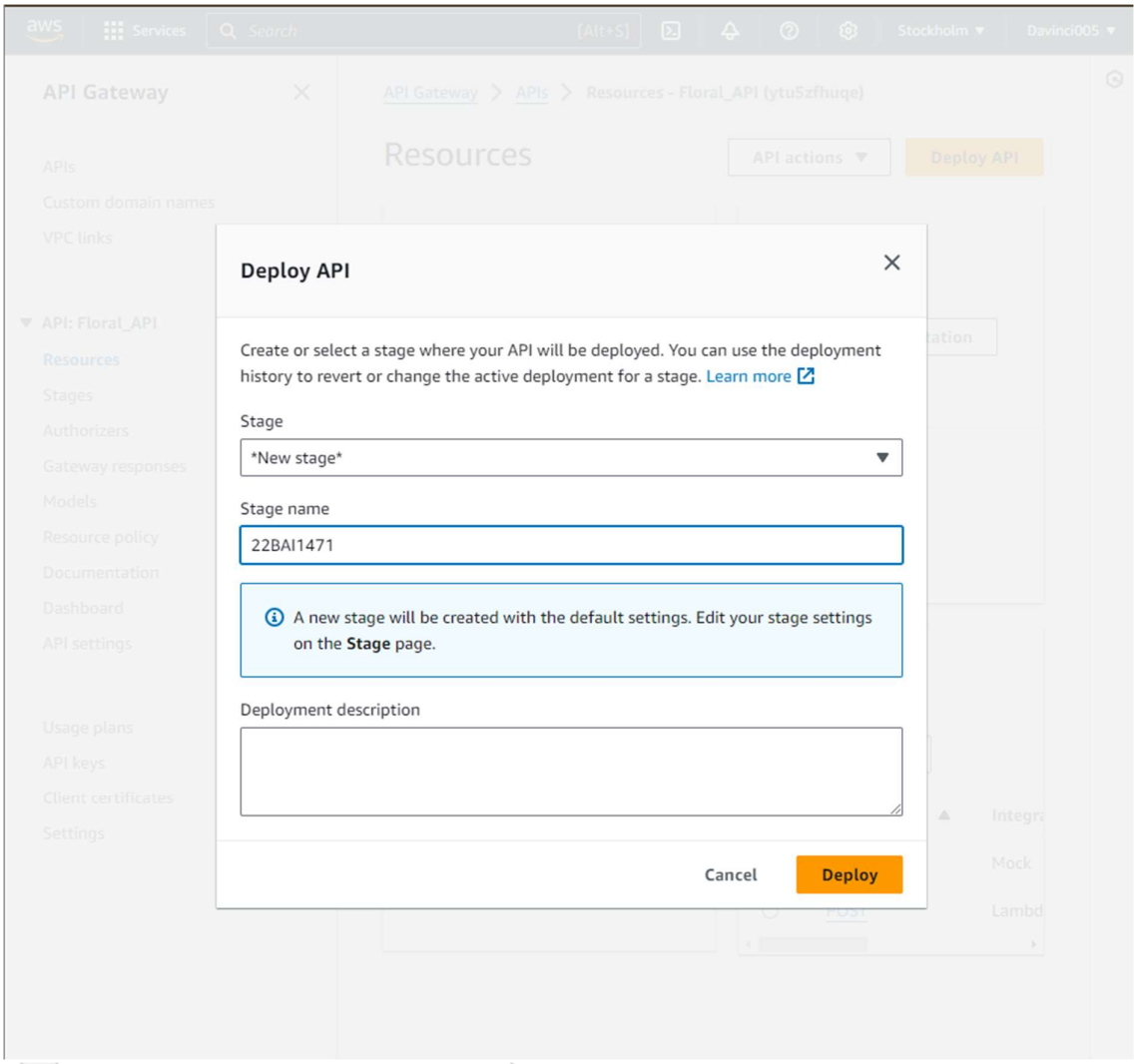

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, configure the JSON event, then choose Test.
Test event action
Create new event
Edit saved event
Event name
Event2
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
Event JSON
Format JSON
1 {
2 "operation": "getPlants"
3 }
Cancel
Invoke
Save

