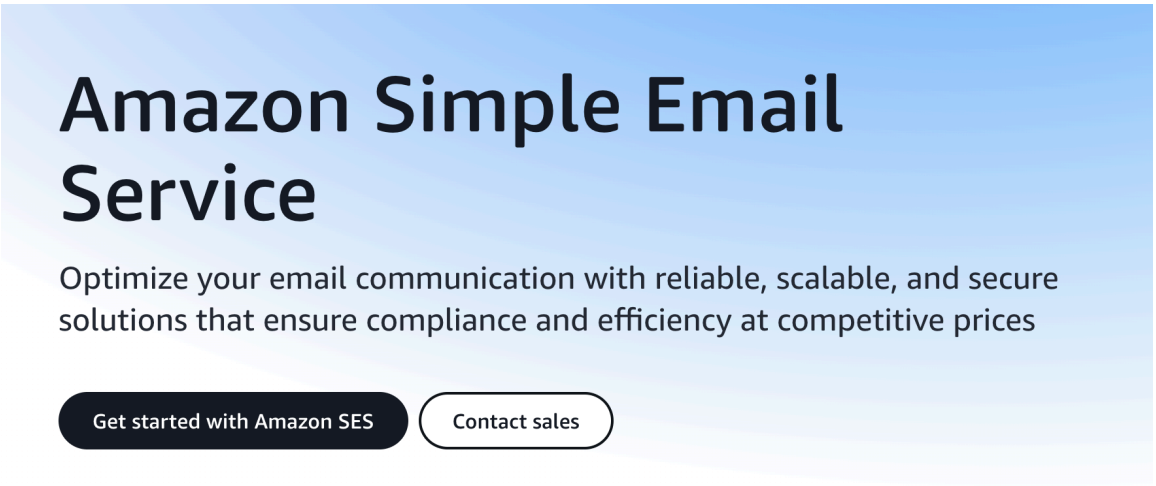
Project 2 - SES

This project is going to build off what I've learned so far this semester. At the end, it will utilize Serverless, NOSQL databases, IAM, CloudWatch, and a few new services to create a system that will automatically send a “stat of the day” email.

Section 1: Simple Email Service (SES) Introduction setup

Source: https://aws.amazon.com/ses/

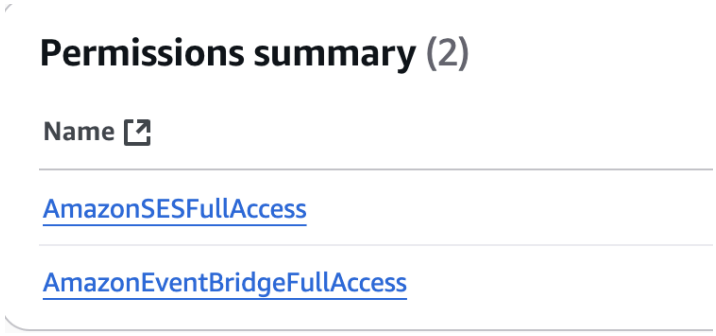
> We’ll use SES with Lambda to send out a notification email when an event is triggered. SES (Simple Email Service) is a cloud-based email service provider that can integrate into any application for high-volume email automation.



Section 2: IAM Permissions and setup

1. Before we can use AWS SES, we need to give our IAM account permissions to do so.
2. Sign into AWS console using your **root** account.
3. Add permissions “**AmazonSESFullAccess**” and “**AmazonEventBridgeFullAccess”** to your EC2 IAM user.
    - Be sure to “attach policies directly”

It should look like this:

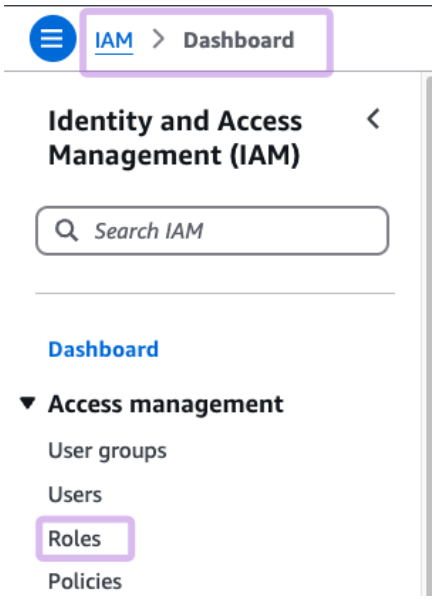


Section 3: IAM roles – EC2

We'll be testing our code in VS Code, and then eventually applying it to Lambda, which will also use DynamoDB and SES. So, we need to establish **Roles** to permit these services to communicate with EC2.

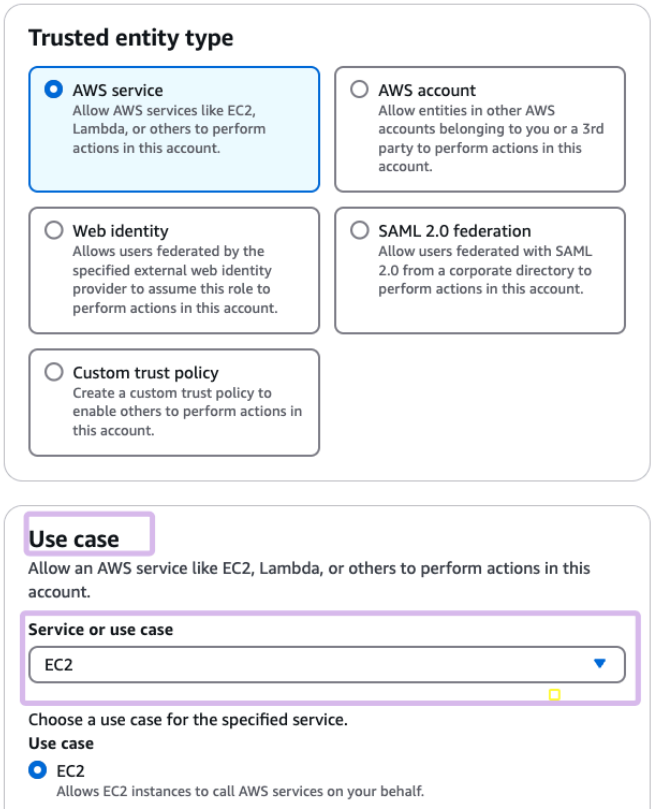
So, we'll need to create new IAM **roles** with the appropriate permissions to allow this to happen.

