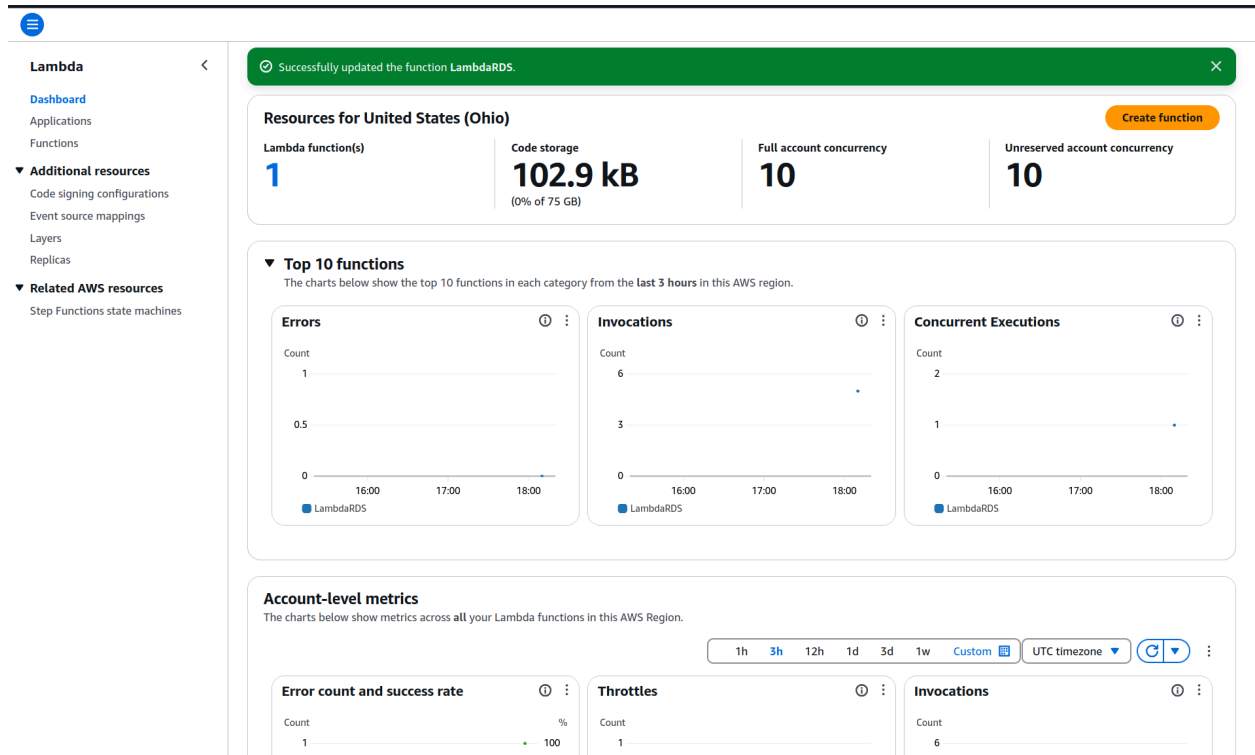


Create Lambda Function

Go to dashboard and create the function



Name your function something descriptive. We'll be using Python 3.10 (because I said so)

Basic information

Function name
Enter a name that describes the purpose of your function.

FormPostInsertRDS

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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 3.10

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](#)

For additional configurations, enable function url for the function to get a public url. Disable authentication for it.

▼ **Additional Configurations**
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

☐ **Enable Code signing** [Info](#)
Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.

☐ **Enable encryption with an AWS KMS customer managed key** [Info](#)
By default, Lambda encrypts the .zip file archive using an AWS owned key.

☒ **Enable function URL** [Info](#)
Use function URLs to assign HTTP(S) endpoints to your Lambda function.

Auth type
Choose the auth type for your function URL. [Learn more](#)

☐ **AWS_IAM**
Only authenticated IAM users and roles can make requests to your function URL.

☒ **NONE**
Lambda won't perform IAM authentication on requests to your function URL. The URL endpoint will be public unless you implement your own authorization logic in your function.