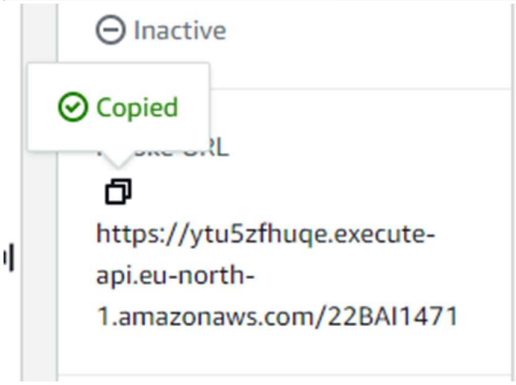


10) Now we need to enable CORS where post method is given access

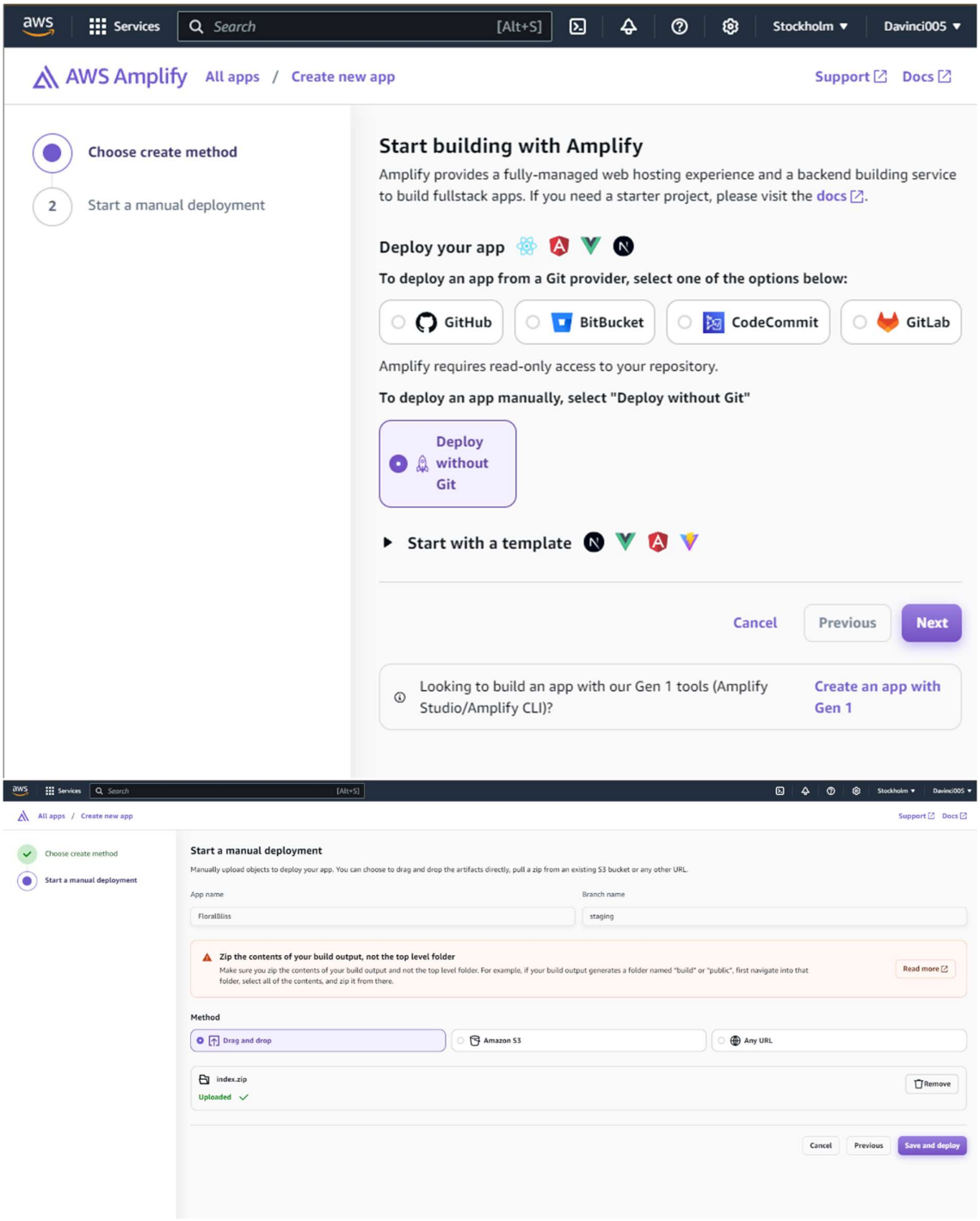


11) Now deploy the API

