Assignments

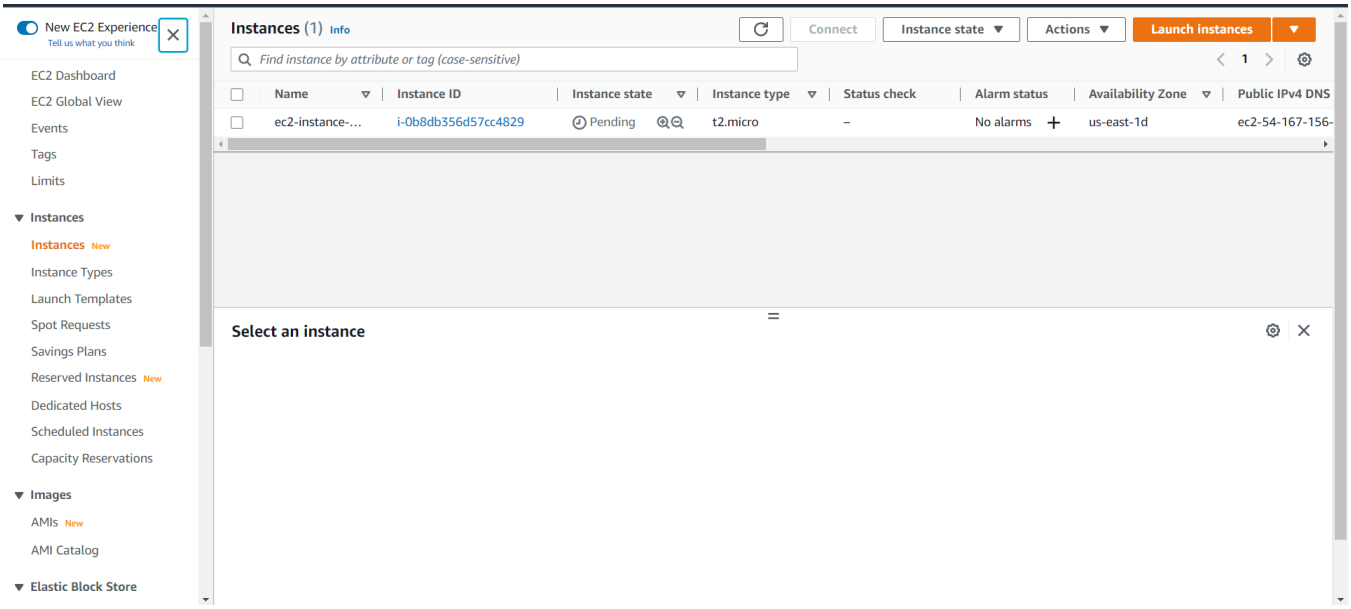
Assignment: Create a web server and an Amazon RDS DB instance.

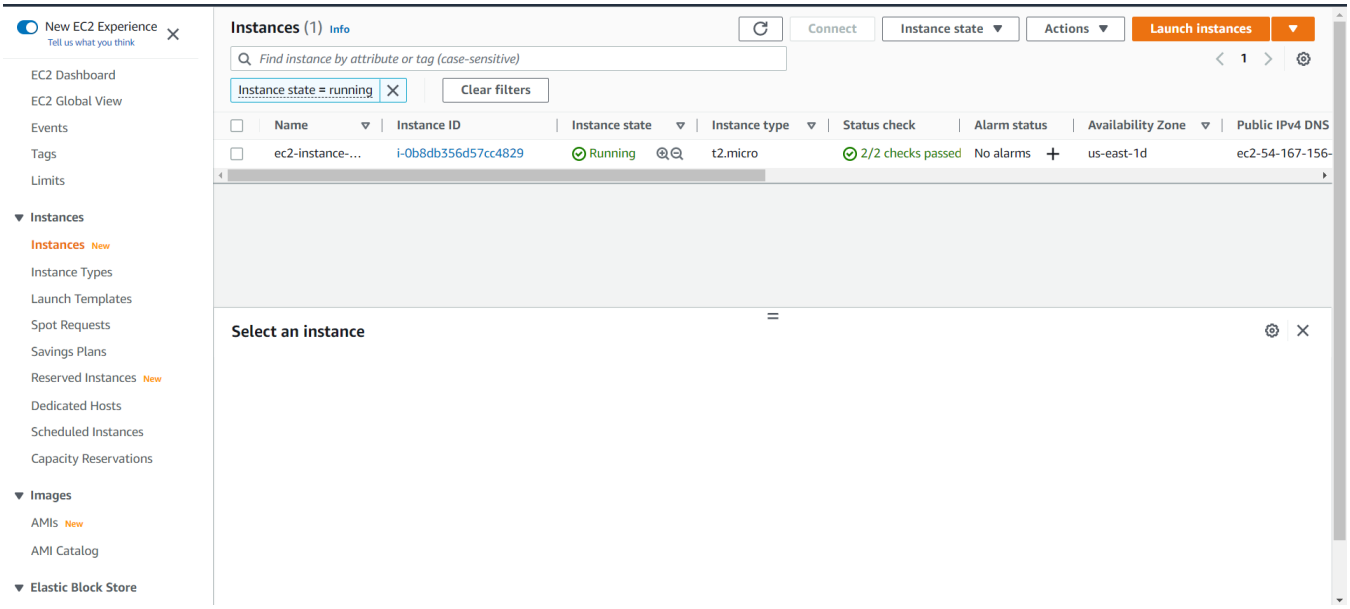
Install an Apache web server with PHP and create a MySQL database. The web server runs on an Amazon EC2 instance using Amazon Linux, and the MySQL database is a MySQL DB instance. Both the Amazon EC2 instance and the DB instance run in a virtual private cloud (VPC) based on the Amazon VPC service.

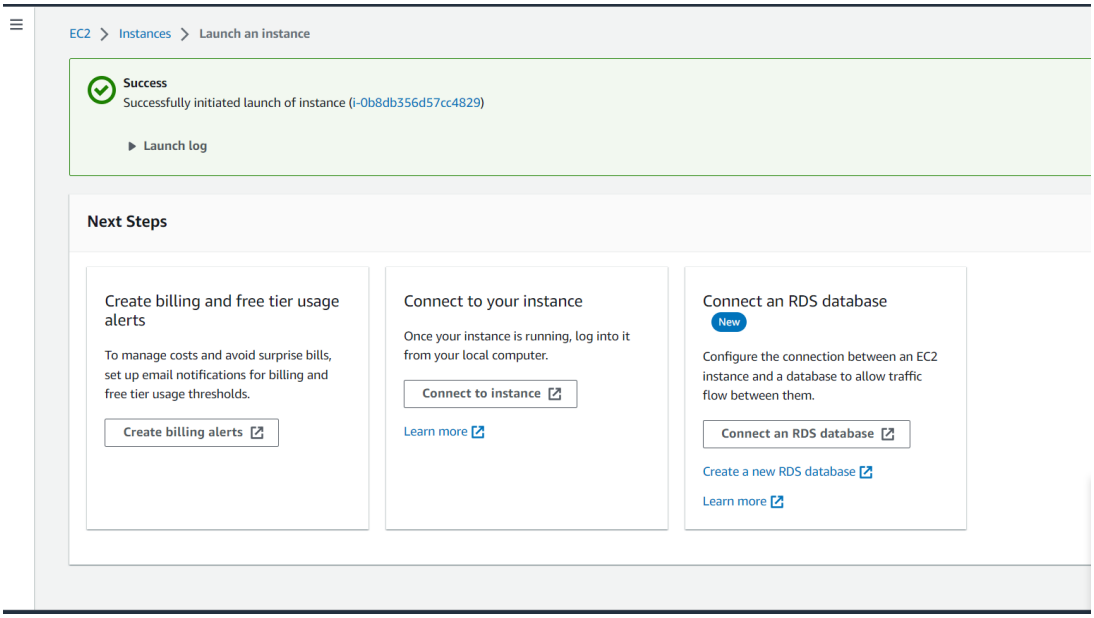
1. Launch an EC2 instance
2. Create a DB instance
3. Install a web server on your EC2 instance

1. Create an Amazon EC2 instance in the public subnet of your VPC:

- Sign into the AWS Management Console and open the Amazon EC2 console.
- In the upper-right corner of the AWS Management Console, choose the AWS Region where you want to create the EC2 instance.
- Choose EC2 Dashboard, and then choose Launch instance, as shown following
- Make sure you have opted into the new launch experience
- Under Name and tags, for Name, enter ec2-instance-web-server.
- Under Application and OS Images (Amazon Machine Image), choose Amazon Linux, and then choose the Amazon Linux 2 AMI. Keep the defaults for the other choices.
- Under Instance type, choose t2.micro.
- Under Key pair (login), choose a Key pair name to use an existing key pair. To create a new key pair for the Amazon EC2 instance, choose Create new key pair and then use the Create key pair window to create it.
- Under Network settings, set these values and keep the other values as their defaults:
  For Allow SSH traffic from, choose the source of SSH connections to the EC2 instance.
  Turn on Allow HTTPs traffic from the internet.
  Turn on Allow HTTP traffic from the internet.
- Leave the default values for the remaining sections.
- Review a summary of your instance configuration in the Summary panel, and when you're ready, choose Launch instance.
- On the Launch Status page, shown following, note the identifier for your new EC2 instance.
- Choose View all instances to find your instance.
- Wait until Instance state for your instance is Running before continuing.


New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Capacity Reservations
Images
AMIs New
AMI Catalog
Elastic Block Store
Instances (1) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
1
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
ec2-instance-...
i-0b8db356d57cc4829
Pending
t2.micro
–
No alarms
us-east-1d
ec2-54-167-156-
Select an instance


New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Capacity Reservations
Images
AMIs New
AMI Catalog
Elastic Block Store
Instances (1) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
1
Instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
ec2-instance-...
i-0b8db356d57cc4829
Running
t2.micro
2/2 checks passed
No alarms
us-east-1d
ec2-54-167-156-
Select an instance