1. In the AWS console (still with your Root account), go to IAM (Identity and Management)
2. Select Roles



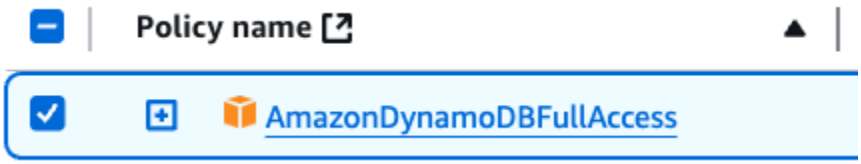
3. Create a new role 

4. Select "EC2" as the common use case



5. Click on the Next button

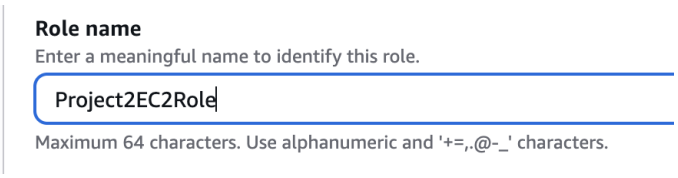
6. Add permission for **AmazonDynamnoDBFullAccess** and **SESFullAccess**



Before you click a button. Also add permissions for “SESFullAccess”

7. Then click on “Next”

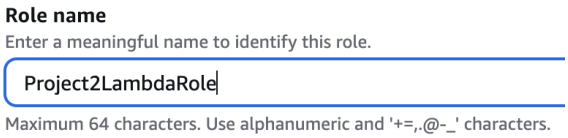
8. Call the Role: “Project2EC2Role”

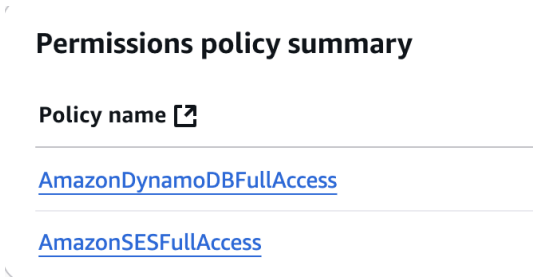


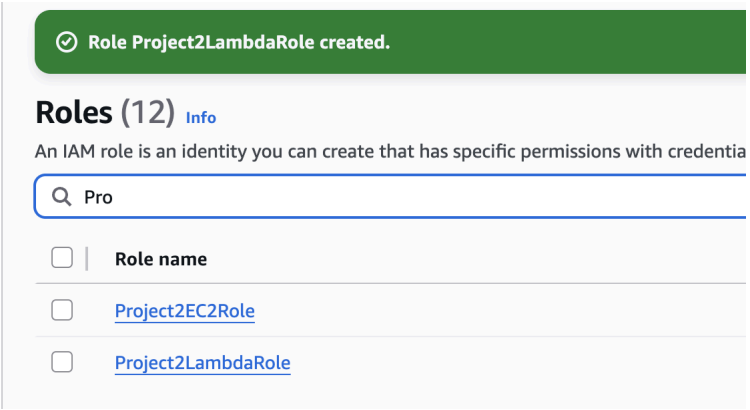
9. Click on “Create role”

Section 4: IAM roles -- Lambda

Repeat Section 3. Except this time in step 4, select “Lambda” as the common use case. Repeat all of the other steps the same except name this role “Project2LambdaRole”







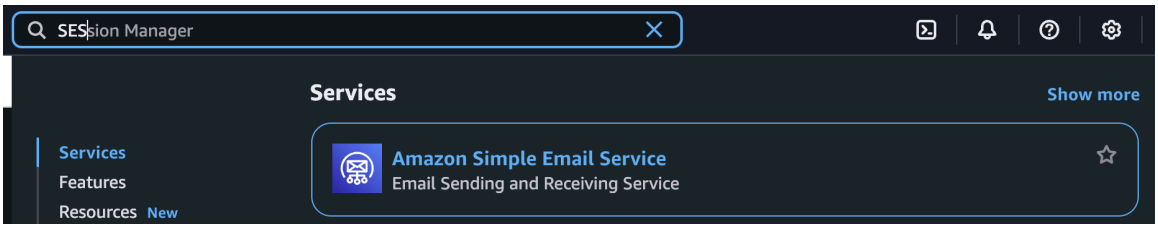
Section 5: Update your EC2 IAM

1. You'll need to update the Role for the EC2 instance to the new role created in Section 3.
2. Go to the AWS EC2 dashboard
3. Select your instance
4. Select Actions → Security → Modify IAM role
5. Select the Project2EC2Role
6. Press "Update IAM role"
7. Log out of your Root Account
8. Log in to your EC2 Account

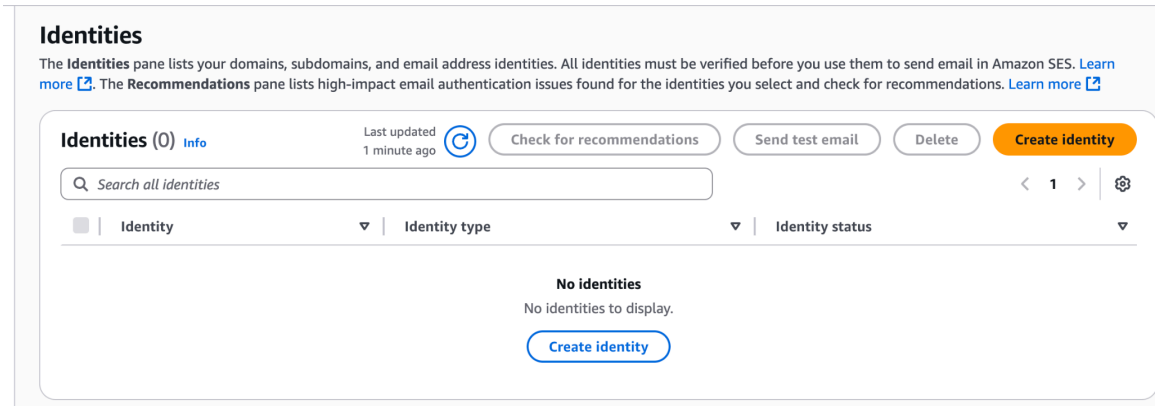
Section 6: SES

In this section, you'll use **Amazon SES (Simple Email Service)** to send your stat of the day via email. SES lets you send messages to a "topic," and anyone subscribed to that topic will receive the messages by email, SMS, etc.

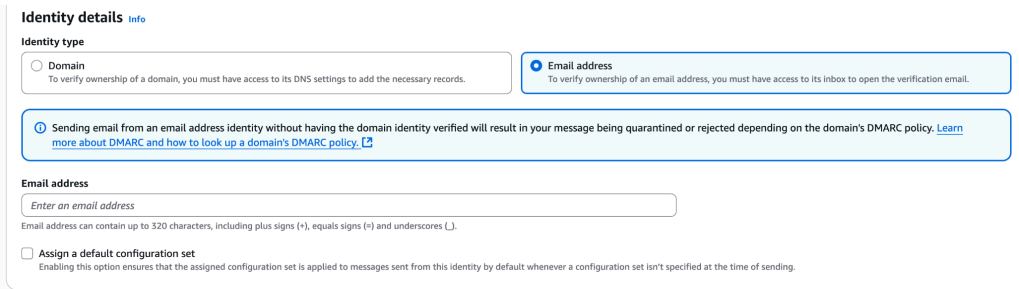
1. In the AWS Console, go to SES



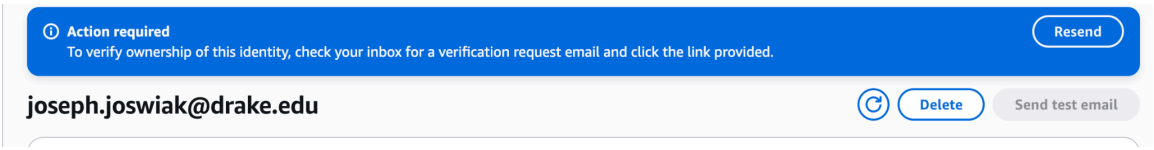
1. Click on **Identities** on the left-hand side