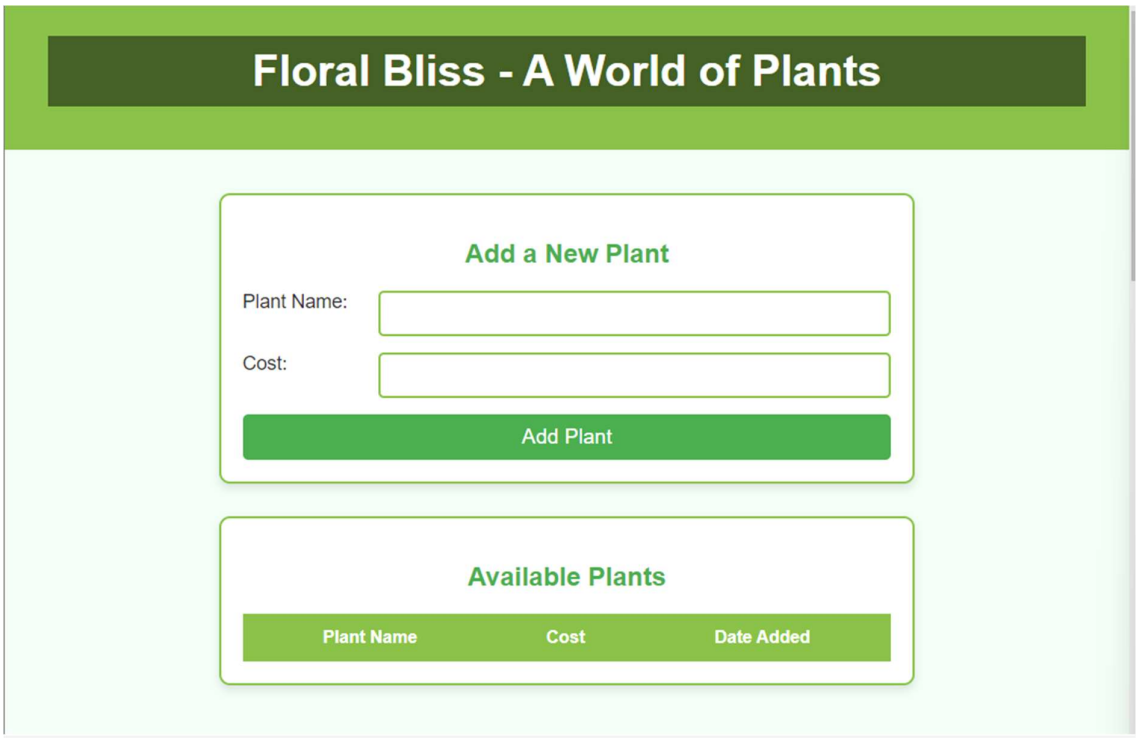


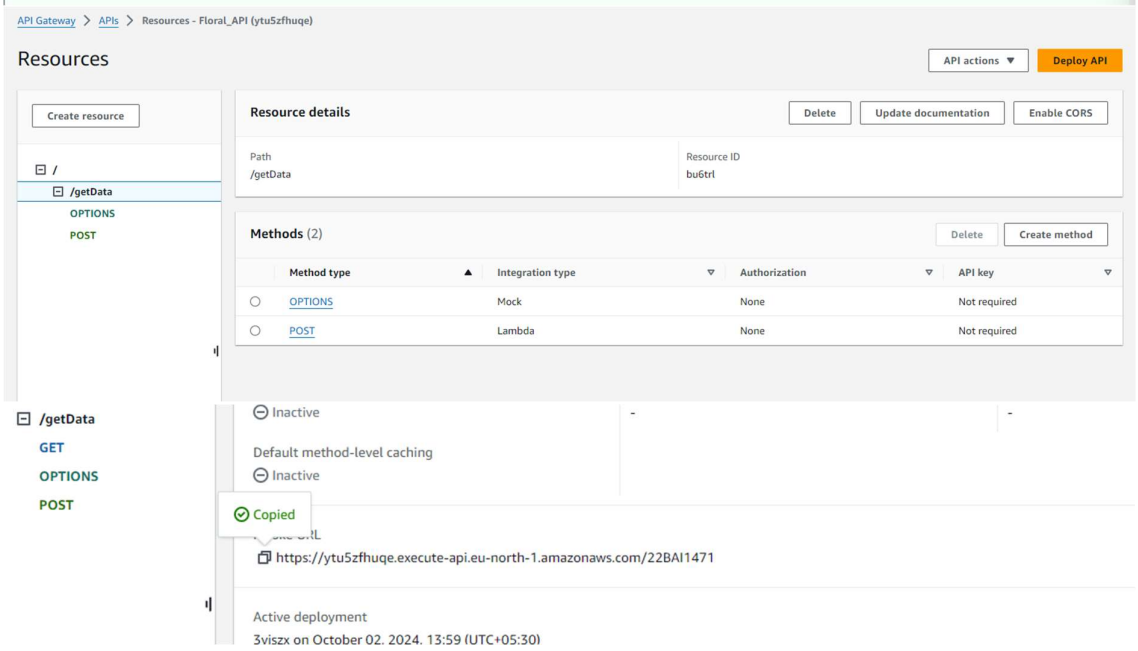


12) Now create a html file and make it as zip file that has overall website design and JavaScript for backend operations and upload the zip file into the aws amplify (deploy without git option) and hence the app URL will be available