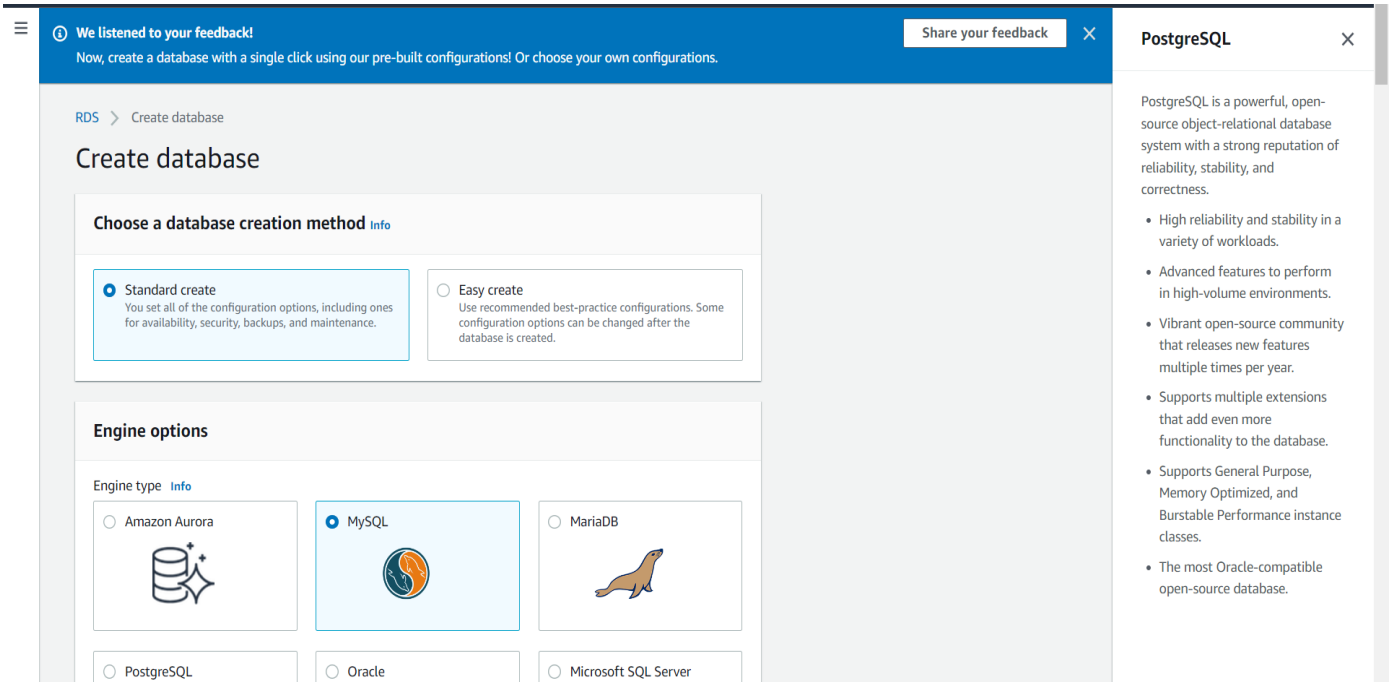


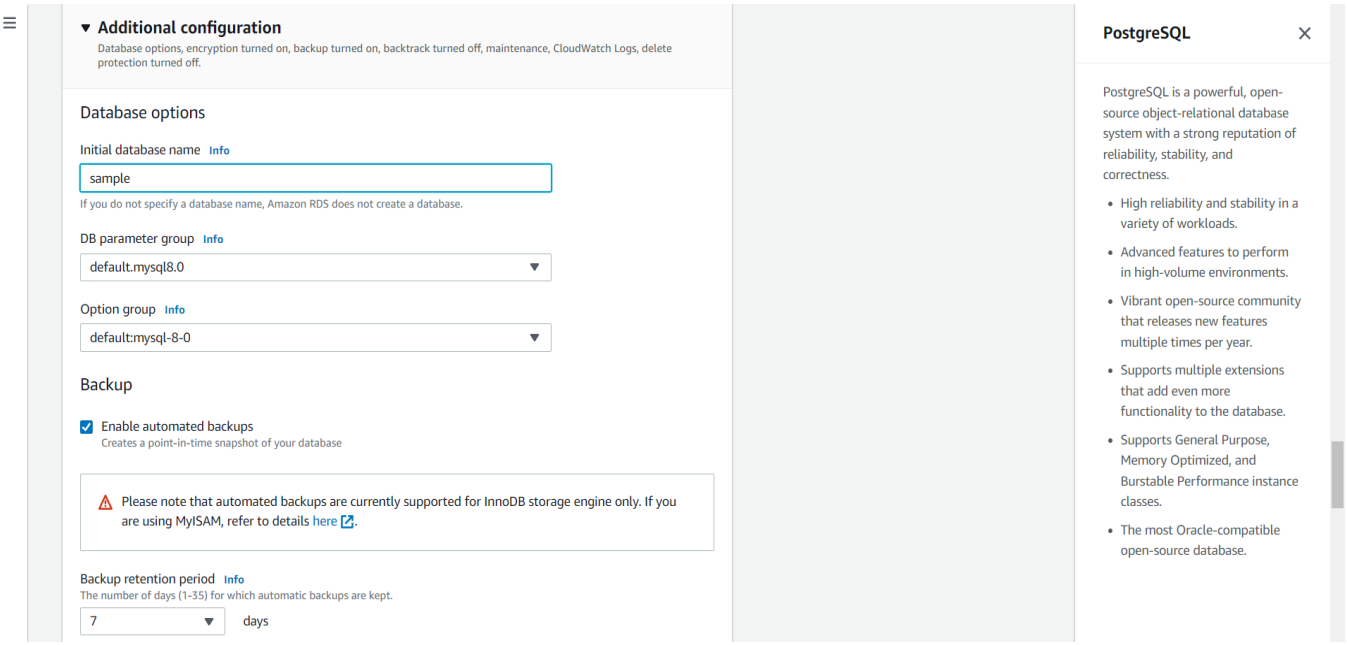
2. Create an Amazon RDS for MySQL DB instance that maintains the data used by a web application.

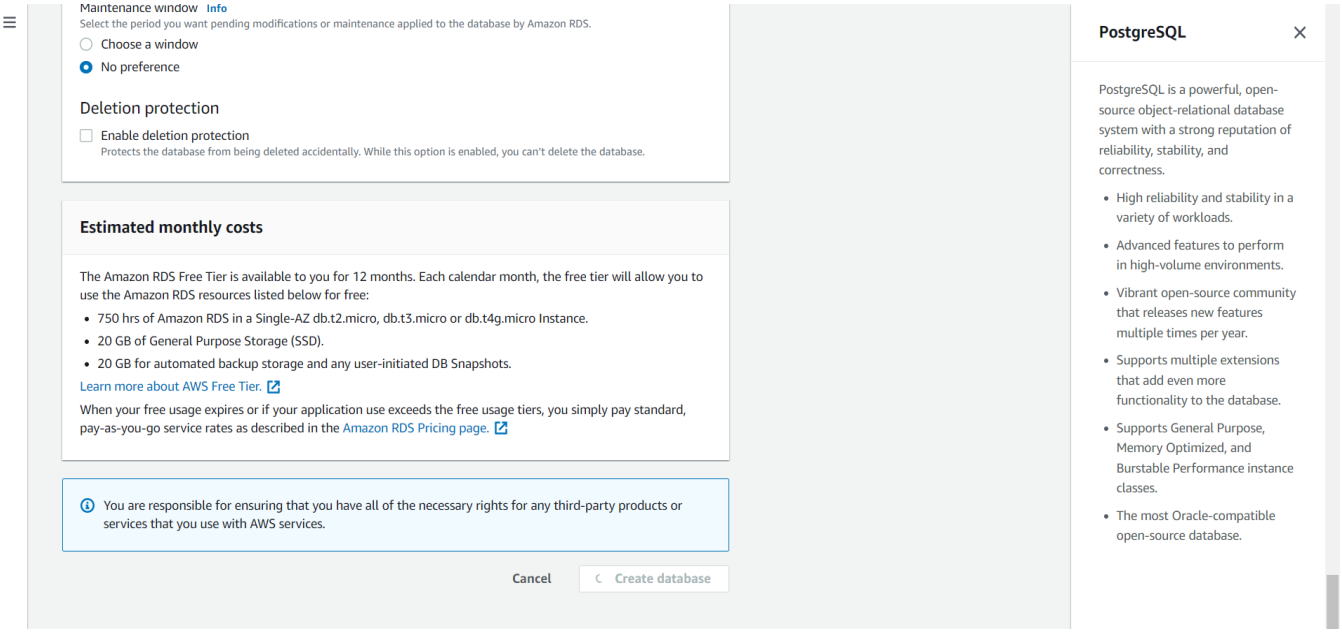
- In the upper-right corner of the AWS Management Console, check the AWS Region. It should be the same as the one where you created your EC2 instance
- In the navigation pane, choose Databases.
- Choose Create database.
- On the Create database page, shown following, make sure that the Standard create option is chosen, and then choose MySQL.
- In the Templates section, choose Free tier.
- In the Availability and durability section, keep the defaults.
- In the Settings section, set these values:
  DB instance identifier – db-instance
  Master username – admi
  Auto generate a password – Leave the option turned off.
  Master password – **********
  Confirm password – **********
- In the Instance configuration section, set these values: Burstable classes (includes t classes)db.t3.micro
- In the Storage section, keep the defaults.
- In the Connectivity section, set these values and keep the other values as their defaults:
  For Compute resource, choose Connect to an EC2 compute resource.
  For EC2 instance, choose the EC2 instance you created previously, such as ec2-instance-web-server.

- In the Database authentication section, make sure Password authentication is selected.
- Open the Additional configuration section, and enter sample for Initial database name. Keep the default settings for the other options.
- To create your MySQL DB instance, choose Create database.

  Your new DB instance appears in the Databases list with the status Creating.
- Wait for the Status of your new DB instance to show as Available. Then choose the DB instance name to show its details.
- In the Connectivity & security section, view the Endpoint and Port of the DB instance.







3. Install a web server on your EC2 instance
   Install an Apache web server with PHP and MariaDB

- Connect to the EC2 instance that you created earlier by following the steps in Connect to your Linux instance.
- Get the latest bug fixes and security updates by updating the software on your EC2 instance. To do this, use the following command.
  sudo yum update -y

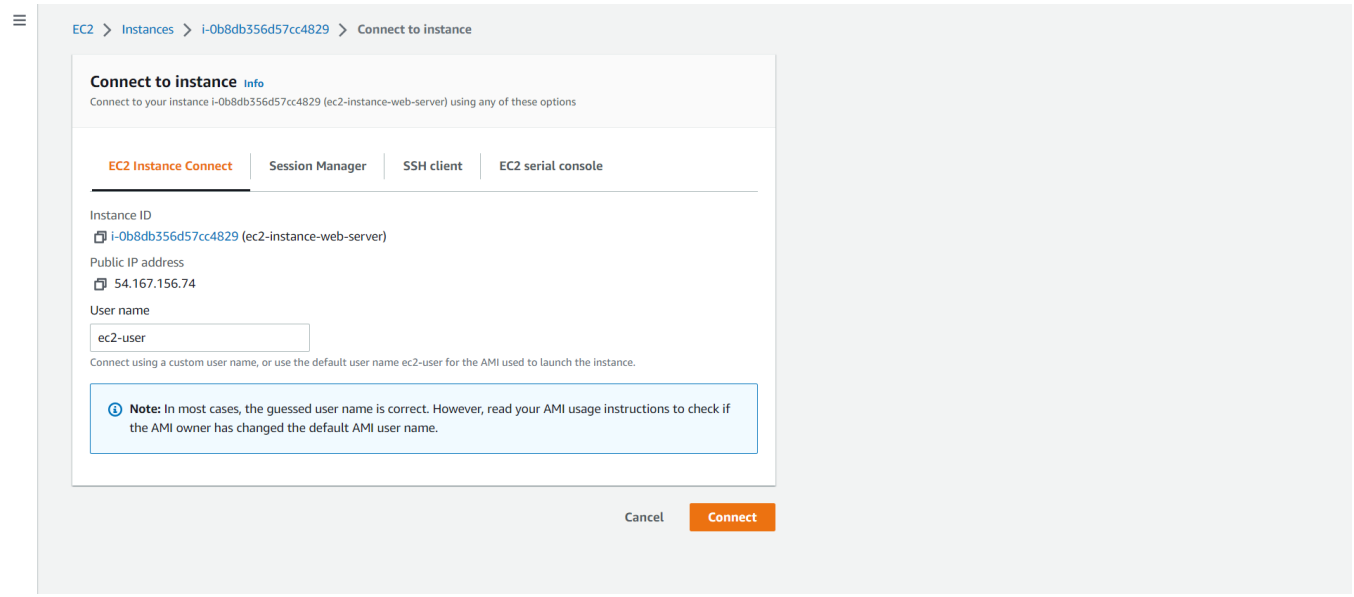
- After the updates complete, install the PHP software using the amazon-linux-extras install command. This command installs multiple software packages and related dependencies at the same time.
  sudo amazon-linux-extras install php8.0 mariadb10.5
- Install the Apache web server.
  sudo yum install -y httpd
- Start the web server with the command shown following.
  sudo systemctl start httpd
- Configure the web server to start with each system boot using the systemctl command.
  sudo systemctl enable httpd

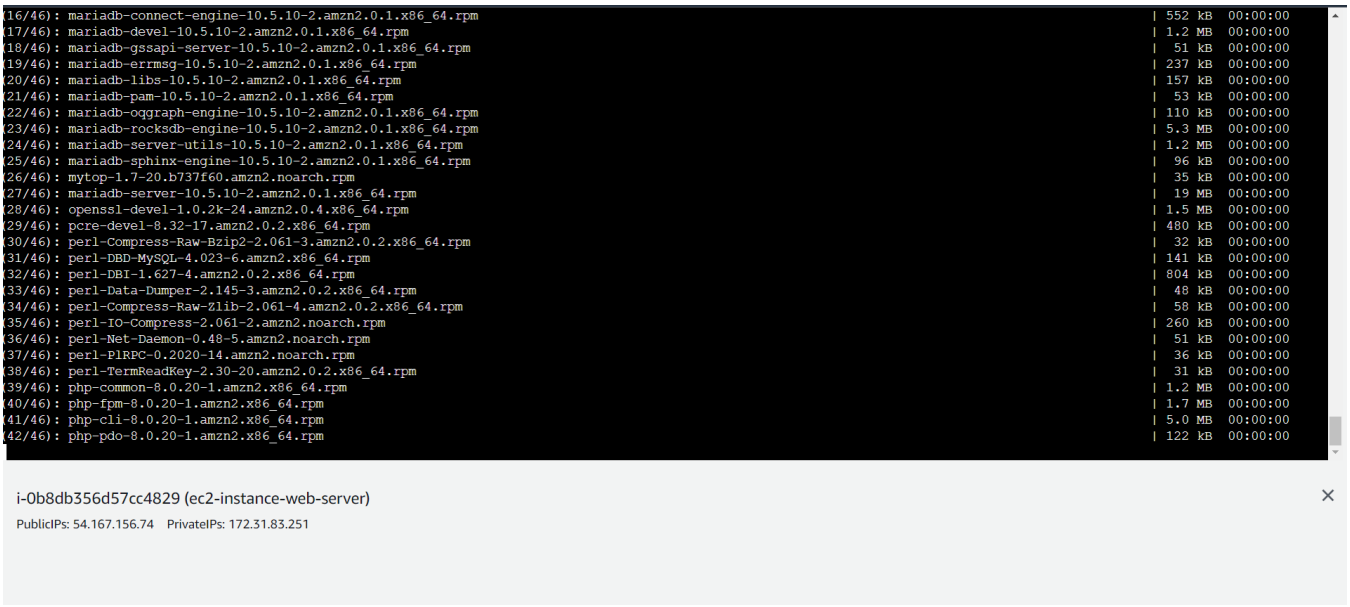
To set file permissions for the Apache web server:

- Add the ec2-user user to the apache group.
  sudo usermod -a -G apache ec2-user
- Log out to refresh your permissions and include the new apache group.
  Exit
- Change the group ownership of the /var/www directory and its contents to the apache group.
  sudo chown -R ec2-user:apache /var/www
- Recursively change the permissions for files in the /var/www directory and its subdirectories to add group write permissions.
  find /var/www -type f -exec sudo chmod 0664 {} \;

Connect your Apache web server to your DB instance:

- While still connected to your EC2 instance, change the directory to /var/www and create a new subdirectory named inc.
- Create a new file in the inc directory named dbinfo.inc, and then edit the file by calling nano (or the editor of your choice).
- Add the following contents to the dbinfo.inc file. Here, *db_instance_endpoint* is your DB instance endpoint, without the port, and *master password* is the master password for your DB instance.
- Save and close the dbinfo.inc file.
- Change the directory to /var/www/html.
- Create a new file in the html directory named SamplePage.php, and then edit the file by calling nano (or the editor of your choice).
- Add the following contents to the SamplePage.php file:
- Save and close the SamplePage.php file.
- Verify that your web server successfully connects to your DB instance by opening a web browser and browsing to http://*EC2 instance endpoint*/SamplePage.php


EC2 > Instances > i-0b8db356d57cc4829 > Connect to instance
Connect to instance Info
Connect to your instance i-0b8db356d57cc4829 (ec2-instance-web-server) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-0b8db356d57cc4829 (ec2-instance-web-server)
Public IP address
54.167.156.74
User name
ec2-user
Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.
Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.
Cancel
Connect