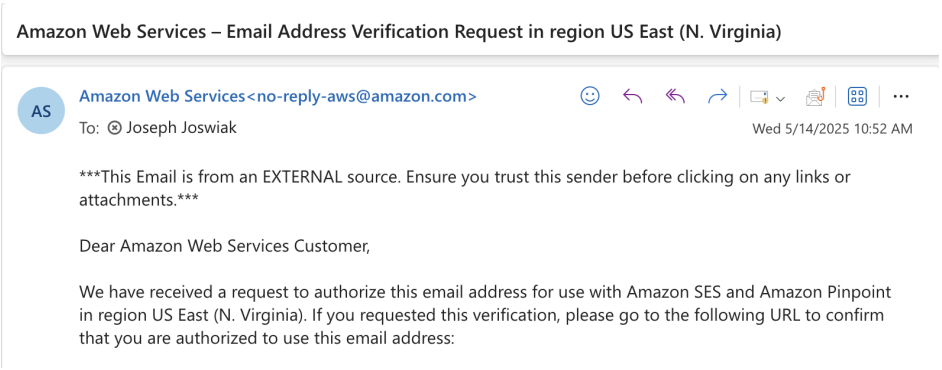
2. Click Create Identity
    - click “Next”
3. Choose Email address and enter your email address



4. Click Create Identity

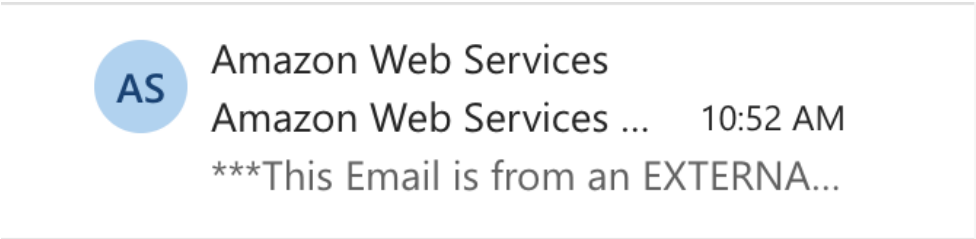


5. Verify your identity by checking your inbox and clicking the confirmation link

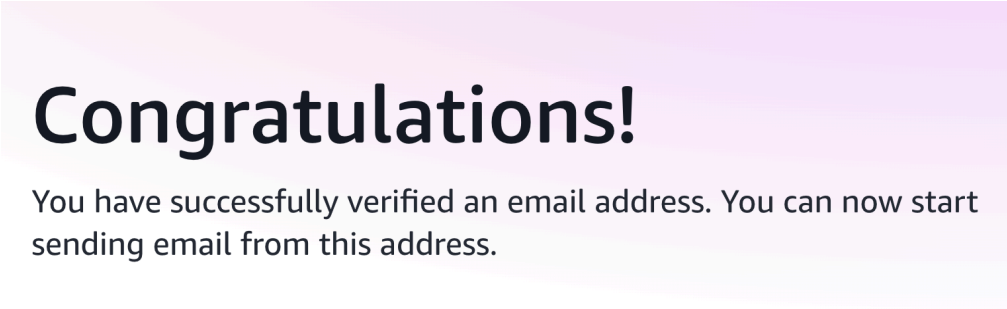


6. Confirm the subscription:

a. Go to your email inbox, open the confirmation email from “Amazon Web Services”



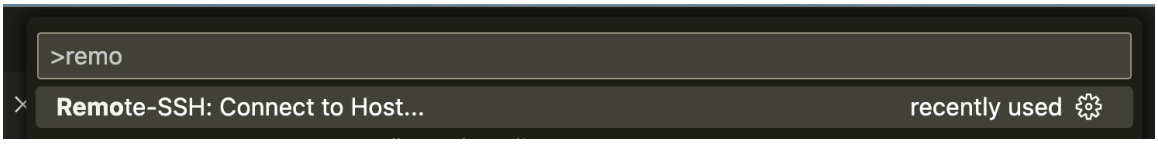
b. Click on the link to confirm
c. Your subscription status should now say “Congratulations” in the AWS console.

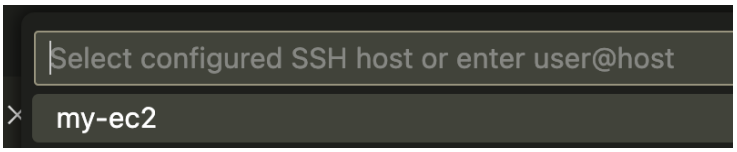


Section 7: Python for Sending Email

1. Connect to your EC2 instance:

   a. Use the Remote-SSH extension, and connect to your EC2 instance via the extension.





2. Open a Terminal:
3. Now, go ahead and clone the starter repository to your EC2 instance: https://github.com/merriekay/cs178-lab16-starter with the following command in your EC2 Terminal:

```
git clone https://github.com/merriekay/cs178-lab16-starter.git
```

- This will download the starter code files sendJoke.py and emailTest.py to your EC2 instance.

```
[ec2-user@ip-172-31-20-48 ~]$ git clone https://github.com/merriekay/cs178-lab16-starter.git
Cloning into 'cs178-lab16-starter'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 18 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (18/18), 5.89 KiB | 1.96 MiB/s, done.
Resolving deltas: 100% (7/7), done.
[ec2-user@ip-172-31-20-48 ~]$
```

- You can skip this step if you have it from Lab16Lambda2

4. Now, if you type **ls** into the terminal, you should see a new folder on your EC2 instance called cs178-lab16-starter.
   - You can skip this step if you have it from Lab16Lambda2

```
[ec2-user@ip-172-31-80-102 ~]$ ls
CS178_Project1  FlaskApp  cs178-lab16-starter  emailTest.py  exit  movies.sql  mysql80-community-release-el9-1.noarch.rpm
[ec2-user@ip-172-31-80-102 ~]$
```

5. Let's change directory into the cs178-lab16-starter by running:
   - **cd cs178-lab16-starter**
   - You can skip this step if you have it from Lab16Lambda2

```
[ec2-user@ip-172-31-80-102 ~]$ cd cs178-lab16-starter
[ec2-user@ip-172-31-80-102 cs178-lab16-starter]$
```

6. Now, run the command **ls** to list the contents of the folder. You should see something like this:

```
[ec2-user@ip-172-31-80-102 cs178-lab16-starter]$ ls
README.md  emailTest.py  sendJoke.py
[ec2-user@ip-172-31-80-102 cs178-lab16-starter]$
```

7. We'll be using the built in file editor nano for the Terminal to make a couple of small updates to these files.
   a. Run the command: **nano emailTest.py**

```
[ec2-user@ip-172-31-20-48 cs178-lab16-starter]$ nano emailTest.py
```

8. Replace everything in emailTest.py with this SES + DynamoDB version:

```
import boto3
import random
from botocore.exceptions import ClientError

def lambda_handler(event, context):
# DynamoDB setup
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('Project2Stats')

# Pull random stat
    response = table.scan()
    stat = random.choice(response['Items'])['stat']

# SES setup
```

```
ses = boto3.client('ses', region_name='us-east-1')

SENDER = "your_verified_email@example.com"
RECIPIENT = "your_verified_email@example.com"

try:
    ses.send_email(
        Source=SENDER,
        Destination={'ToAddresses': [RECIPIENT]},
        Message={
            'Subject': {'Data': 'Stat of the Day'},
            'Body': {'Text': {'Data': stat}}
        }
    )
    print("Email sent!")
except ClientError as e:
    print("Error:", e.response['Error']['Message'])
```

a. update "your_verified_email@example.com" with your email address
b. To save, **ctrl + O**, press **Enter** to confirm the filename, and then **ctrl + X** to exit.