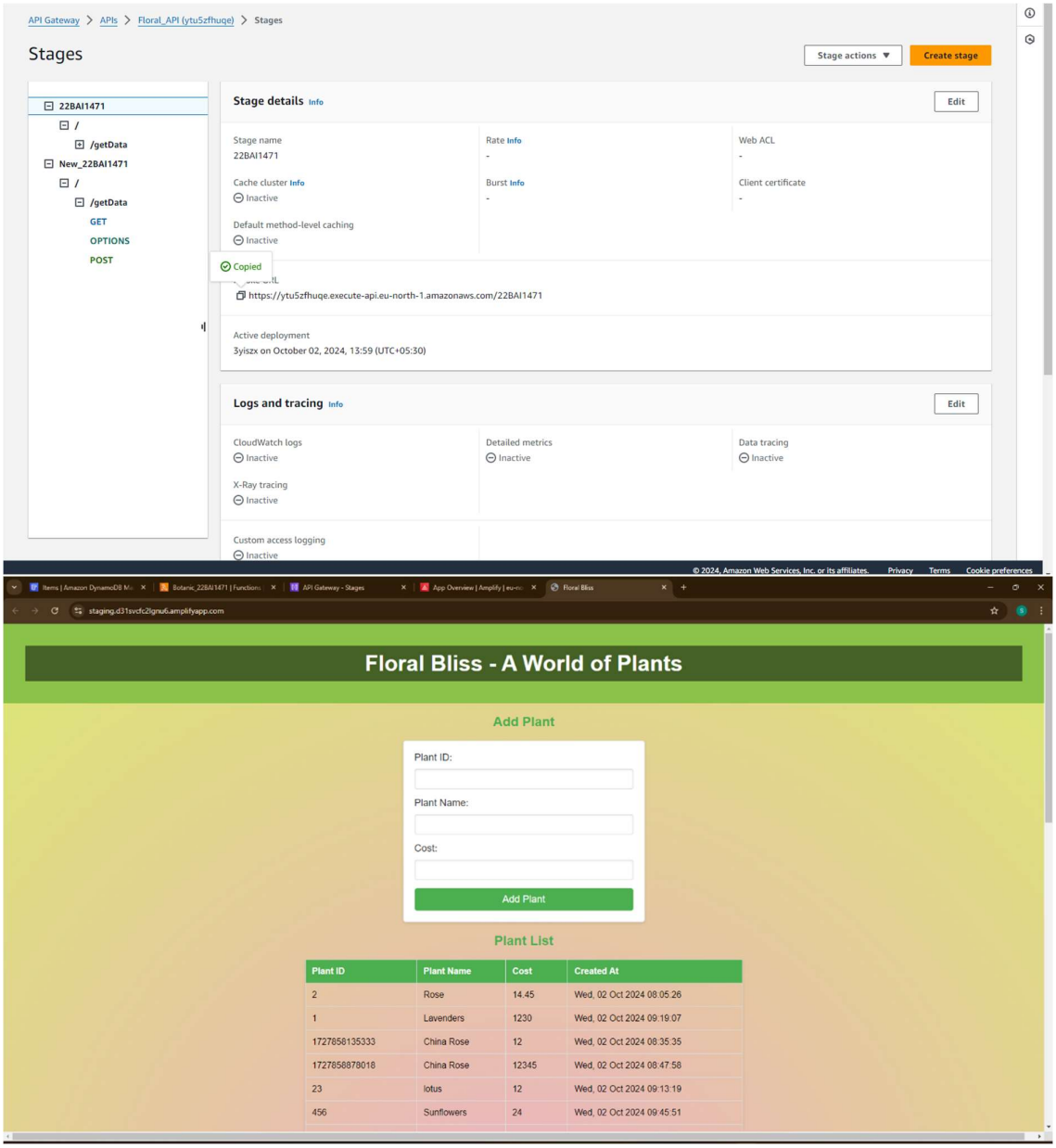

aws
Services
Search
[Alt+S]
Stockholm
Davinci005
AWS Amplify
All apps / Create new app
Support
Docs
Choose create method
2
Start a manual deployment
Start building with Amplify
Amplify provides a fully-managed web hosting experience and a backend building service to build fullstack apps. If you need a starter project, please visit the docs.
Deploy your app
To deploy an app from a Git provider, select one of the options below:
GitHub
BitBucket
CodeCommit
GitLab
Amplify requires read-only access to your repository.
To deploy an app manually, select "Deploy without Git"
Deploy without Git
Start with a template
Cancel
Previous
Next
Looking to build an app with our Gen 1 tools (Amplify Studio/Amplify CLI)?
Create an app with Gen 1

Choose create method
Start a manual deployment
Start a manual deployment
Manually upload objects to deploy your app. You can choose to drag and drop the artifacts directly, pull a zip from an existing S3 bucket or any other URL.
App name
FloralBliss
Branch name
staging
Zip the contents of your build output, not the top level folder
Make sure you zip the contents of your build output and not the top level folder. For example, if your build output generates a folder named "build" or "public", first navigate into that folder, select all of the contents, and zip it from there.
Read more
Method
Drag and drop
Amazon S3
Any URL
index.zip
Uploaded
Remove
Cancel
Previous
Save and deploy