(16/46): mariadb-connect-engine-10.5.10-2.amzn2.0.1.x86_64.rpm | 552 kB 00:00:00
(17/46): mariadb-devel-10.5.10-2.amzn2.0.1.x86_64.rpm | 1.2 MB 00:00:00
(18/46): mariadb-gssapi-server-10.5.10-2.amzn2.0.1.x86_64.rpm | 51 kB 00:00:00
(19/46): mariadb-errmsg-10.5.10-2.amzn2.0.1.x86_64.rpm | 237 kB 00:00:00
(20/46): mariadb-libs-10.5.10-2.amzn2.0.1.x86_64.rpm | 157 kB 00:00:00
(21/46): mariadb-pam-10.5.10-2.amzn2.0.1.x86_64.rpm | 53 kB 00:00:00
(22/46): mariadb-oqgraph-engine-10.5.10-2.amzn2.0.1.x86_64.rpm | 110 kB 00:00:00
(23/46): mariadb-rocksdb-engine-10.5.10-2.amzn2.0.1.x86_64.rpm | 5.3 MB 00:00:00
(24/46): mariadb-server-utils-10.5.10-2.amzn2.0.1.x86_64.rpm | 1.2 MB 00:00:00
(25/46): mariadb-sphinx-engine-10.5.10-2.amzn2.0.1.x86_64.rpm | 96 kB 00:00:00
(26/46): mytop-1.7-20.b737f60.amzn2.noarch.rpm | 35 kB 00:00:00
(27/46): mariadb-server-10.5.10-2.amzn2.0.1.x86_64.rpm | 19 MB 00:00:00
(28/46): openssl-devel-1.0.2k-24.amzn2.0.4.x86_64.rpm | 1.5 MB 00:00:00
(29/46): pcre-devel-8.32-17.amzn2.0.2.x86_64.rpm | 480 kB 00:00:00
(30/46): perl-Compress-Raw-Bzip2-2.061-3.amzn2.0.2.x86_64.rpm | 32 kB 00:00:00
(31/46): perl-DBD-MySQL-4.023-6.amzn2.x86_64.rpm | 141 kB 00:00:00
(32/46): perl-DBI-1.627-4.amzn2.0.2.x86_64.rpm | 804 kB 00:00:00
(33/46): perl-Data-Dumper-2.145-3.amzn2.0.2.x86_64.rpm | 48 kB 00:00:00
(34/46): perl-Compress-Raw-Zlib-2.061-4.amzn2.0.2.x86_64.rpm | 58 kB 00:00:00
(35/46): perl-IO-Compress-2.061-2.amzn2.noarch.rpm | 260 kB 00:00:00
(36/46): perl-Net-Daemon-0.48-5.amzn2.noarch.rpm | 51 kB 00:00:00
(37/46): perl-PlRPC-0.2020-14.amzn2.noarch.rpm | 36 kB 00:00:00
(38/46): perl-TermReadKey-2.30-20.amzn2.0.2.x86_64.rpm | 31 kB 00:00:00
(39/46): php-common-8.0.20-1.amzn2.x86_64.rpm | 1.2 MB 00:00:00
(40/46): php-fpm-8.0.20-1.amzn2.x86_64.rpm | 1.7 MB 00:00:00
(41/46): php-cli-8.0.20-1.amzn2.x86_64.rpm | 5.0 MB 00:00:00
(42/46): php-pdo-8.0.20-1.amzn2.x86_64.rpm | 122 kB 00:00:00
i-0b8db356d57cc4829 (ec2-instance-web-server)
PublicIPs: 54.167.156.74 PrivateIPs: 172.31.83.251

```
  Installing : mod_http2-1.15.19-1.amzn2.0.1.x86_64                                                                  8/9
  Installing : httpd-2.4.54-1.amzn2.x86_64                                                                           9/9
  Verifying  : apr-util-1.6.1-5.amzn2.0.2.x86_64                                                                     1/9
  Verifying  : apr-util-bdb-1.6.1-5.amzn2.0.2.x86_64                                                                 2/9
  Verifying  : httpd-tools-2.4.54-1.amzn2.x86_64                                                                     3/9
  Verifying  : mod_http2-1.15.19-1.amzn2.0.1.x86_64                                                                  4/9
  Verifying  : httpd-2.4.54-1.amzn2.x86_64                                                                           5/9
  Verifying  : mailcap-2.1.41-2.amzn2.noarch                                                                         6/9
  Verifying  : generic-logos-httpd-18.0.0-4.amzn2.noarch                                                             7/9
  Verifying  : httpd-filesystem-2.4.54-1.amzn2.noarch                                                                8/9
  Verifying  : apr-1.7.0-9.amzn2.x86_64                                                                              9/9

Installed:
  httpd.x86_64 0:2.4.54-1.amzn2

Dependency Installed:
  apr.x86_64 0:1.7.0-9.amzn2                 apr-util.x86_64 0:1.6.1-5.amzn2.0.2   apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2   generic-logos-httpd.noarch 0:18.0.0-4.amzn2
  httpd-filesystem.noarch 0:2.4.54-1.amzn2   httpd-tools.x86_64 0:2.4.54-1.amzn2   mailcap.noarch 0:2.1.41-2.amzn2           mod_http2.x86_64 0:1.15.19-1.amzn2.0.1

Complete!
[ec2-user@ip-172-31-83-251 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-83-251 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-83-251 ~]$ sudo systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-83-251 ~]$ sudo usermod -a -G apache ec2-user
[ec2-user@ip-172-31-83-251 ~]$ exit
logout
```

i-0b8db356d57cc4829 (ec2-instance-web-server)

PublicIPs: 54.167.156.74 PrivateIPs: 172.31.83.251

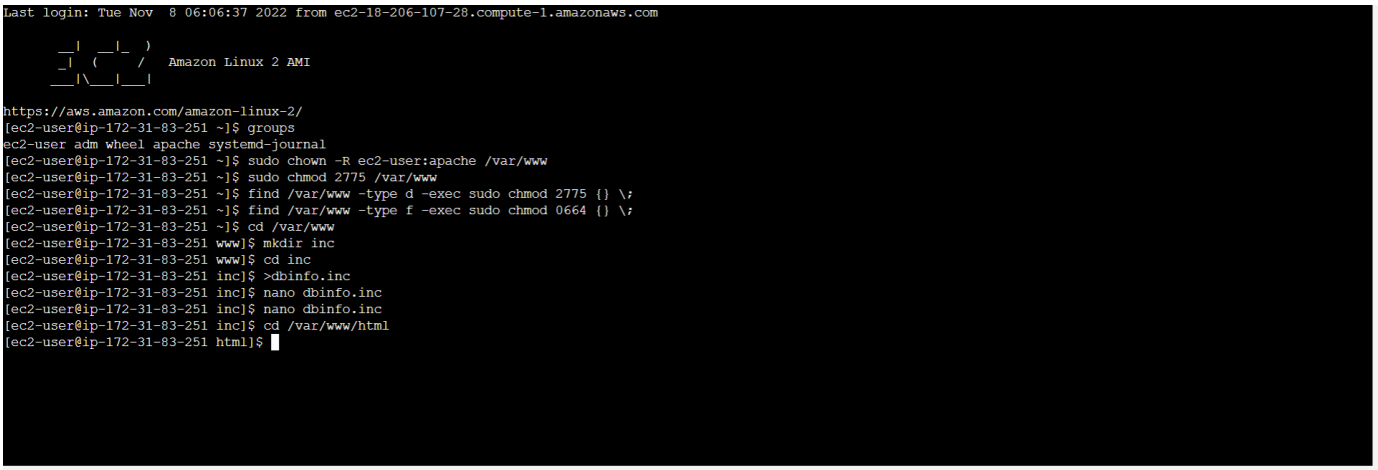
```
Last login: Tue Nov  8 06:06:37 2022 from ec2-18-206-107-28.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-83-251 ~]$ groups
ec2-user adm wheel apache systemd-journal
[ec2-user@ip-172-31-83-251 ~]$ sudo chown -R ec2-user:apache /var/www
[ec2-user@ip-172-31-83-251 ~]$
```

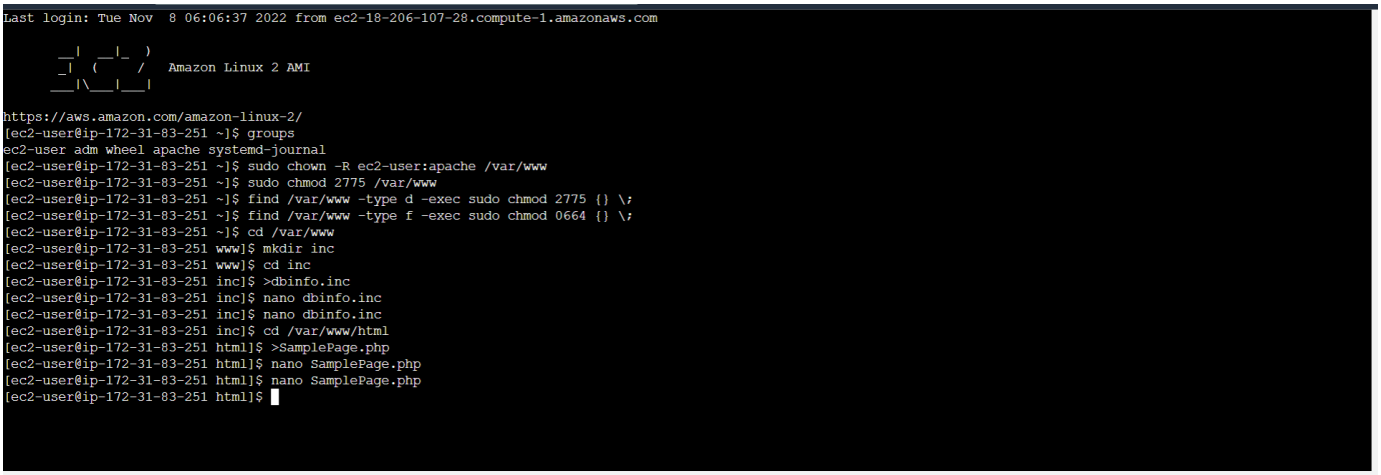
i-0b8db356d57cc4829 (ec2-instance-web-server)

PublicIPs: 54.167.156.74 PrivateIPs: 172.31.83.251



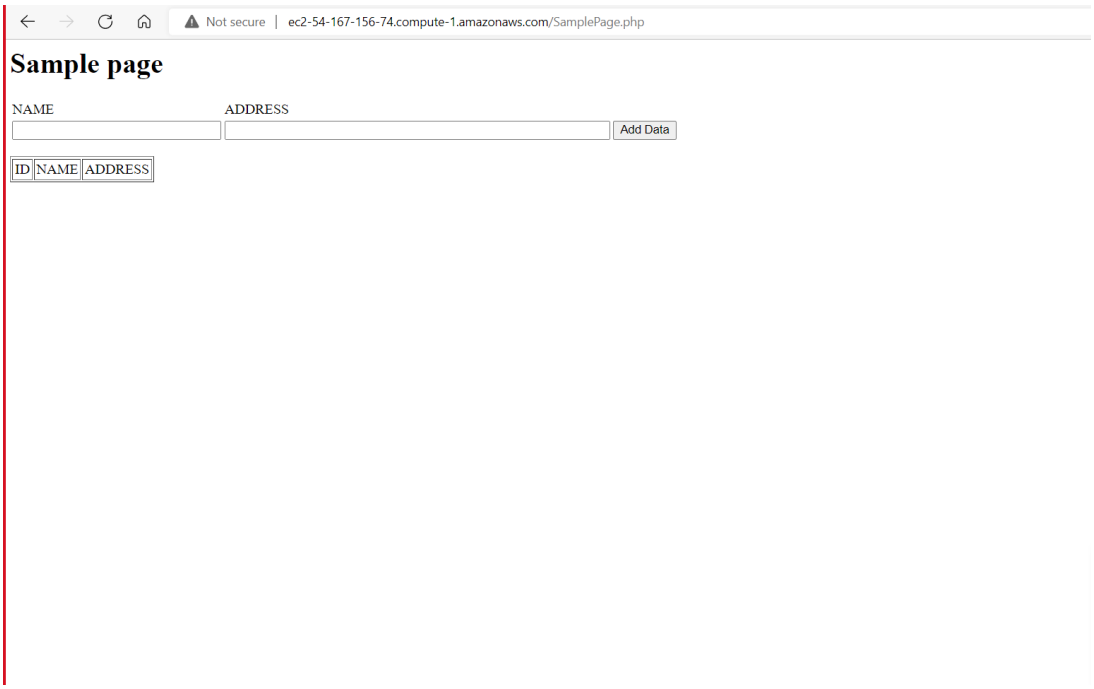
i-0b8db356d57cc4829 (ec2-instance-web-server)

PublicIPs: 54.167.156.74 PrivateIPs: 172.31.83.251



i-0b8db356d57cc4829 (ec2-instance-web-server)