Function URL permissions

Create the function

Upload the zip layer attached

Layers (0)

[+ Add destination](#)

23 seconds ago

Function ARN
[arn:aws:lambda:us-east-2:831926625893:function:FormPostInsertRDS](#)

Function URL [Info](#)
<https://tckvd5uxbic43mc7ozhzeqjhay0xokog.lambda-url.us-east-2.on.aws/>

Aliases | **Versions**

Upload from ▲ ▼

.zip file

Amazon S3 location

FormPostInsertRDS

```
da_function.py x
bda_function.py
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
```

Changing configurations

We need to change several configurations to make it work with our database

Timeout

Change the timeout to 30 seconds, so that the function has time to make a connection

CodeTestMonitorConfigurationAliasesVersions

General configuration

TriggersPermissionsDestinationsFunction URL

General configuration info

Description-

Timeout0 min 3 sec

Memory128 MB

SnapStart infoNone

Ephemeral storage512 MB

Edit

Environment Variables

This is the same as your .env file that you created. This will store the credentials to access the database.

CodeTestMonitorConfigurationAliasesVersions

General configuration

TriggersPermissionsDestinationsFunction URL

Environment variables (0)

Find environment variables

Key

Value

No environment variables
No environment variables associated with this function.

Edit

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

Key

Value

DB_HOST

portfolordserver.cjgoi4kygls9.us-east-2.rds.amazonaws.com

Remove

DB_USER

admin

Remove

DB_PASSWORD

avwmKKEE7H5xtfQcEm9O

Remove

DB_NAME

form

Remove

DB_PORT

3306

Remove

Add environment variable

► Encryption configuration

Cancel Save

RDS Database Network Connection

We need to connect the database and the function with a network connection.

General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools

Function not connected to a VPC
When a function is not connected to a VPC, the following table will be empty. You can configure a VPC separately or when connecting to an RDS database.

RDS database connections (0)
Add Proxy
Connect to RDS database

The following table shows databases that match this function's VPC and security group. Review the details to confirm a connection.

DB identifier	Proxy identifier	Database Status	Lambda security group	DB security group
No connected databases				
You don't have any connected databases				
Connect to RDS database				

Connect to RDS database

RDS database
Connect FormPostInsertRDS (not attached to a VPC) to an RDS database.

☒ Use an existing database
☐ Create a new database

RDS database
portfoliordserver

Lambda will configure security groups and add the required execution role permissions for the function. Lambda will then connect your function to the existing database VPC.

Advanced permissions
By default, Lambda console will attempt to add the necessary permissions to your function's execution role.

Cancel
Create

Add some test events

Create a successful test

```
{
  "httpMethod": "POST",
  "body": "{\"name\":\"Test User\",\"email\":\"test@example.com\",\"subject\":\"Test Subject Line\",\"message\":\"This is a test message\"}"
}
```

Put in JSON and then save

[AWS Lambda console](#)

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

Test event [Info](#)

CloudWatch Logs Live Tail Save Test

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

GoodTest

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

```
1 {
2   "httpMethod": "POST",
3   "body": "{\"name\":\"Test User\",\"email\":\"test@example.com\",\"subject\":\"Test Subject Line\",\"message\":\"This is a test message\"}"
4 }
5
```

Create some bad tests

```
{
  "body": "{\"name\":\"Test User\",\"email\":\"test@example.com\",\"message\":\"This is a test message\"}"
}
```

Put in JSON and then save

Test event [Info](#)