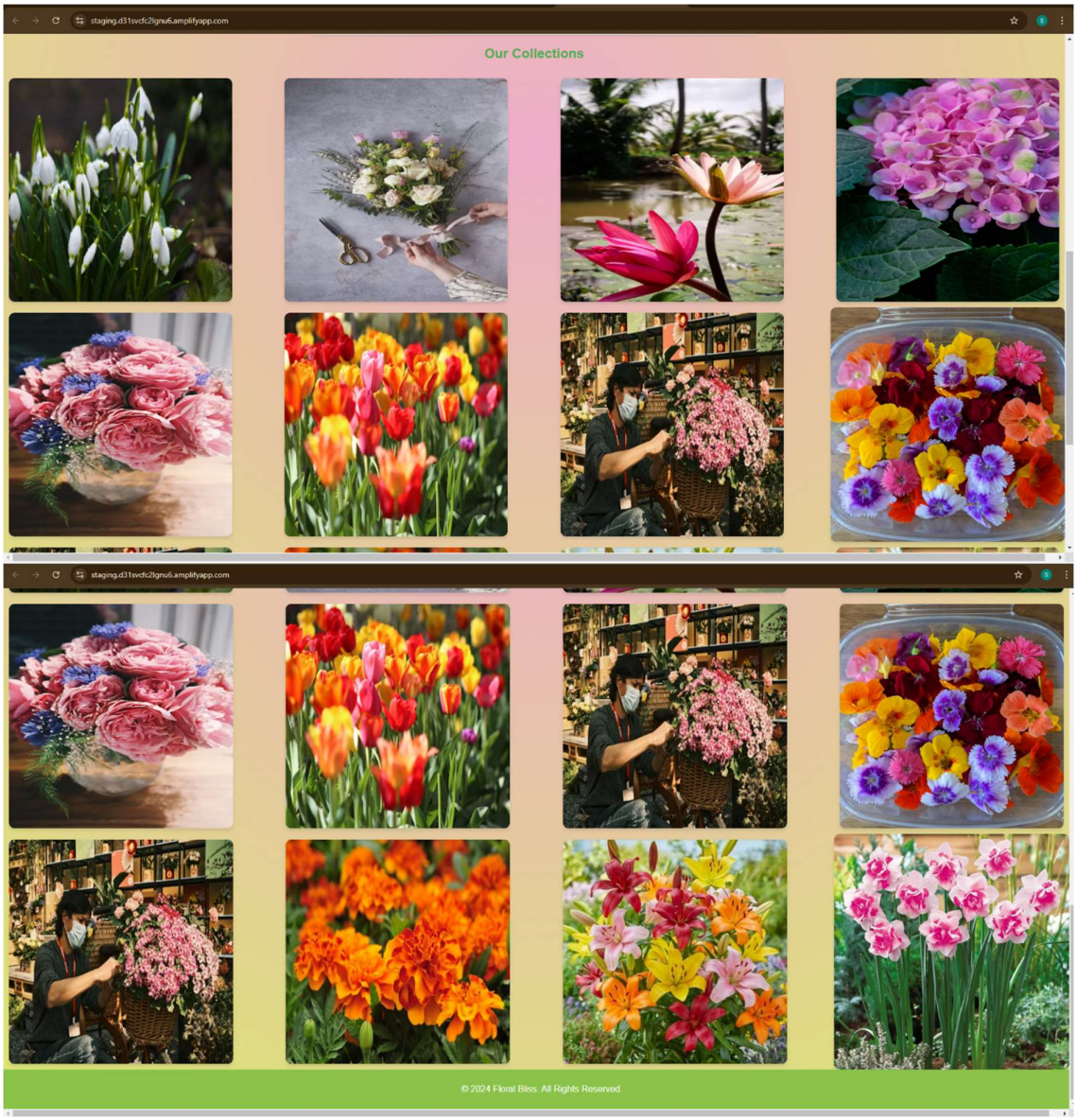

Floral Bliss - A World of Plants
Add a New Plant
Plant Name:
Cost:
Add Plant
Available Plants
Plant Name
Cost
Date Added