Section 8: Introduction to UUIDs

A **universally unique identifier** (**UUID**) is a label used for information in computer systems. You may also hear them referred to as a **globally unique identifier** (**GUID**). UUIDs are, for practical purposes, unique, and are often used as unique keys for databases. Here are a few examples:

90d1e9fe-b59d-11ec-b909-0242ac120002

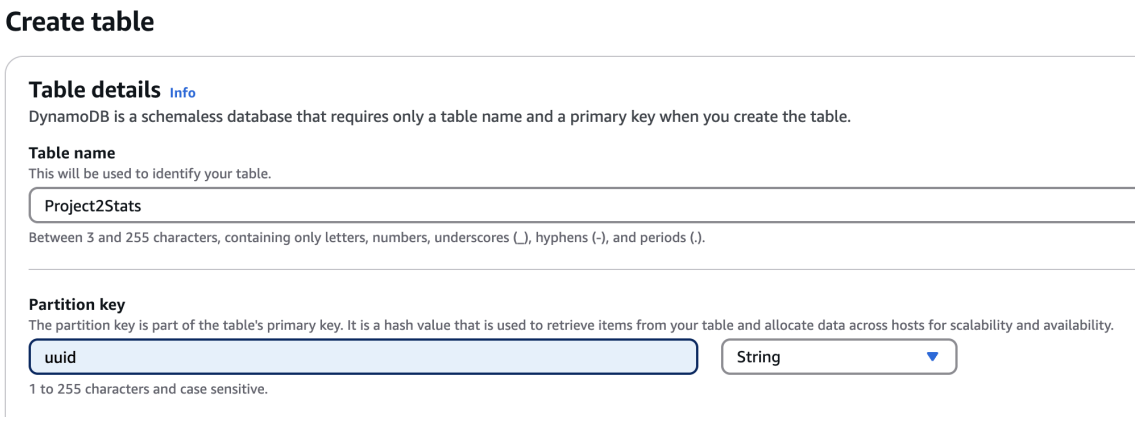
c42937c6-b59d-11ec-b909-0242ac120002

77cd8b61-11b9-4b83-9291-bf99df236676

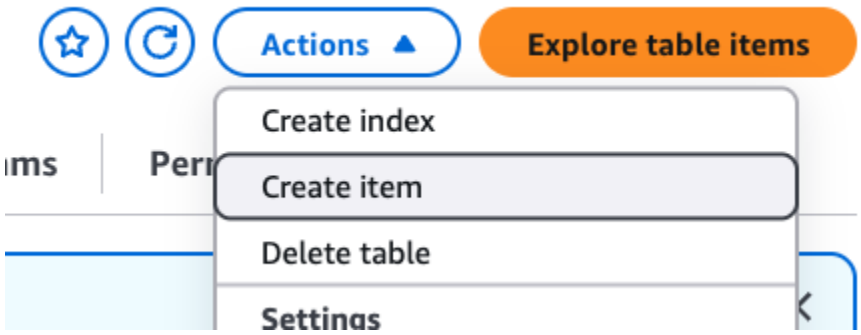
Take a look at the following website that will generate UUIDs:
https://www.uuidgenerator.net/

Section 9: Create a DynamoDB table

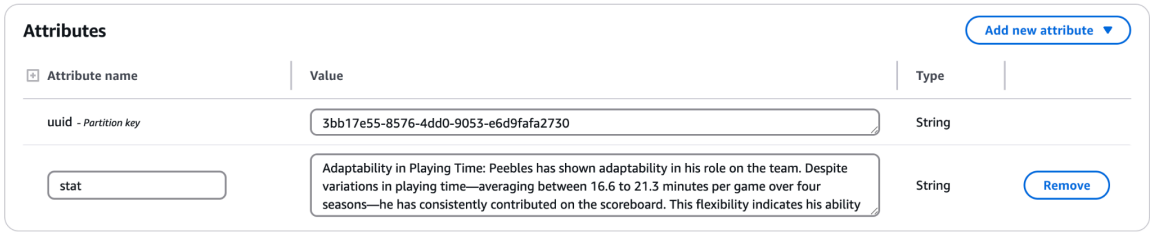
1. Go to DynamoDB
2. Create a new table called “Project2Stats” and call the Partition key “uuid” (which will be stored as a String)



3. Keep the rest of the default values and press “Create table”
4. Once created, select the “Project2Stats” table and select Actions → Create item

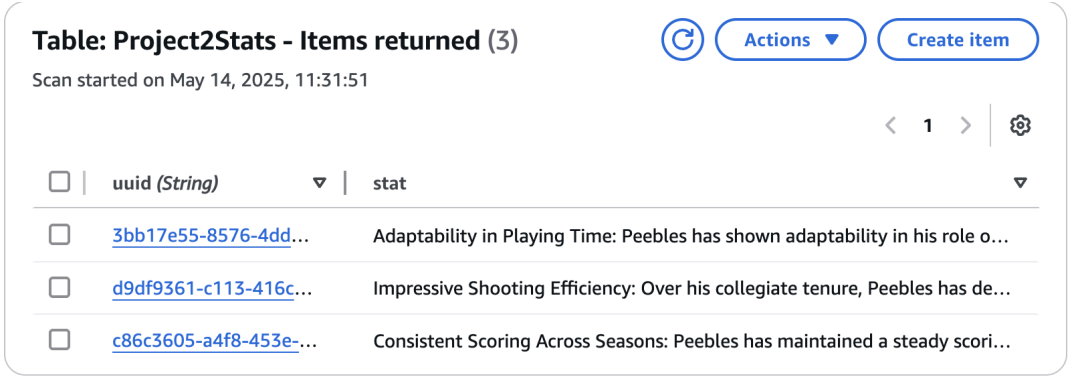


5. Go to https://www.uuidgenerator.net/ and copy a newly-generated UUID
6. paste the UUID in for the uuid Partition key
7. Click on “Add new attribute” → String. Call the attribute “stat”
8. Enter in a stat for the value.



1. Add at least three items to your table. Each item should have a newly-generated UUID for the partition key and an attribute called “stat”. The stat can be anything you want. Make sure that the attribute name is always “stat” (exactly spelling; exact capitalization).

   At the end, your “Project2Stats” table should look something like this:

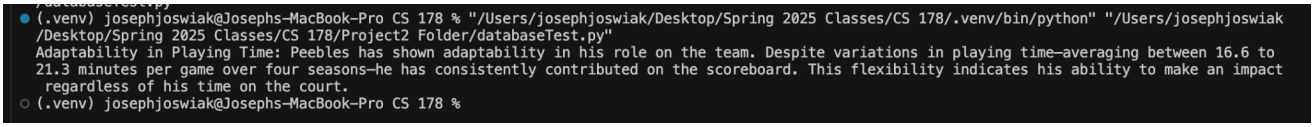


Section 10: Python for getting stats

1. Go back to VS Code
2. Create a new file in your Project2 Folder called **databaseTest.py**, you can create this one locally (rather than on your EC2 instance).
3. Enter the following code:

```
import boto3
import random

dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
table = dynamodb.Table('Project2Stats')

response = table.scan()
stat = random.choice(response['Items'])['stat']
print(stat)
```

4. Run it to verify that you get a stat printed. Run it several times to verify that you get a random joke selected from your database.