PublicIPs: 54.167.156.74 PrivateIPs: 172.31.83.251

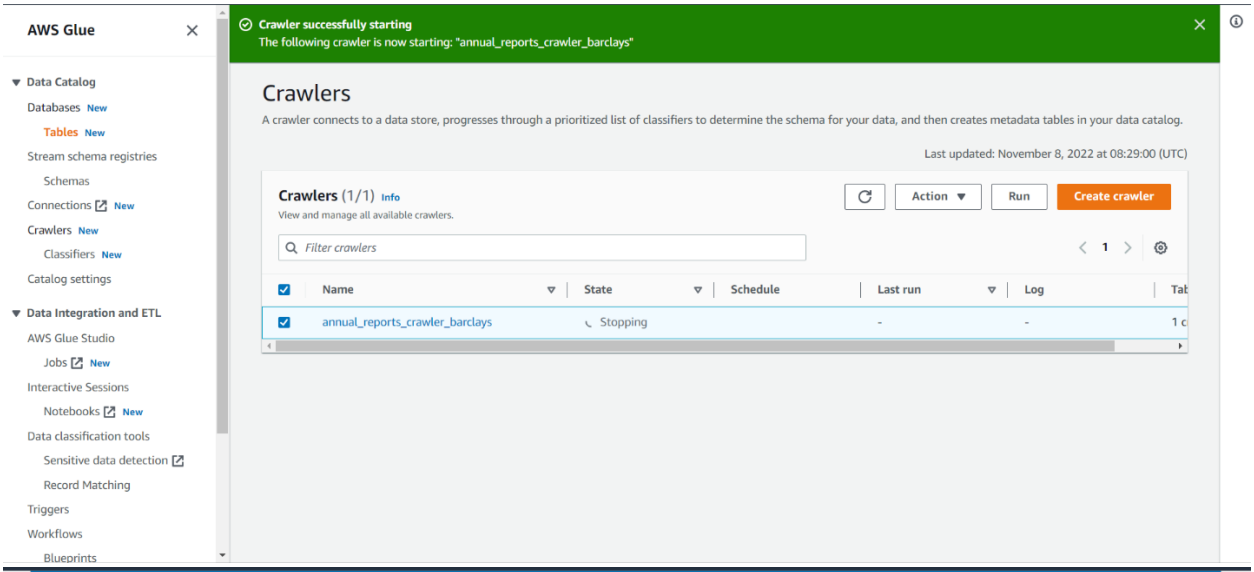

Not secure | ec2-54-167-156-74.compute-1.amazonaws.com/SamplePage.php
Sample page
NAME
ADDRESS
Add Data
ID
NAME
ADDRESS

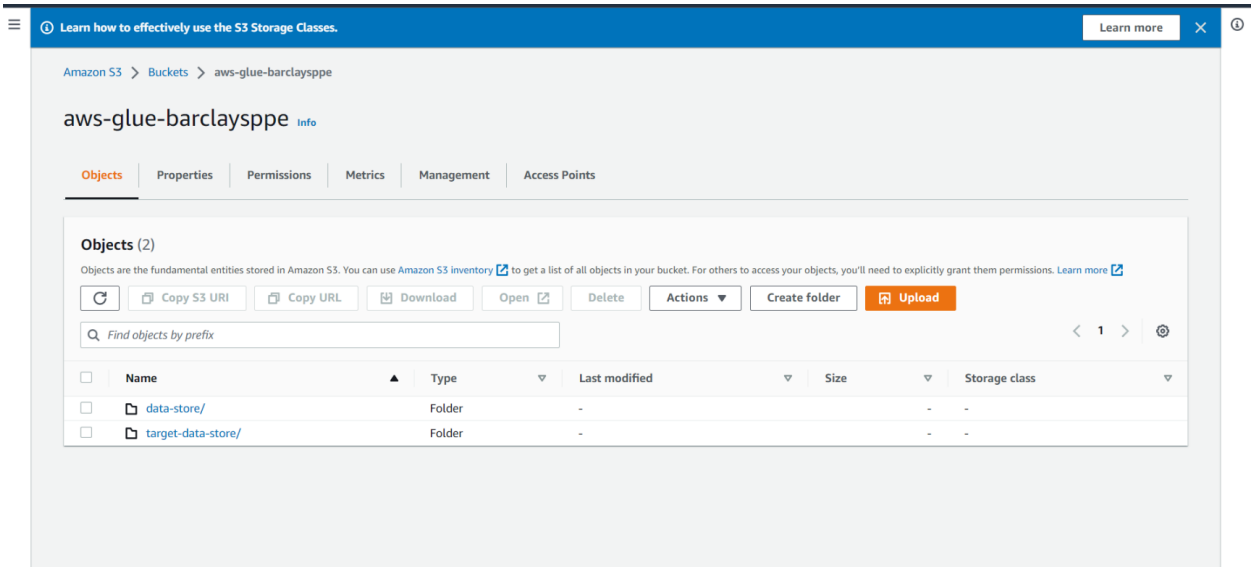
Assignment: GLUE Crawler

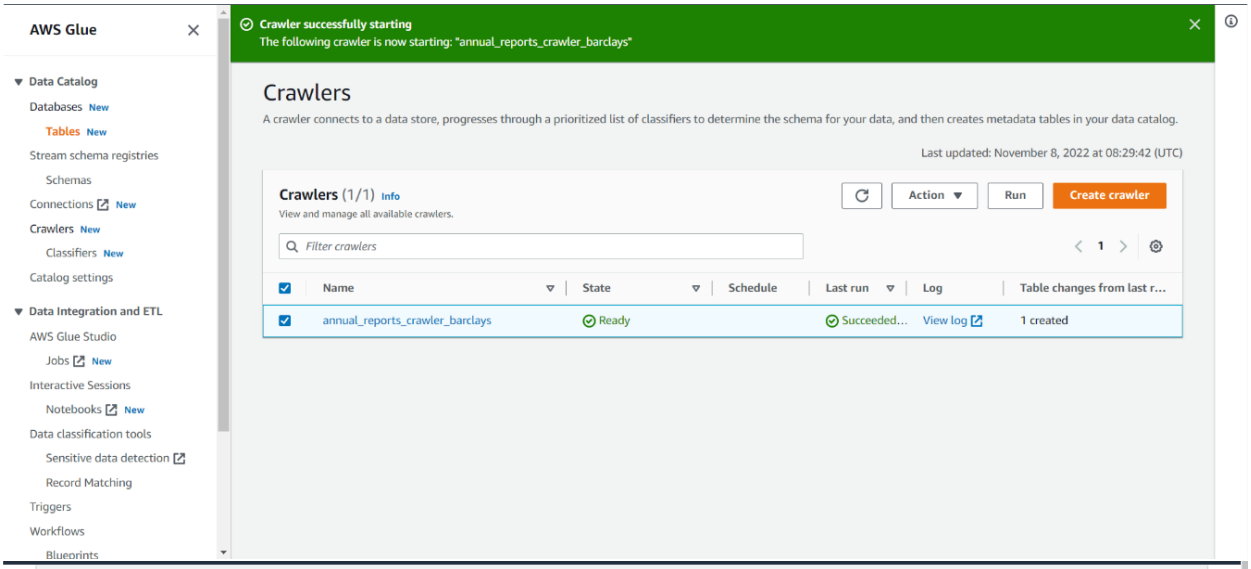
1: Creation of IAM Role for Glue with AWSGlueServiceRole and S3FullAccess Permission

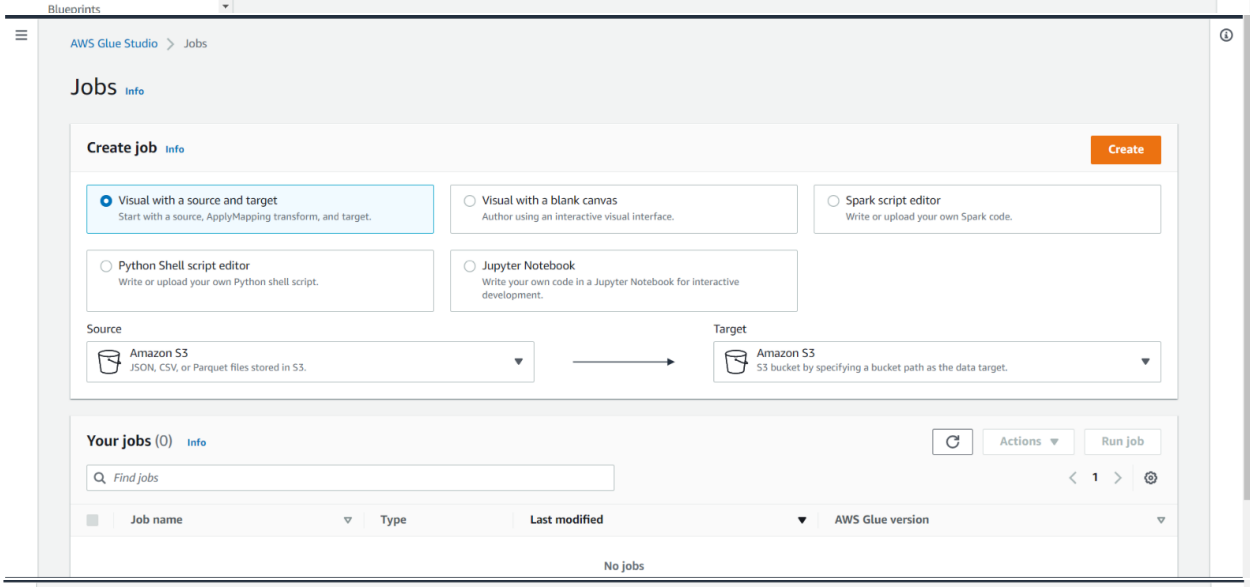
2: Now, IN S3 directory structure needs to be created. First directory will be created along with uploaded csv file. Second Structure will be created without any file.

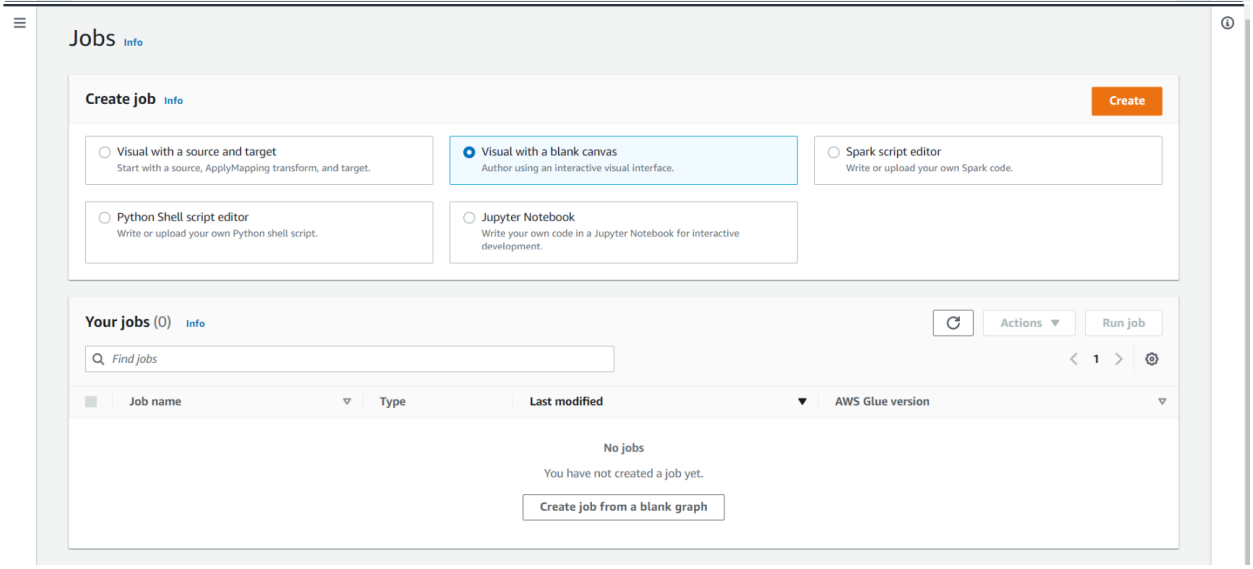
3: Now in Glue, database configuration, table information and manage glue studio for declaring jobs needs to be done.