API Gateway > APIs > Resources - Floral_API (ytu5zfhuqe)
Resources
API actions
Deploy API
Create resource
Resource details
Delete
Update documentation
Enable CORS
/
/getData
OPTIONS
POST
Path
/getData
Resource ID
bu6trl
Methods (2)
Delete
Create method
Method type
Integration type
Authorization
API key
OPTIONS
Mock
None
Not required
POST
Lambda
None
Not required

/getData
GET
OPTIONS
POST
Inactive
-
-
Default method-level caching
Inactive
Copied
https://ytu5zfhuqe.execute-api.eu-north-1.amazonaws.com/22BAI1471
Active deployment
3viszx on October 02, 2024, 13:59 (UTC+05:30)


API Gateway > APIs > Floral_API (ytu5zfhuqe) > Stages
Stages
Stage actions
Create stage
22BAI1471
/
/getData
New_22BAI1471
/
/getData
GET
OPTIONS
POST
Stage details Info
Edit
Stage name
22BAI1471
Rate Info
-
Web ACL
-
Cache cluster Info
Inactive
Burst Info
-
Client certificate
-
Default method-level caching
Inactive
Copied
https://ytu5zfhuqe.execute-api.eu-north-1.amazonaws.com/22BAI1471
Active deployment
3yiszx on October 02, 2024, 13:59 (UTC+05:30)
Logs and tracing Info
Edit
CloudWatch logs
Inactive
Detailed metrics
Inactive
Data tracing
Inactive
X-Ray tracing
Inactive
Custom access logging
Inactive
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

staging.d31svcfc2lgnu6.amplifyapp.com
Floral Bliss - A World of Plants
Add Plant
Plant ID:
Plant Name:
Cost:
Add Plant
Plant List
Plant ID | Plant Name | Cost | Created At
2 | Rose | 14.45 | Wed, 02 Oct 2024 08:05:26
1 | Lavenders | 1230 | Wed, 02 Oct 2024 09:19:07
1727858135333 | China Rose | 12 | Wed, 02 Oct 2024 08:35:35
1727858878018 | China Rose | 12345 | Wed, 02 Oct 2024 08:47:58
23 | lotus | 12 | Wed, 02 Oct 2024 09:13:19
456 | Sunflowers | 24 | Wed, 02 Oct 2024 09:45:51



Overall steps

Step 1: Create a DynamoDB Table

1. **Go to DynamoDB Console**:
    - Navigate to the DynamoDB Console.
2. **Create a Table**:
    - Click on **Create Table**.
    - Define a **Table Name** (e.g., FloralBliss).
    - Choose a **Partition Key** (Primary Key) like PlantID (String). This serves as the unique identifier for each plant.
    - Choose **On-Demand** or **Provisioned** throughput as needed.
    - Click **Create** to finish.

Reason: The table will store the plant details, with PlantID as the primary key to ensure each plant has a unique identifier.

Step 2: Create a Lambda Function

1. **Navigate to the Lambda Console**:
    - Go to the AWS Lambda Console.
2. **Create a New Function**:
    - Click **Create Function**.
    - Choose **Author from Scratch**.
    - Name your function (e.g., PlantHandlerFunction).
    - Choose your desired language (e.g., Python or Node.js).
    - For **Execution Role**, select **Create a New Role with Basic Lambda Permissions**.
3. **Write Lambda Code**:
    - Inside your Lambda function, write code to handle both reading (scanning) and writing (putting) data into DynamoDB. (Refer to the code provided earlier for get_plants and add_plant functionality.)

Reason: The Lambda function will act as the backend logic to handle plant data, performing read and write operations on the DynamoDB table.