CloudWatch Logs Live Tail Save Test

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

BadTest1

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

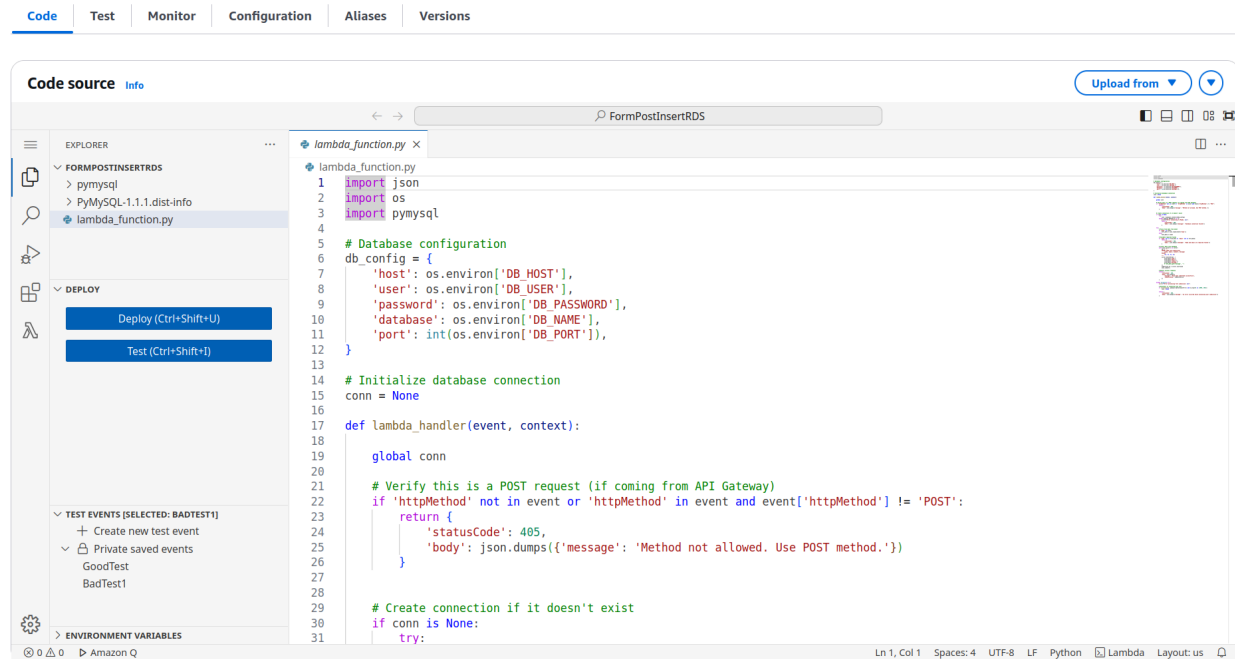
Format JSON

```
1 {
2   "body": "{\"name\":\"Test User\",\"email\":\"test@example.com\",\"subject\":\"Test Subject Line\",\"message\":\"This is a test message\"}"
3 }
4
```

Run your function

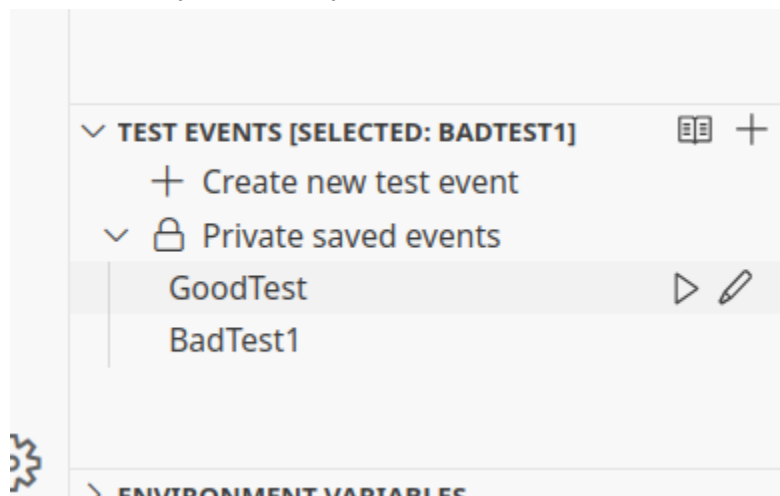
Deploy your code.

The function is not updated until you deploy your code to it. **Press ctrl+s on the py file to save it**, then press deploy



Run the tests

Press the play button on your saved tests



You should get outputs like this



The first screenshot shows a successful test event named 'GoodTest'. The response is a JSON object with a status code of 200 and a body containing a success message and a submission ID.