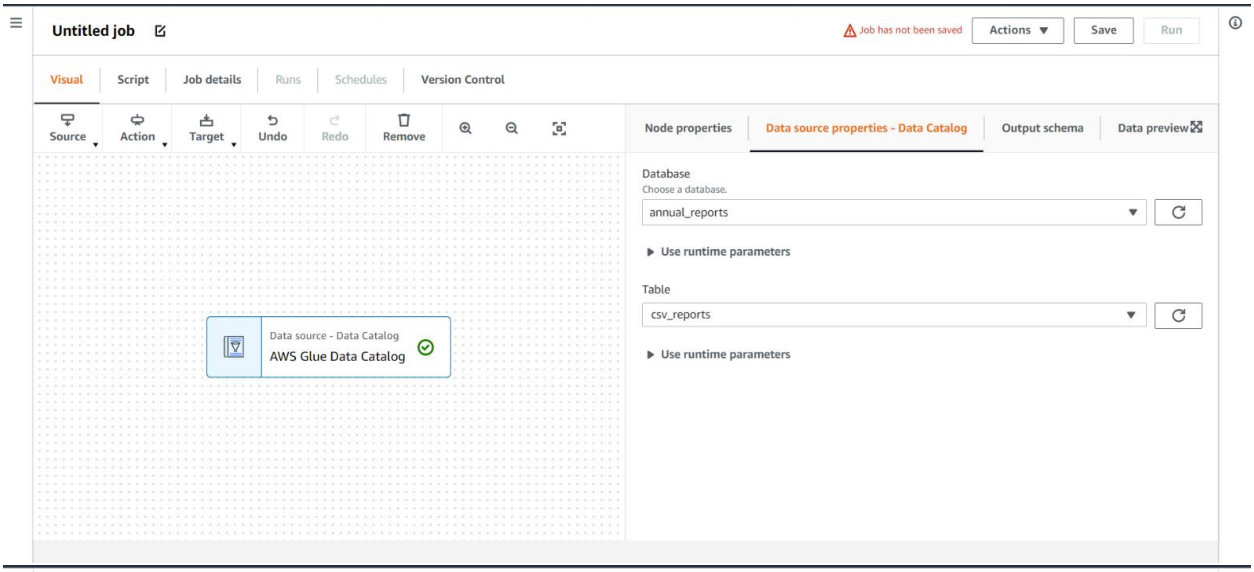


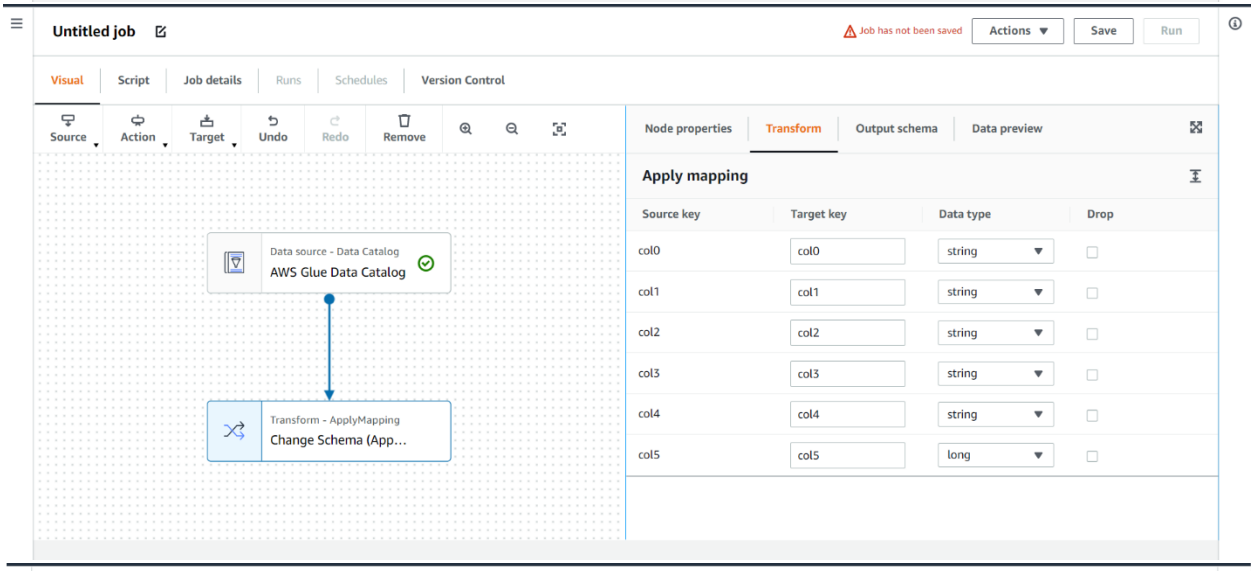


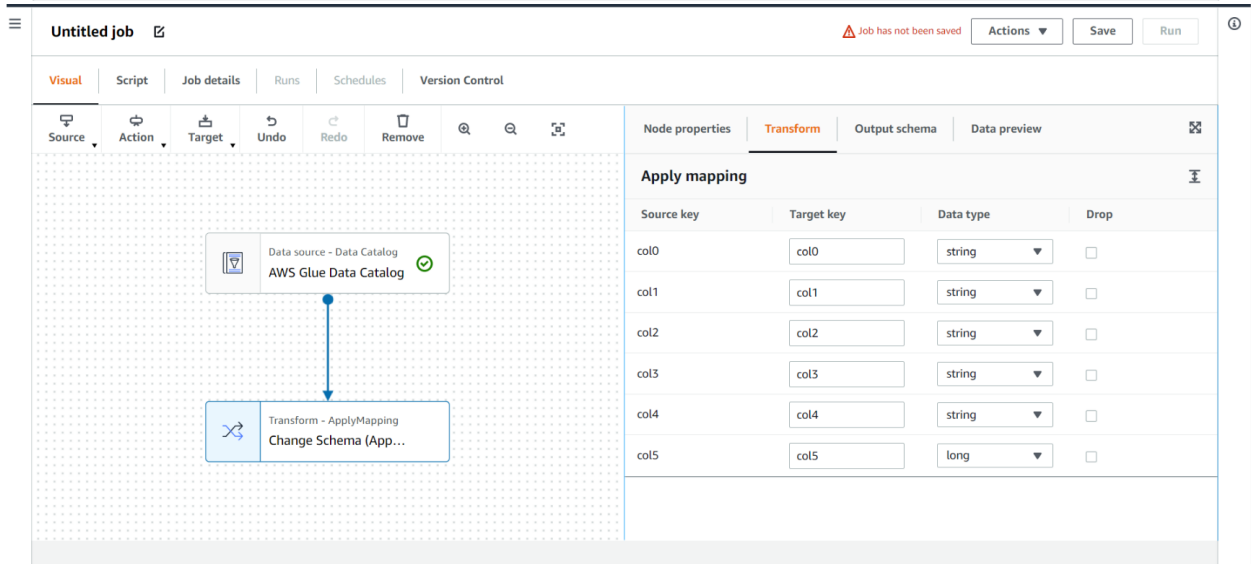

AWS Glue
Crawler successfully starting
The following crawler is now starting: "annual_reports_crawler_barclays"
Data Catalog
Databases New
Tables New
Stream schema registries
Schemas
Connections New
Crawlers New
Classifiers New
Catalog settings
Data Integration and ETL
AWS Glue Studio
Jobs New
Interactive Sessions
Notebooks New
Data classification tools
Sensitive data detection
Record Matching
Triggers
Workflows
Blueprints
Crawlers
A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.
Last updated: November 8, 2022 at 08:29:42 (UTC)
Crawlers (1/1) Info
View and manage all available crawlers.
Action
Run
Create crawler
Filter crawlers
Name
State
Schedule
Last run
Log
Table changes from last r...
annual_reports_crawler_barclays
Ready
Succeeded...
View log
1 created

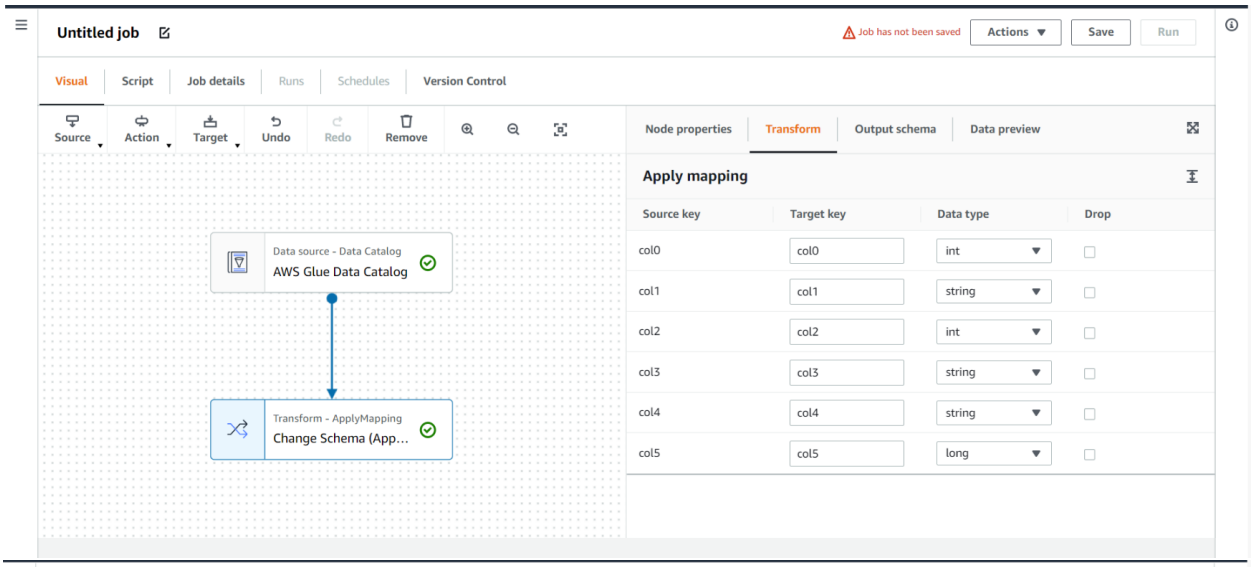

AWS Glue Studio
Jobs
Jobs Info
Create job Info
Create
Visual with a source and target
Start with a source, ApplyMapping transform, and target.
Visual with a blank canvas
Author using an interactive visual interface.
Spark script editor
Write or upload your own Spark code.
Python Shell script editor
Write or upload your own Python shell script.
Jupyter Notebook
Write your own code in a Jupyter Notebook for interactive development.
Source
Amazon S3
JSON, CSV, or Parquet files stored in S3.
Target
Amazon S3
S3 bucket by specifying a bucket path as the data target.
Your jobs (0) Info
Actions
Run job
Find jobs
Job name
Type
Last modified
AWS Glue version
No jobs

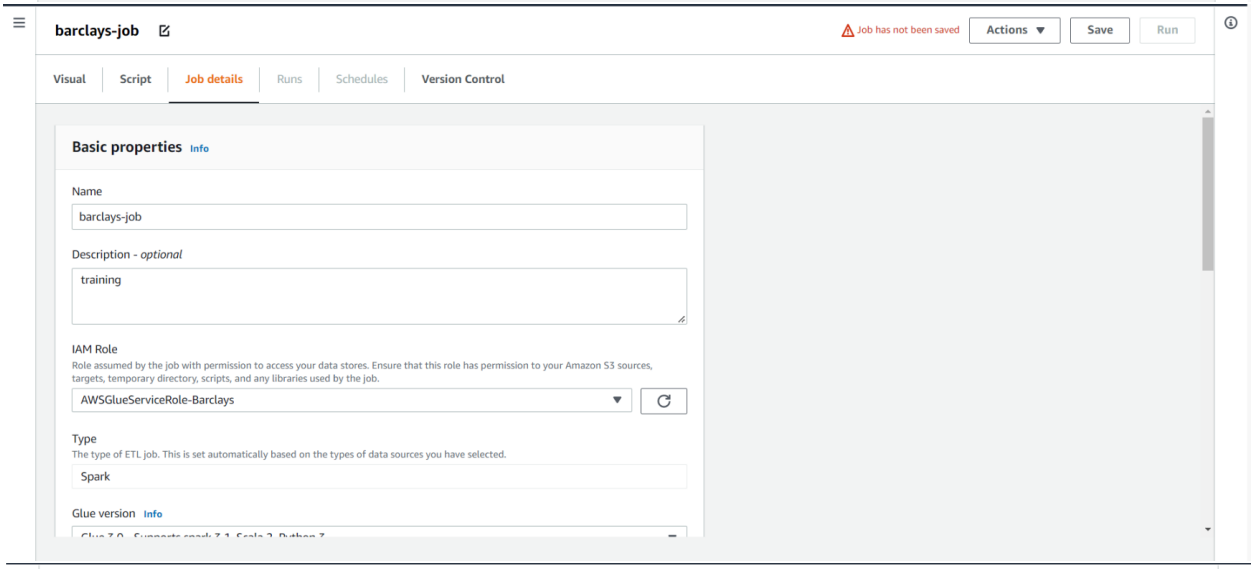

Jobs Info
Create job Info
Create
Visual with a source and target
Start with a source, ApplyMapping transform, and target.
Visual with a blank canvas
Author using an interactive visual interface.
Spark script editor
Write or upload your own Spark code.
Python Shell script editor
Write or upload your own Python shell script.
Jupyter Notebook
Write your own code in a Jupyter Notebook for interactive development.
Your jobs (0) Info
Actions
Run job
Find jobs
Job name
Type
Last modified
AWS Glue version
No jobs
You have not created a job yet.
Create job from a blank graph

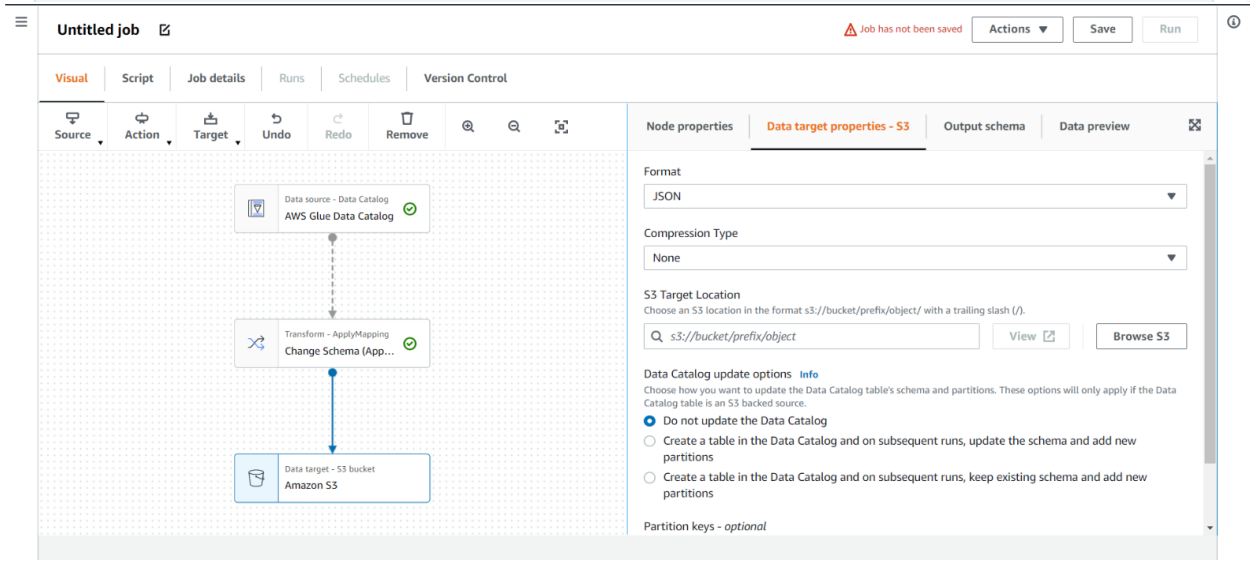

Untitled job
Job has not been saved
Actions
Save
Run
Visual
Script
Job details
Runs
Schedules
Version Control
Source
Action
Target
Undo
Redo
Remove
Node properties
Data source properties - Data Catalog
Output schema
Data preview
Database
Choose a database.
annual_reports
Use runtime parameters
Table
csv_reports
Use runtime parameters
Data source - Data Catalog
AWS Glue Data Catalog