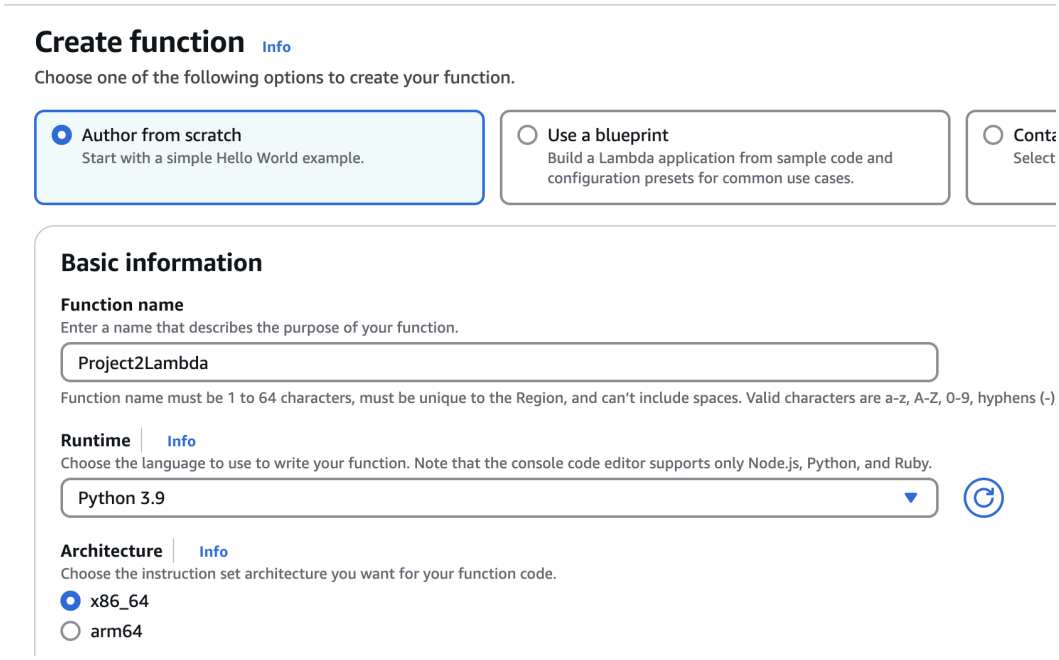


- If you get a **python not found** error, try using the run button instead of typing python3 databaseTest.py.

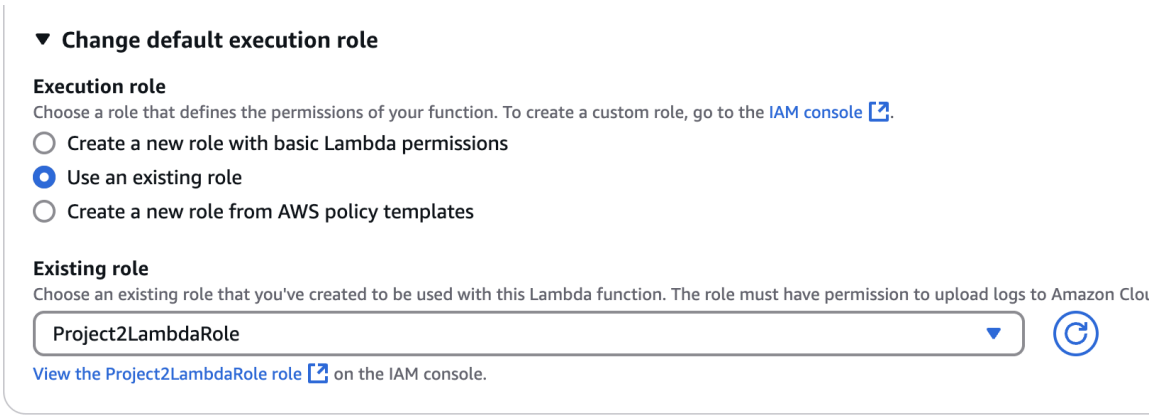
5. Copy this code into your **emailTest.py** file (back on your EC2 instance). Update the email code so that the body of the email includes your random stat.
    - Note that you may need to make sure the indentation of the code is consistent (e.g. using tabs vs. spaces) in emailTest.py.

Section 11: Lambda testing

1. Go to AWS Lambda.
2. Create a new Lambda function. Call the function “Project2Lambda”. Change the runtime to Python 3.9.



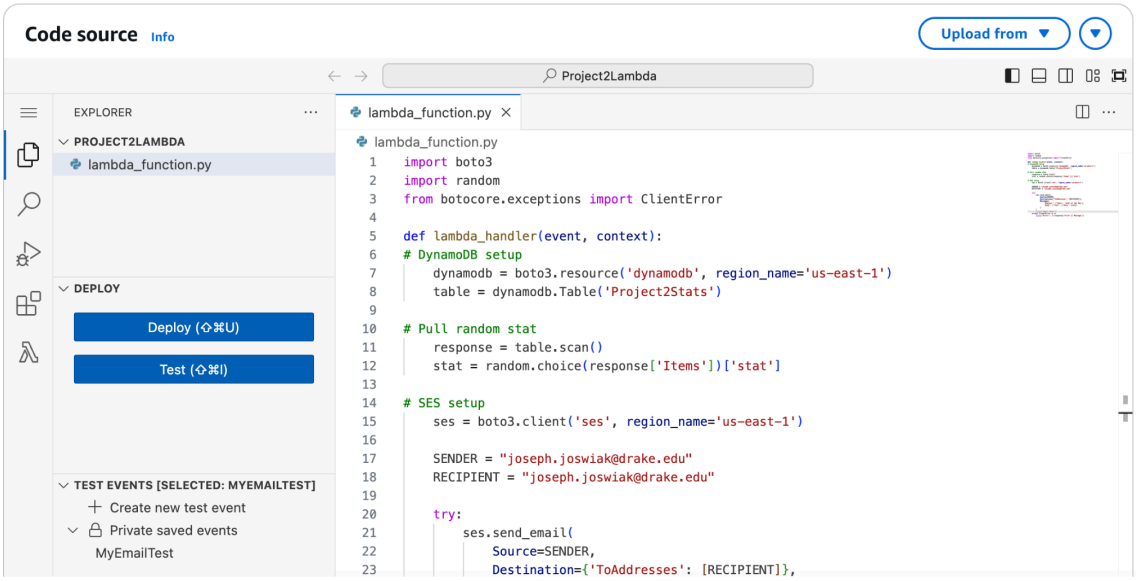
Also scroll down and change the default execution role to “Project2Lambda” that you created in Section 4.