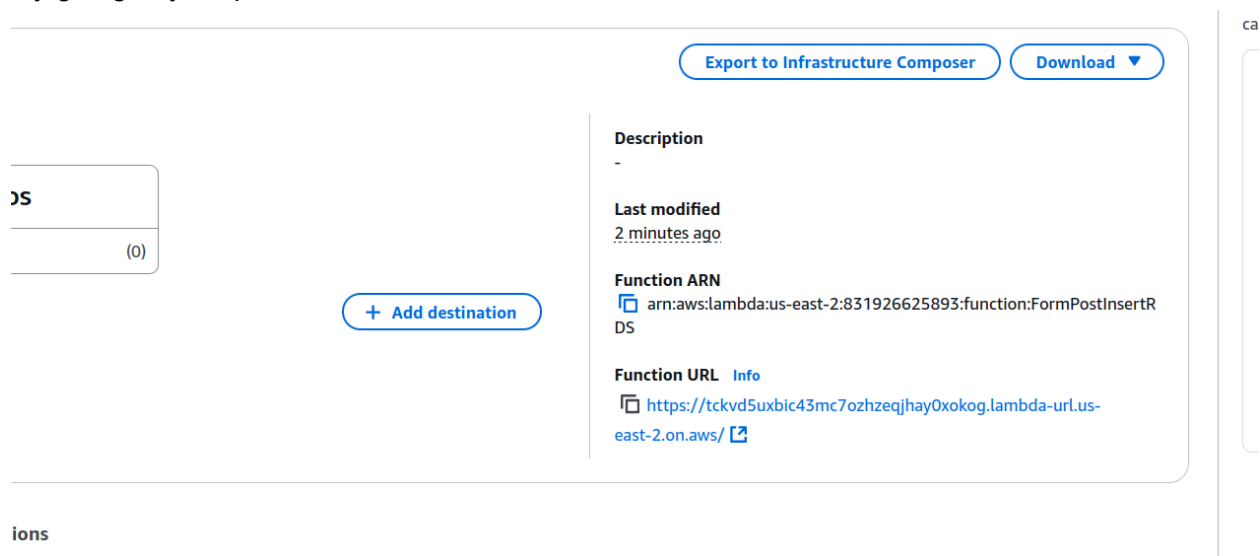
```
Response:
{
  "statusCode": 200,
  "body": "{\"message\": \"Form data submitted successfully\", \"submissionId\": 4}"
}
```

The second screenshot shows a failed test event named 'BadTest1'. The response is a JSON object with a status code of 405 and a body containing an error message about the HTTP method.

```
Response:
{
  "statusCode": 405,
  "body": "{\"message\": \"Method not allowed. Use POST method.\"}"
}
```

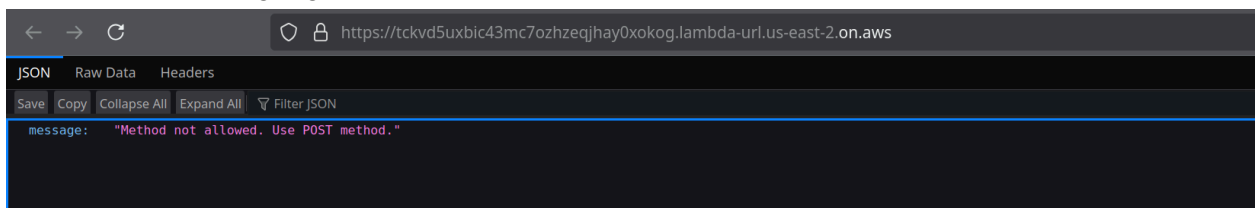
Try the public link

Try going to your public url now to test it



The screenshot shows the AWS Lambda console for a function named 'FormPostInsertRDS'. The function is in the 'Ready' state. The 'Description' field is empty. The 'Last modified' timestamp is '2 minutes ago'. The 'Function ARN' is 'arn:aws:lambda:us-east-2:831926625893:function:FormPostInsertRDS'. The 'Function URL' is 'https://tckvd5uxbic43mc7ozhzeqjhay0xokog.lambda-url.us-east-2.on.aws/'. There are buttons for 'Export to Infrastructure Composer', 'Download', and '+ Add destination'.

Because we're using a get request, it won't work.



The screenshot shows a web browser with the address bar displaying 'https://tckvd5uxbic43mc7ozhzeqjhay0xokog.lambda-url.us-east-2.on.aws'. The browser shows a 405 error message: 'message: \"Method not allowed. Use POST method.\"'.

Now add it to your website.