Step 3: Grant Lambda Function Access to DynamoDB

1. **Create an Inline Policy**:
    - In the **Lambda Console**, click on the **Permissions** tab.
    - Under **Execution Role**, click on the role associated with your function.
    - In the **IAM Console**, go to **Inline Policies** and select **Create Policy**.
2. **Configure Policy for DynamoDB**:
    - Choose **DynamoDB** as the service.
    - Select the actions PutItem (to write) and Scan (to read).
    - In the resources section, choose your specific **DynamoDB table** (e.g., FloralBliss).
    - Save the policy and attach it to your Lambda execution role.

Reason: The Lambda function requires explicit permissions to access DynamoDB, enabling it to read and write data securely.

Step 4: Test Lambda Function

1. **Create a Test Event**:
    - In the Lambda console, click **Test**.
    - Provide a sample JSON test event, like the following:

json

Copy code

{

"operation": "addPlant",

"PlantID": "001",

"Name": "Rose",

"Category": "Outdoor",

"Price": 10,

"Description": "Beautiful red rose"

}

- Click **Test** to ensure your Lambda function works correctly and stores data in the DynamoDB table.

Reason: Testing verifies that the Lambda function can successfully write data to the table.

Step 5: Create an API Gateway

1. **Navigate to API Gateway Console**:
   - Go to the API Gateway Console.
2. **Create a New REST API**:
   - Click **Create API** and choose **REST API**.
   - Name your API (e.g., FloralBlissAPI).
   - Create a resource (e.g., /getData).
3. **Add POST and GET Methods**:
   - Under the /getData resource, add **POST** and **GET** methods.
   - Link both methods to your Lambda function (PlantHandlerFunction).
   - Ensure both methods are set to use the **Lambda Proxy Integration**.

Reason: API Gateway provides a REST API interface to interact with your Lambda function and thus access the data in DynamoDB.

Step 6: Enable CORS on the API

1. **Enable CORS**:
   - In API Gateway, go to the **POST** and **GET** methods and click **Enable CORS**.
   - This allows your front-end (HTML/JS) to make requests from the browser to your API.

Reason: CORS is necessary to allow the front-end to communicate with your API securely.

Step 7: Deploy the API

1. **Deploy API**:
   - In API Gateway, click on **Deploy API**.
   - Create a new **stage** (e.g., prod or dev) and deploy it.

- Take note of the **API endpoint URL**.

Reason: Deployment makes the API publicly available, so it can be accessed by your front-end code.

Step 8: Create Front-End (HTML/JavaScript)

1. **Design Front-End**:
   - Create an HTML file with a form to add plant data and display existing plants.
   - Use **JavaScript** to send **POST** requests to the API to add plants and **GET** requests to display plants (see code provided in earlier steps).
2. **Zip Front-End Files**:
   - Once the front-end code is ready, zip all the files (index.html, scripts.js, etc.).

Reason: The front-end provides users with an interface to interact with the API (e.g., adding new plants and viewing the list of plants).

Step 9: Deploy Front-End with AWS Amplify

1. **Go to AWS Amplify Console**:
   - Navigate to the AWS Amplify Console.
2. **Deploy Without Git**:
   - In the Amplify Console, select **Deploy Without Git**.
   - Upload the zip file containing your front-end code.
3. **Configure and Deploy**:
   - Amplify will build and deploy your front end.
   - Once the deployment is complete, you will get a **URL** where your app is accessible.

Reason: AWS Amplify is used to host and deploy your front-end code, making your app accessible to users via a web URL.

Step 10: Test the Full Application

1. **Go to Your App's URL**:
   - Use the URL provided by Amplify to access your app.
   - Test adding a plant via the form and verify that it shows up in the list of available plants.
   - Check if the back-end API is working correctly by interacting with the DynamoDB table.

Reason: This step ensures that all components (front-end, API, Lambda, DynamoDB) are working together as intended.