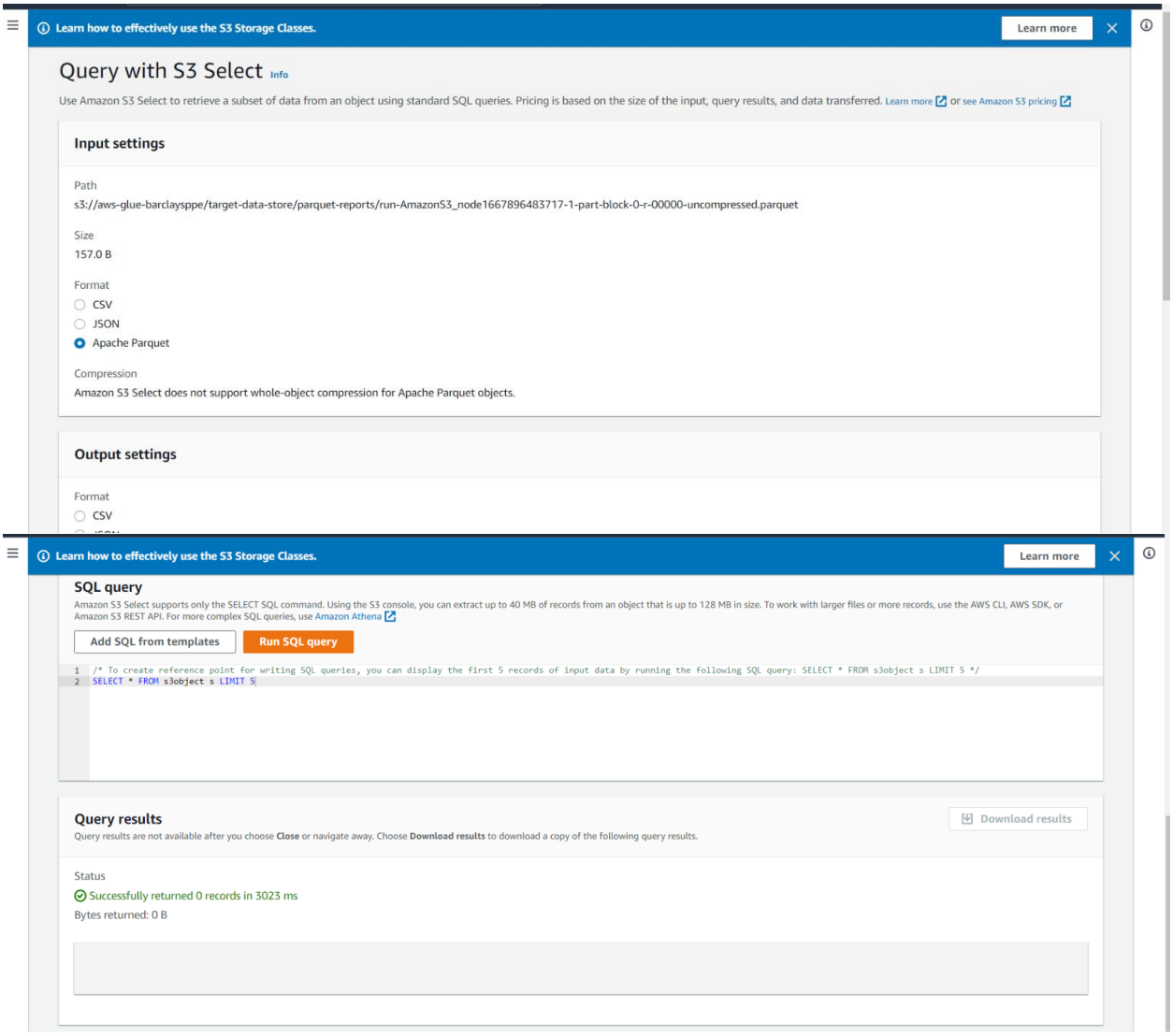

Untitled job
Job has not been saved
Actions
Save
Run
Visual
Script
Job details
Runs
Schedules
Version Control
Source
Action
Target
Undo
Redo
Remove
Node properties
Transform
Output schema
Data preview
Apply mapping
Source key
Target key
Data type
Drop
col0
col0
string
col1
col1
string
col2
col2
string
col3
col3
string
col4
col4
string
col5
col5
long
Data source - Data Catalog
AWS Glue Data Catalog
Transform - ApplyMapping
Change Schema (App...


Untitled job
Job has not been saved
Actions
Save
Run
Visual
Script
Job details
Runs
Schedules
Version Control
Source
Action
Target
Undo
Redo
Remove
Node properties
Transform
Output schema
Data preview
Apply mapping
Source key
Target key
Data type
Drop
col0
col0
string
col1
col1
string
col2
col2
string
col3
col3
string
col4
col4
string
col5
col5
long
Data source - Data Catalog
AWS Glue Data Catalog
Transform - ApplyMapping
Change Schema (App...


Untitled job
Job has not been saved
Actions
Save
Run
Visual
Script
Job details
Runs
Schedules
Version Control
Source
Action
Target
Undo
Redo
Remove
Node properties
Transform
Output schema
Data preview
Apply mapping
Source key
Target key
Data type
Drop
col0
col0
int
col1
col1
string
col2
col2
int
col3
col3
string
col4
col4
string
col5
col5
long
Data source - Data Catalog
AWS Glue Data Catalog
Transform - ApplyMapping
Change Schema (App...


barclays-job
Job has not been saved
Actions
Save
Run
Visual
Script
Job details
Runs
Schedules
Version Control
Basic properties
Info
Name
barclays-job
Description - optional
training
IAM Role
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.
AWSGlueServiceRole-Barclays
Type
The type of ETL job. This is set automatically based on the types of data sources you have selected.
Spark
Glue version
Info


Untitled job
Job has not been saved
Actions
Save
Run
Visual
Script
Job details
Runs
Schedules
Version Control
Source
Action
Target
Undo
Redo
Remove
Node properties
Data target properties - S3
Output schema
Data preview
Data source - Data Catalog
AWS Glue Data Catalog
Transform - ApplyMapping
Change Schema (App...
Data target - S3 bucket
Amazon S3
Format
JSON
Compression Type
None
S3 Target Location
Choose an S3 location in the format s3://bucket/prefix/object/ with a trailing slash (/).
s3://bucket/prefix/object
View
Browse S3
Data Catalog update options
Info
Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog table is an S3 backed source.
Do not update the Data Catalog
Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions
Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions
Partition keys - optional


Learn how to effectively use the S3 Storage Classes.
Learn more
Query with S3 Select Info
Use Amazon S3 Select to retrieve a subset of data from an object using standard SQL queries. Pricing is based on the size of the input, query results, and data transferred. Learn more or see Amazon S3 pricing
Input settings
Path
s3://aws-glue-barclayspe/target-data-store/parquet-reports/run-AmazonS3_node1667896483717-1-part-block-0-r-00000-uncompressed.parquet
Size
157.0 B
Format
CSV
JSON
Apache Parquet
Compression
Amazon S3 Select does not support whole-object compression for Apache Parquet objects.
Output settings
Format
CSV
Learn how to effectively use the S3 Storage Classes.
Learn more
SQL query
Amazon S3 Select supports only the SELECT SQL command. Using the S3 console, you can extract up to 40 MB of records from an object that is up to 128 MB in size. To work with larger files or more records, use the AWS CLI, AWS SDK, or Amazon S3 REST API. For more complex SQL queries, use Amazon Athena
Add SQL from templates
Run SQL query
1 /* To create reference point for writing SQL queries, you can display the first 5 records of input data by running the following SQL query: SELECT * FROM s3object s LIMIT 5 */
2 SELECT * FROM s3object s LIMIT 5
Query results
Download results
Query results are not available after you choose Close or navigate away. Choose Download results to download a copy of the following query results.
Status
Successfully returned 0 records in 3023 ms
Bytes returned: 0 B

Assignment: Amplify, Lambda & DynamoDB

Steps:

1. Create Web App: Deploy static resources for your web application using the AWS Amplify Console.
2. Build Serverless Function: Build a serverless function using AWS Lambda.
3. Link Serverless Function to Web App: Deploy your serverless function with API Gateway.
4. Create Data Table: Persist data in an Amazon DynamoDB table.
5. Add Interactivity to Web App: Modify your web app to invoke your API.

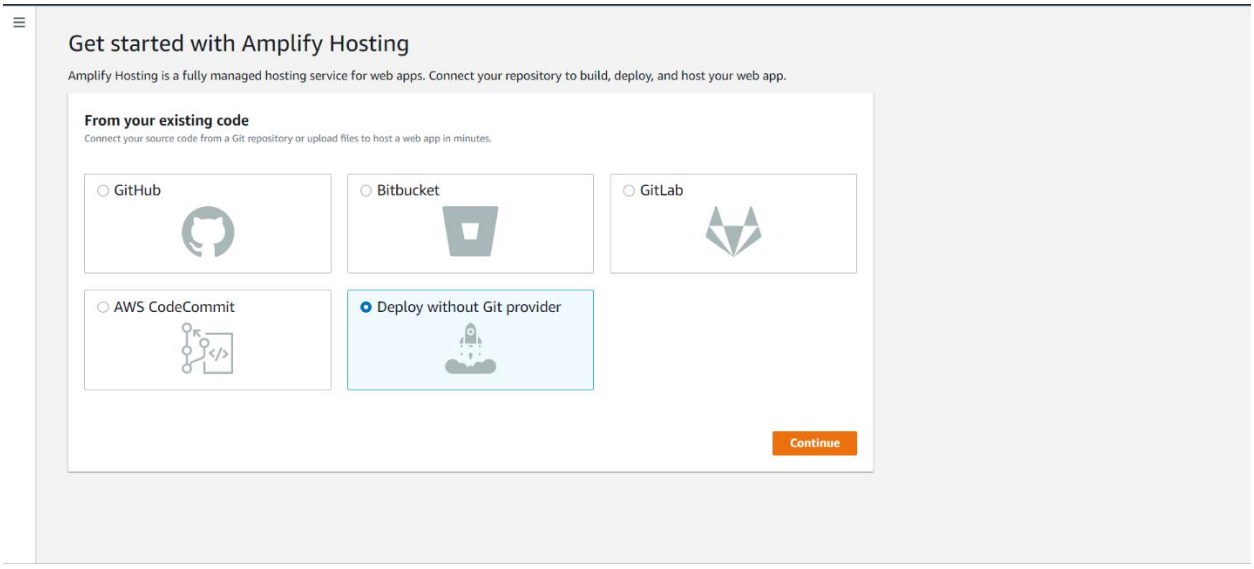
1. Create Web App with Amplify Console

- Open your favorite text editor on your computer. Create a new file and paste the following HTML in it:

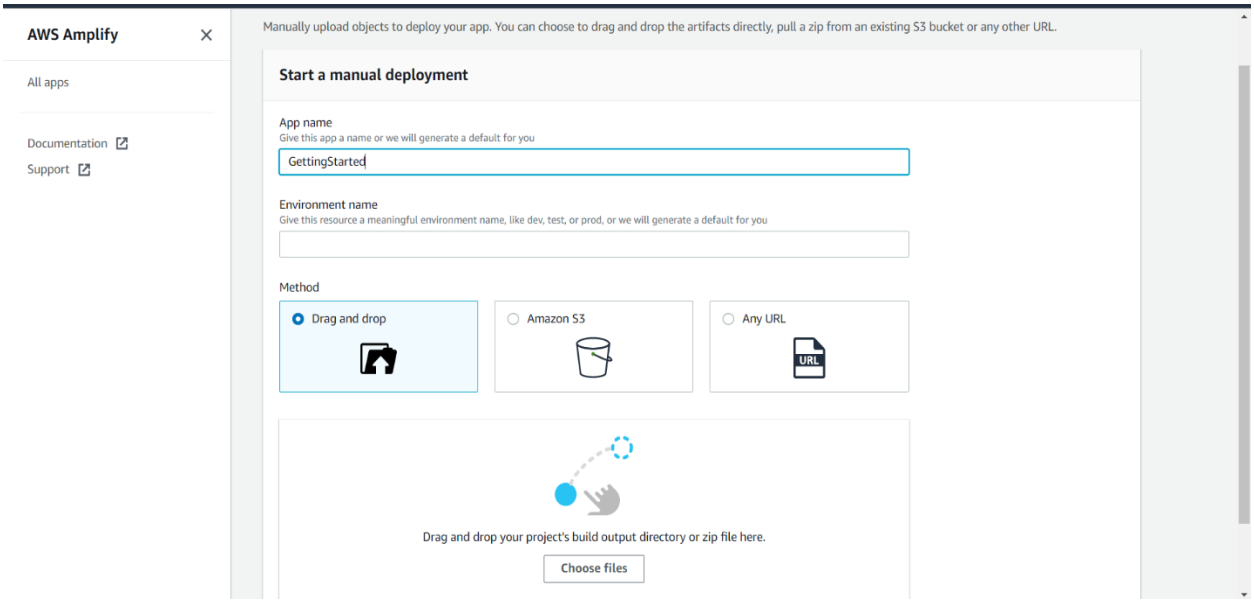
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Hello World</title>
</head>