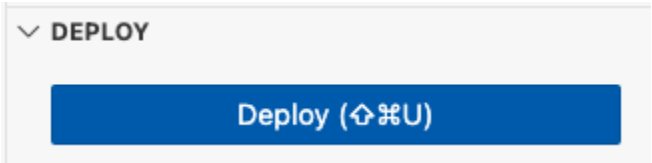
Then press “Create function”

3. Remove all of the code in the default **lambda_function.py** file.

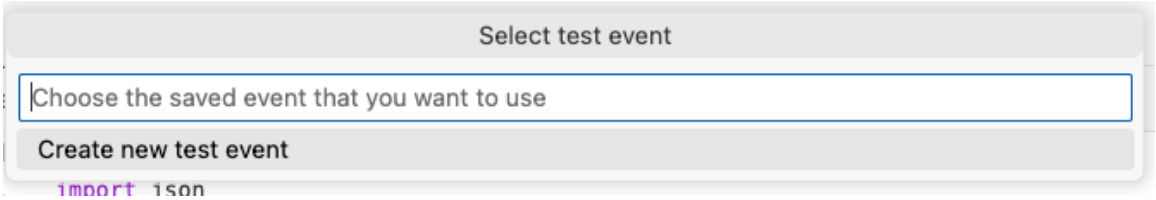
4. Copy the code from **emailTest.py** file to inside of your lambda function, update the code to pull a random joke from your dynamodb table (the code from databaseTesting.py should come in handy here).



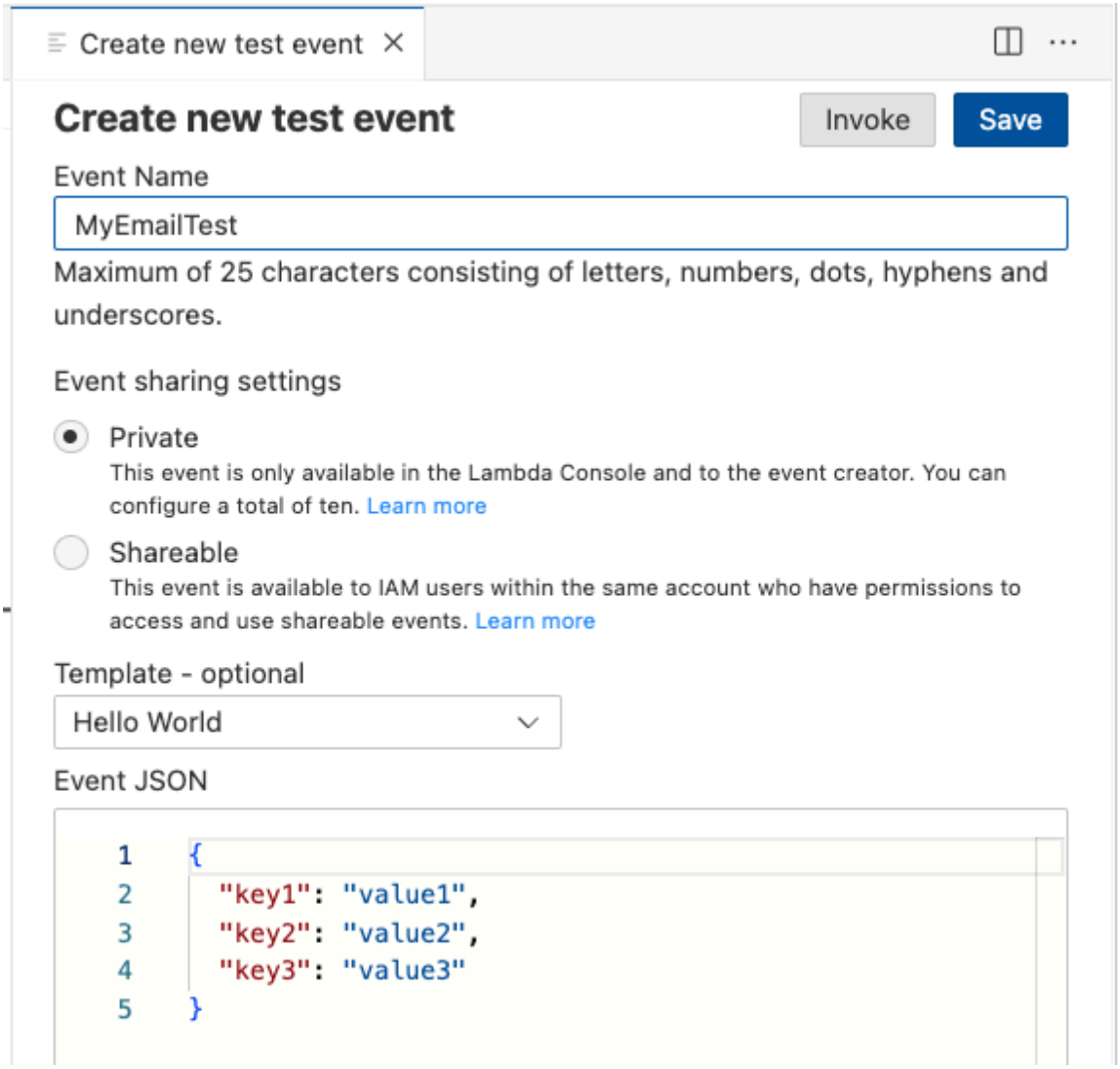
5. Save. Then deploy your code



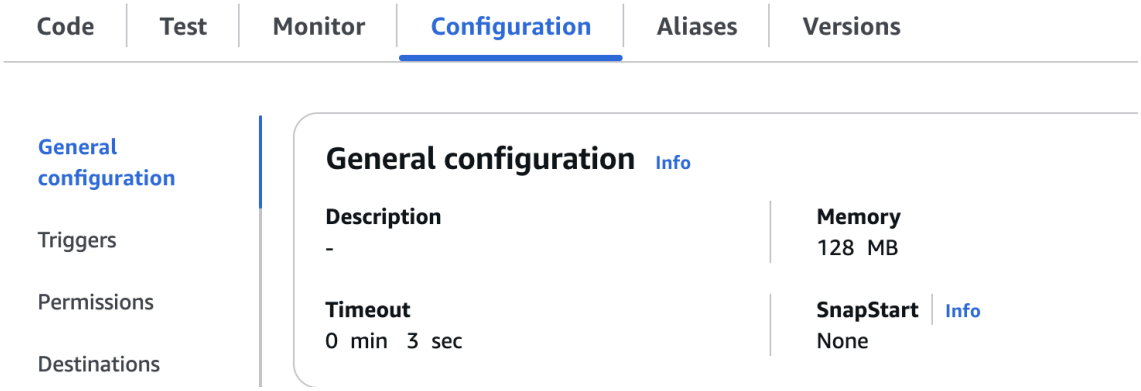
6. Click on “Test”. This will prompt you to configure a test event. Name the event



7. “MyEmailTest”, keep the default Event json and click on “Save”



8. Go back to the lambda console and click on info about Project2Lambda, Scroll down and select “Configuration”



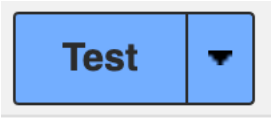
Then, click “Edit”

and change the timeout to 20 seconds:

and click “Save”

9. Make sure any changes are deployed

10. Click on “Test”



11.
12. This will generate a test run of the code in the lambda function. If all goes well, you should receive an email as a result. The logs will look like this:

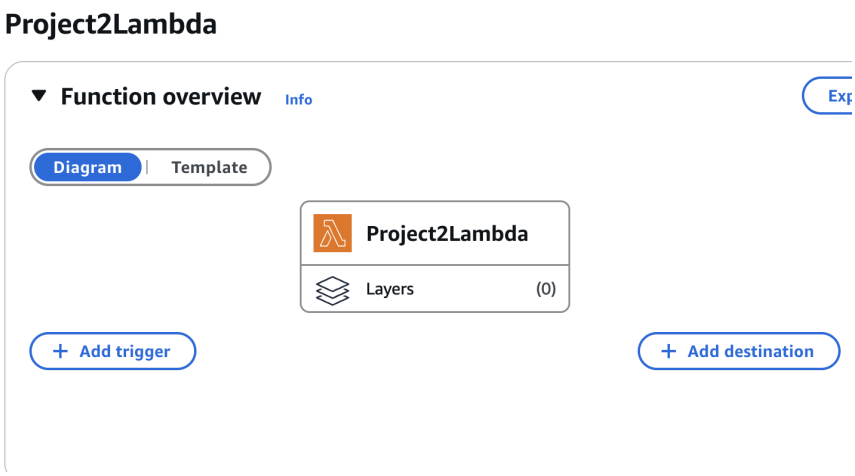
```
Status: Succeeded
Test Event Name: MyEmailTest

Response:
null
```

13.

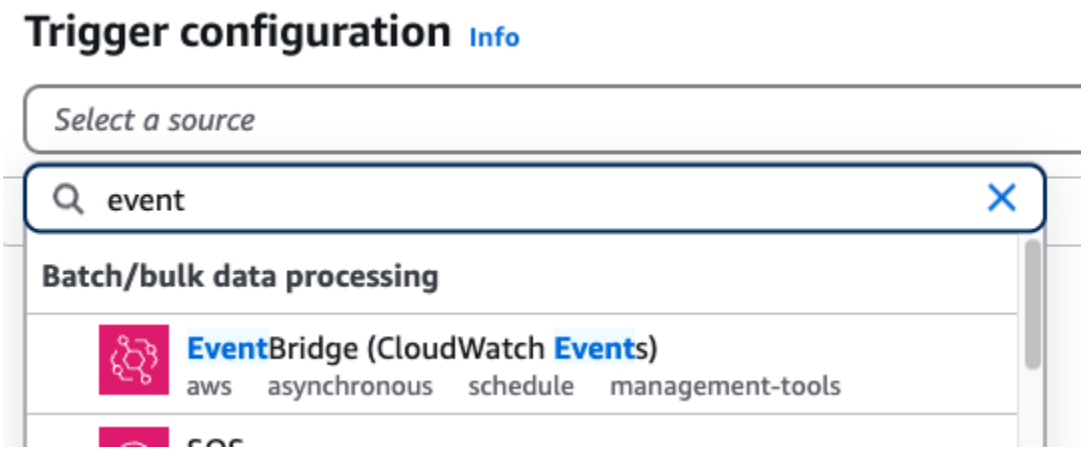
Section 12: Event Trigger

Next, Let’s add a trigger so this lambda function executes at certain times during the day. At first, for testing, we’ll have it fire every minute. After we’re sure that it works, we’ll have it fire only at 7:00am every morning.

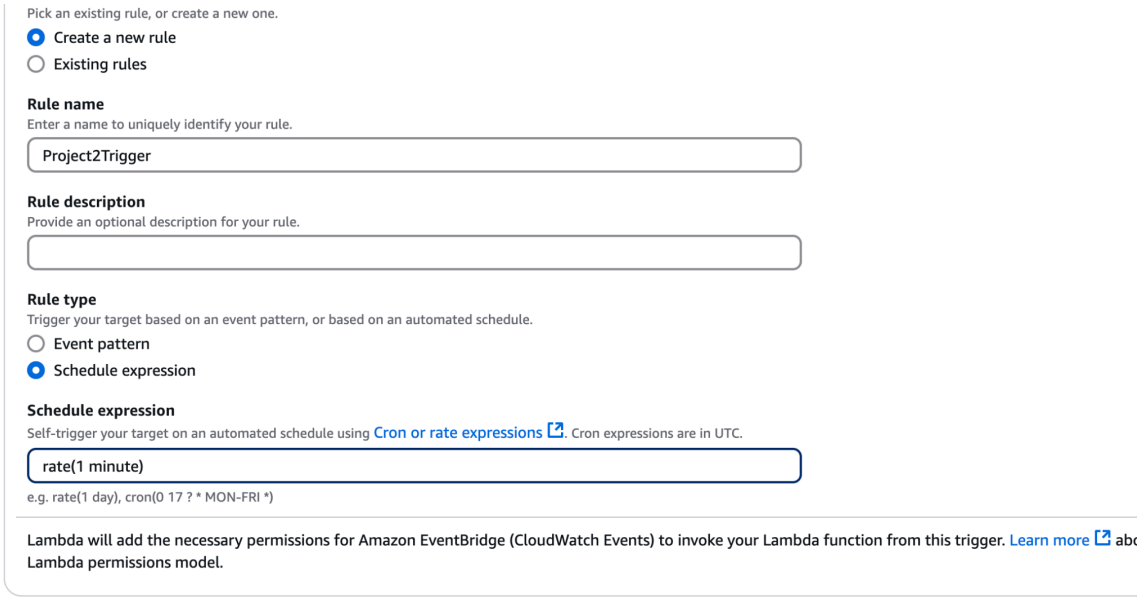
1. In you Lambda Function, click on “Add Trigger”



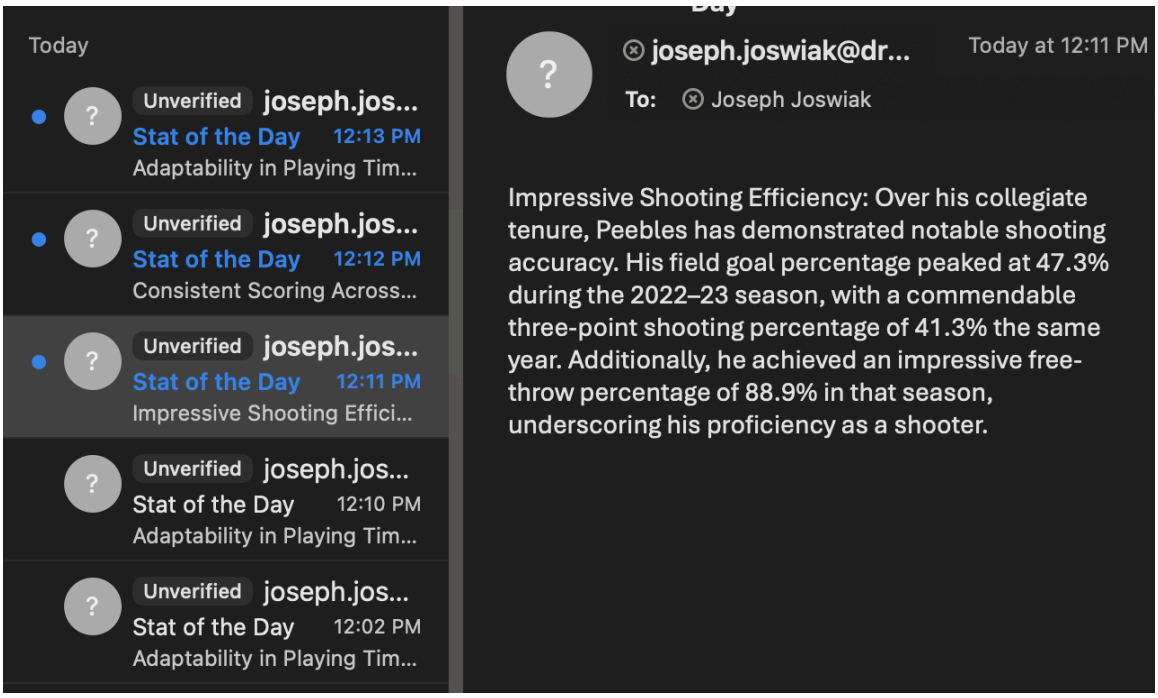
2. Select EventBridge



- Click on “Create a new rule”
- Call the Rule name “Project2Trigger”
- Under the “Schedule expression” enter rate(1 minute)
    - This will make the event trigger every minute – great for testing.
- Click “Add” (see below)



3. Wait a minute and check your email. You should be receiving new joke emails every minute.



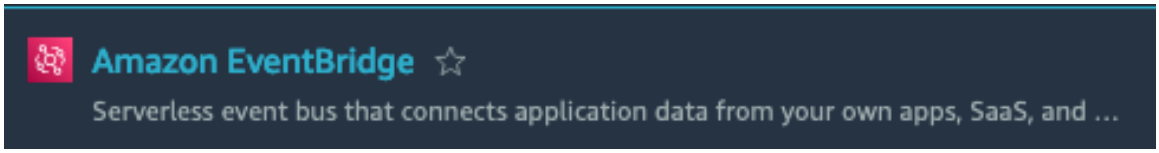
- Note: You're seeing “Unverified” because Amazon SES has not authorized your email domain (drake.edu) to send on its behalf. Even though your individual

email (joseph.joswiak@drake.edu) is verified, the domain itself (drake.edu) is not, which is why recipients see this warning.

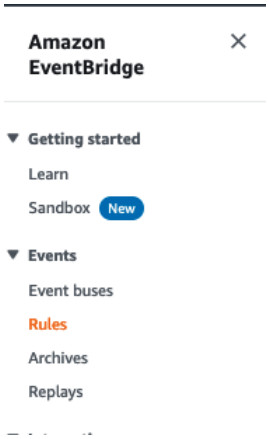
- Ignore the warning (Unverified). This is totally fine for labs, testing, and demos.

Section 13: Event Trigger

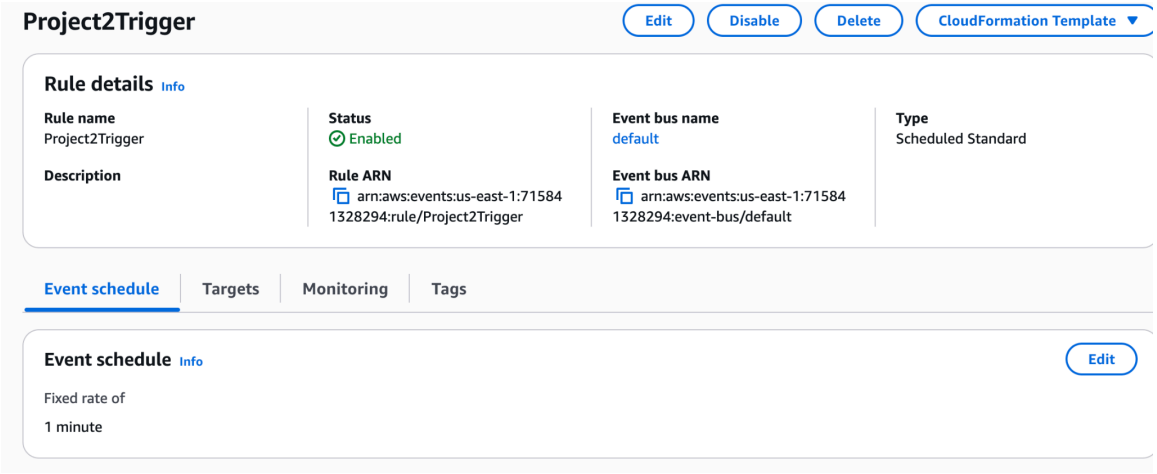
1. Let's change the trigger so it happens at 7:00am instead of every minute.
2. In AWS search for "EventBridge"



3. On the left hand side, click on "Rules"



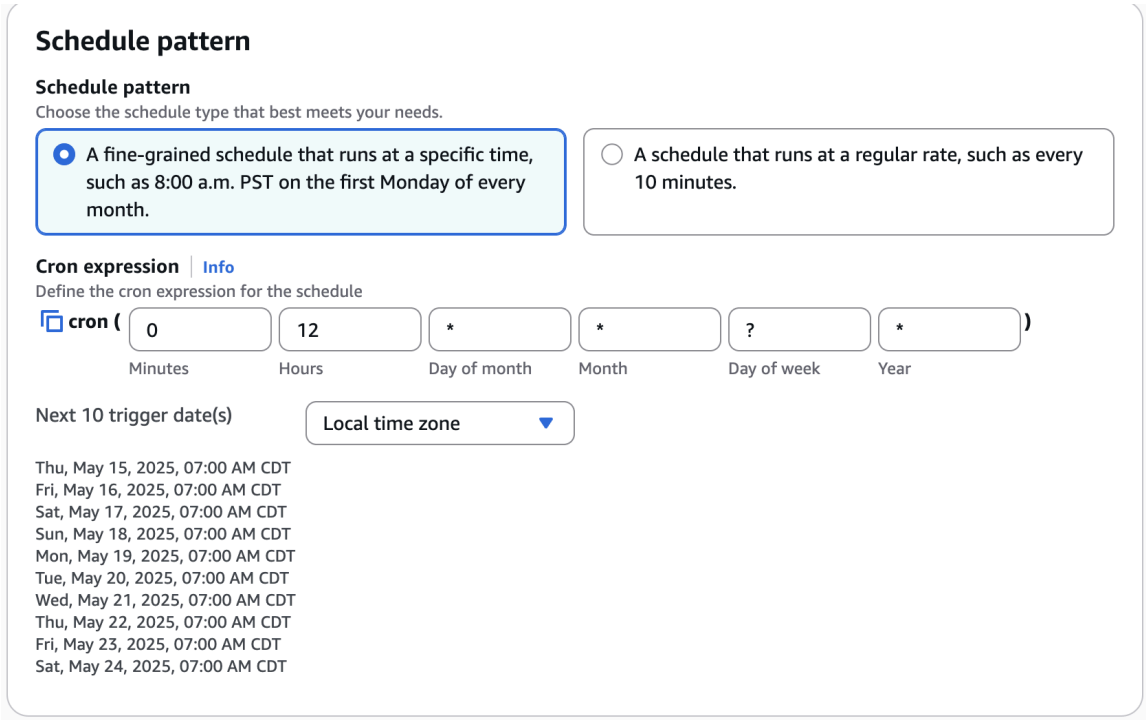
4. You should see your Project2Trigger here. Click on it.



5. Under the "Event schedule", click on "Edit"

6. **Cron** is a utility that can specify a specific time occurrence during a day (e.g. 7:00am every weekday).

7. Change the Schedule pattern to look like this:



8. Click Next. And Next. And Next And finally, “Update Rule”
   With the rule update, the trigger should only fire at 7:00am every morning.