<body>
  Hello World
</body>
</html>
```

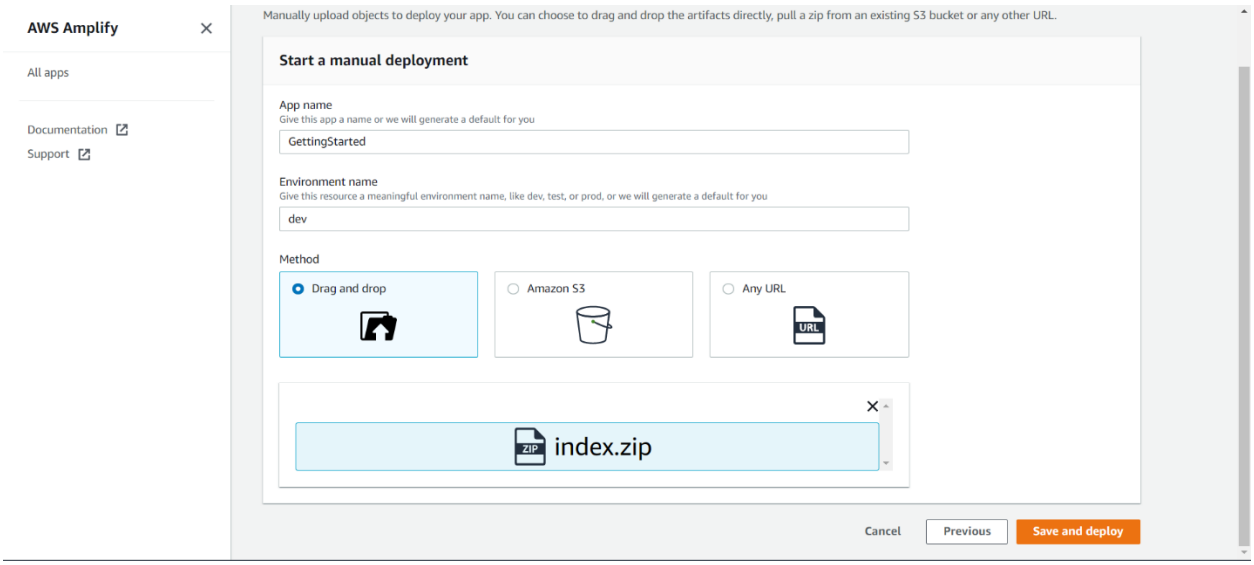
- Save the file as index.html.
- In a new browser window, log into the Amplify console. Note: We will be using the Oregon (us-west-2) Region for this tutorial.]



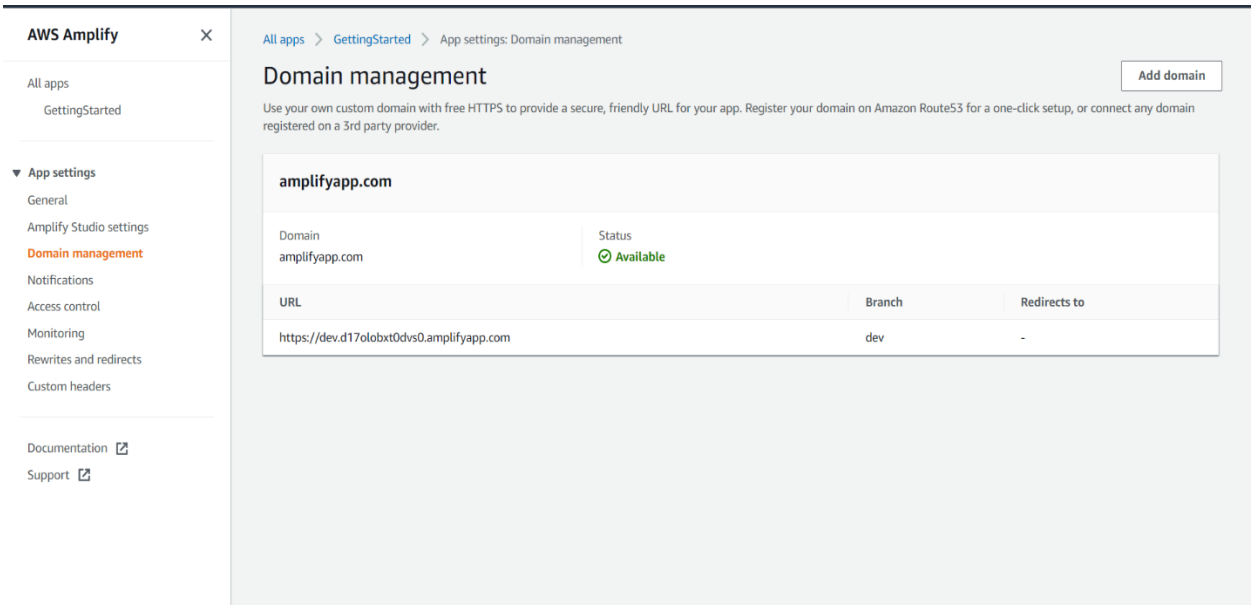
- In the Get Started section, under Host your web app, choose the orange Get started button.
- Select Deploy without Git provider.
- Choose the Continue button.
- In the App name field, enter GettingStarted.



- For Environment name, enter dev.
- Select the Drag and drop method.
- Choose the Choose files button.
- Select the ZIP file you created in Step 3.

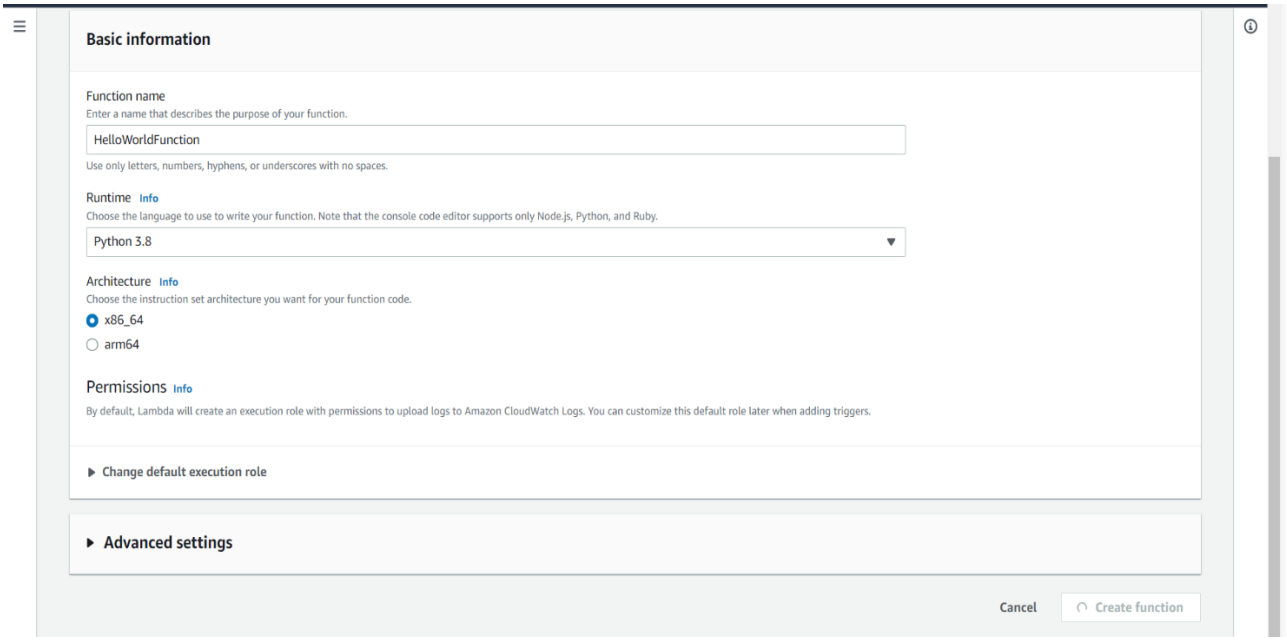


- Choose the Save and deploy button.
- After a few seconds, you should see the message Deployment successfully completed.

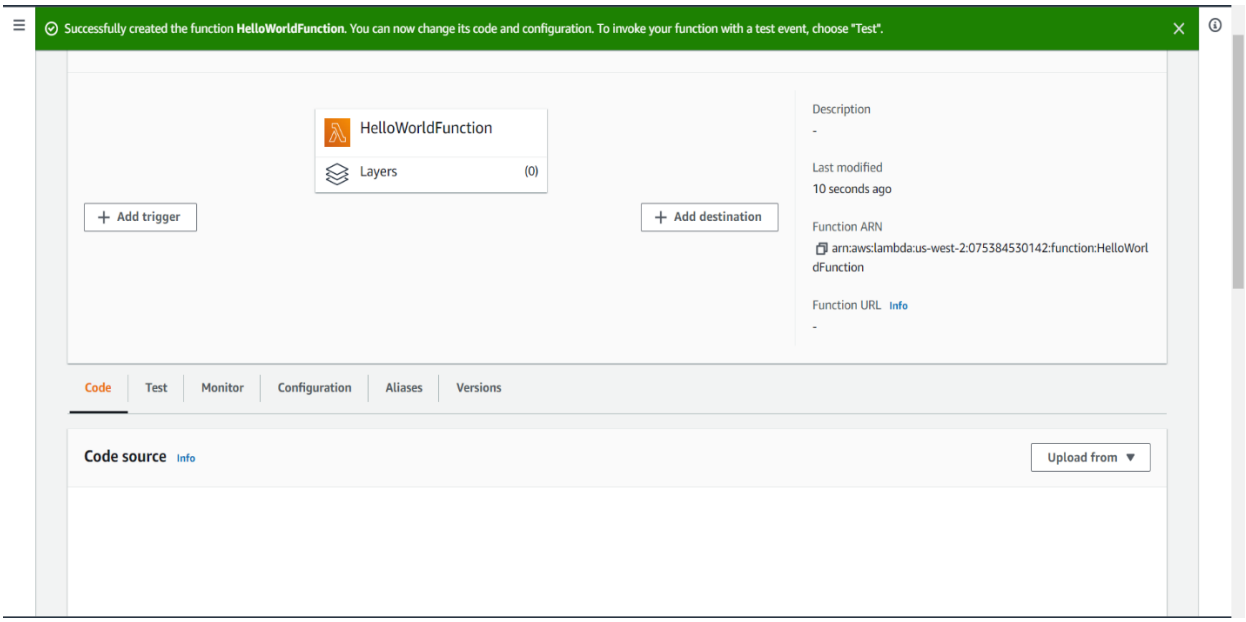


2. Build a Serverless Function

- In a new browser tab, log in to the AWS Lambda console.
- Make sure you create your function in the same Region in which you created the web app in the previous module. You can see this at the very top of the page, next to your account name.
- Choose the orange Create function button.
- Under Function name, enter HelloWorldFunction.

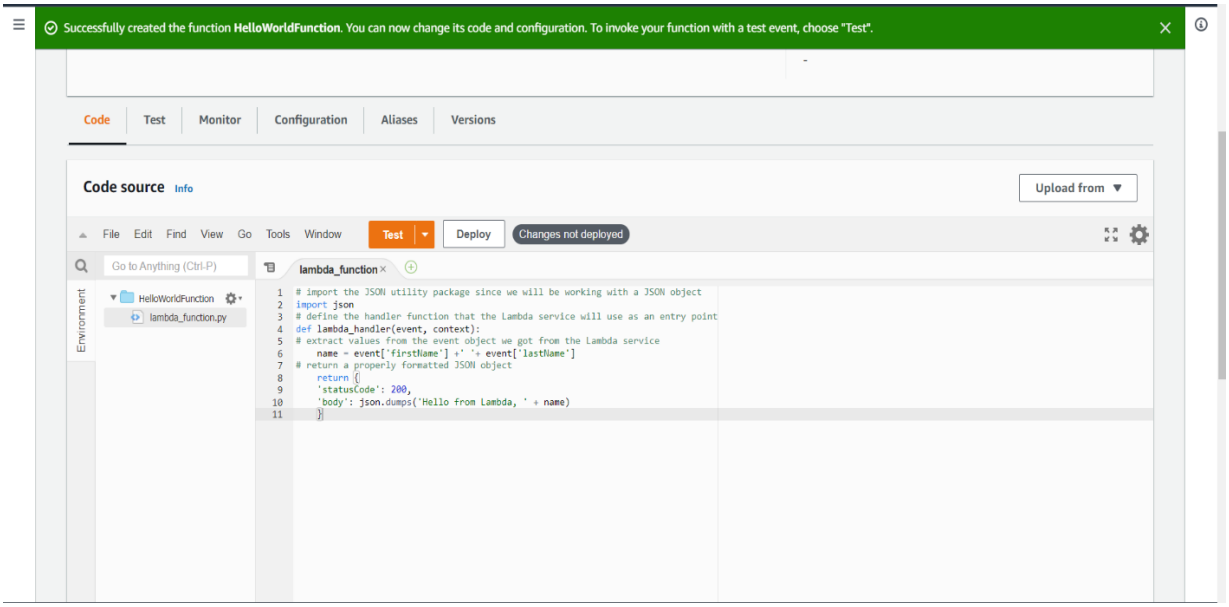


- Select Python 3.8 from the runtime dropdown and leave the rest of the defaults unchanged.
- Choose the orange Create function button.
- You should see a green message box at the top of your screen with the following message "Successfully created the function HelloWorldFunction."

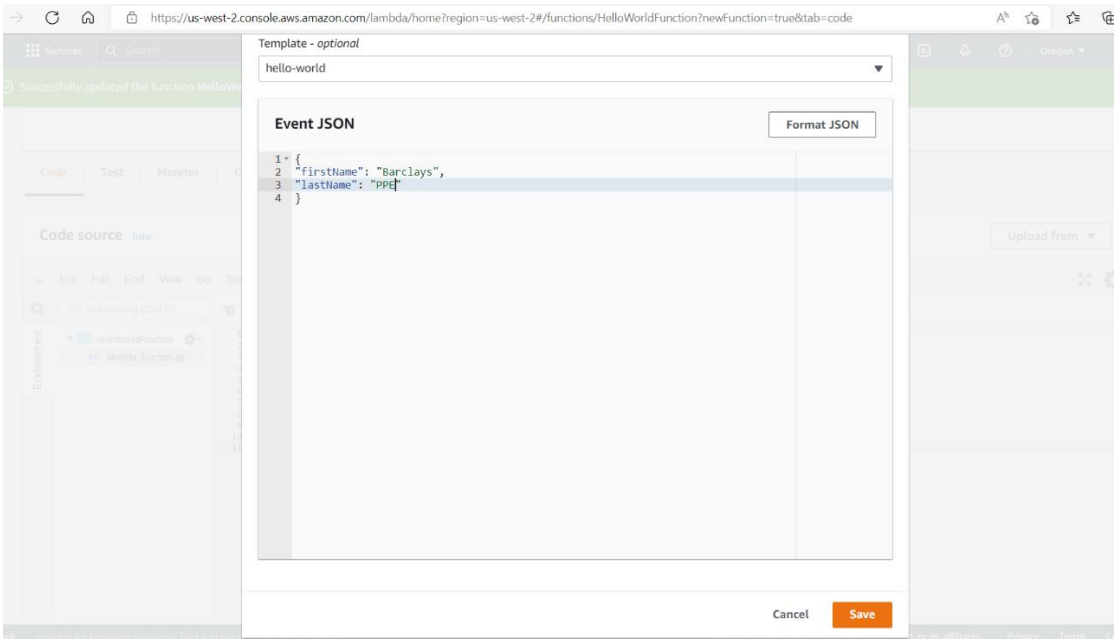


- Under Code source, replace the code in lambda_function.py with the following:

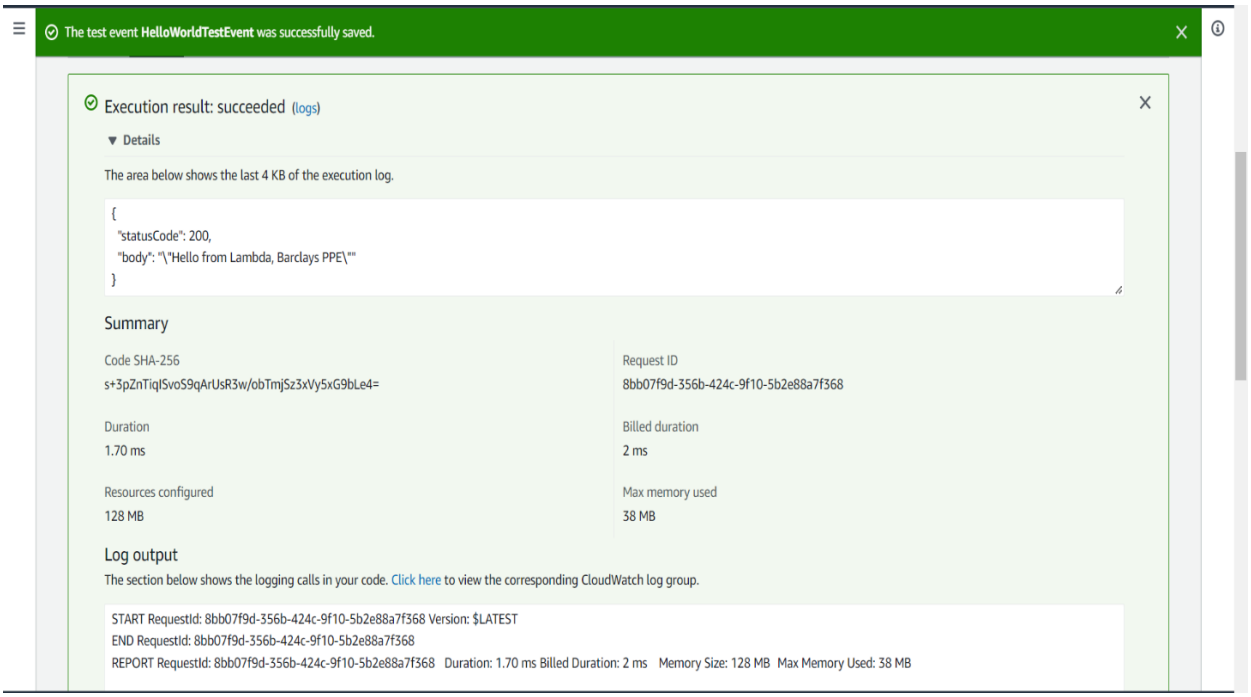
```
# import the JSON utility package since we will be working with a JSON object
import json
# define the handler function that the Lambda service will use as an entry point
def lambda_handler(event, context):
# extract values from the event object we got from the Lambda service
   name = event['firstName'] +' '+ event['lastName']
# return a properly formatted JSON object
   return {
   'statusCode': 200,
   'body': json.dumps('Hello from Lambda, ' + name)
   }
```



- Save by going to the file menu and selecting Save to save the changes.
- Choose Deploy to deploy the changes.
- Let's test our new function. Choose the orange Test button to create a test event by selecting Configure test event.
- Under Event name, enter HelloWorldTestEvent.
- Copy and paste the following JSON object to replace the default one:

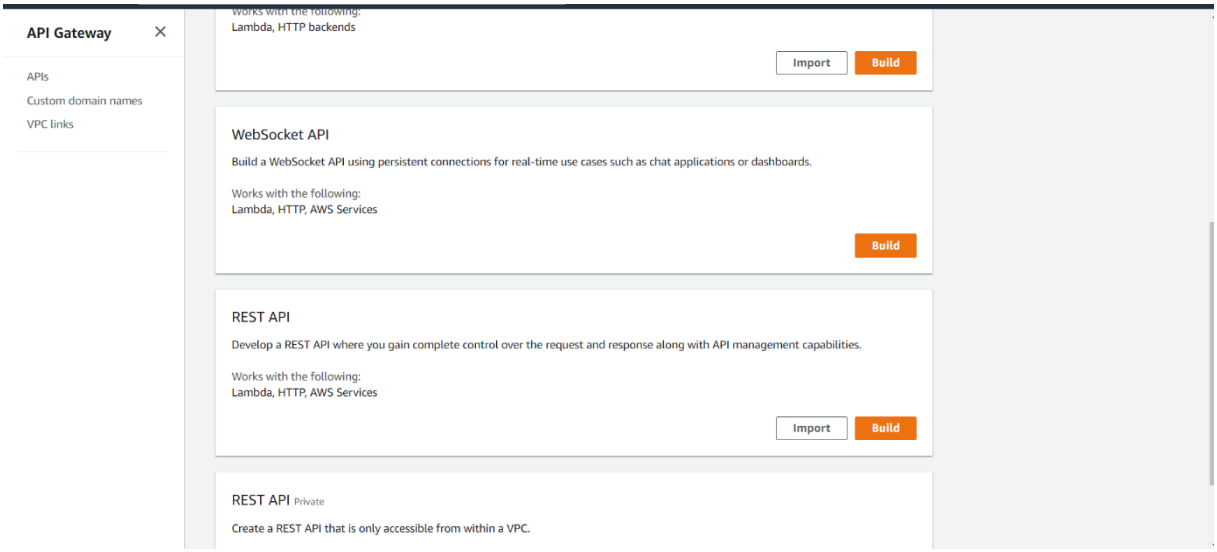


- Choose the orange Create button at the bottom of the page.



4. Link a Serverless Function to a Web App

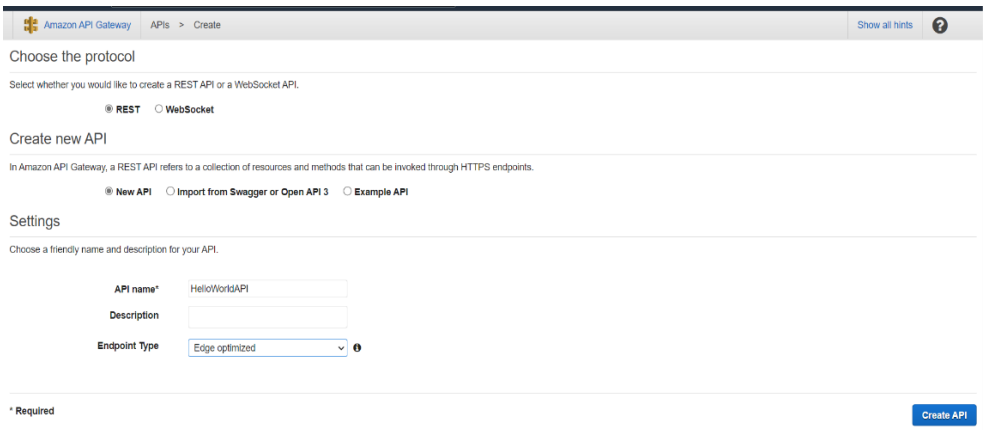
- Log in to the API Gateway console.



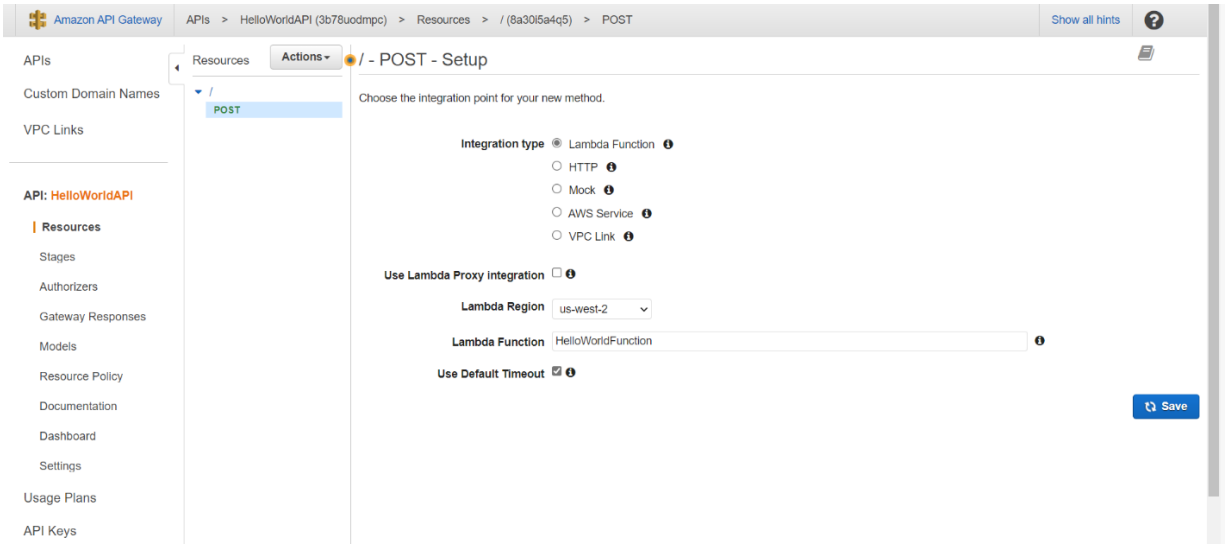
- In the Choose an API type section, find the REST API card and choose the Build button on the card.
- Under Choose the protocol, select REST.
- Under Create new API, select New API.
- In the API name field, enter HelloWorldAPI.
- Select Edge optimized from the Endpoint Type dropdown. (Note: Edge-optimized endpoints are best for geographically distributed clients. This makes them a good choice

for public services being accessed from the internet. Regional endpoints are typically used for APIs that are accessed primarily from within the same AWS Region.)

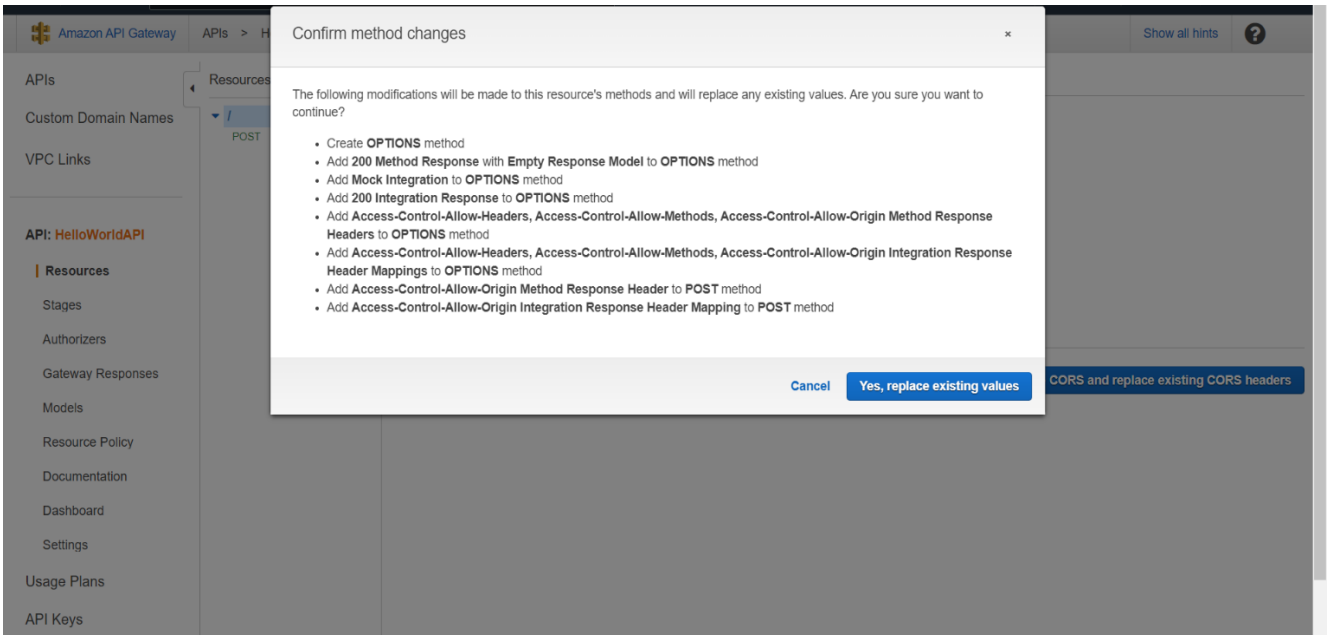
- Choose the blue Create API button. Your settings should look like the accompanying screenshot.



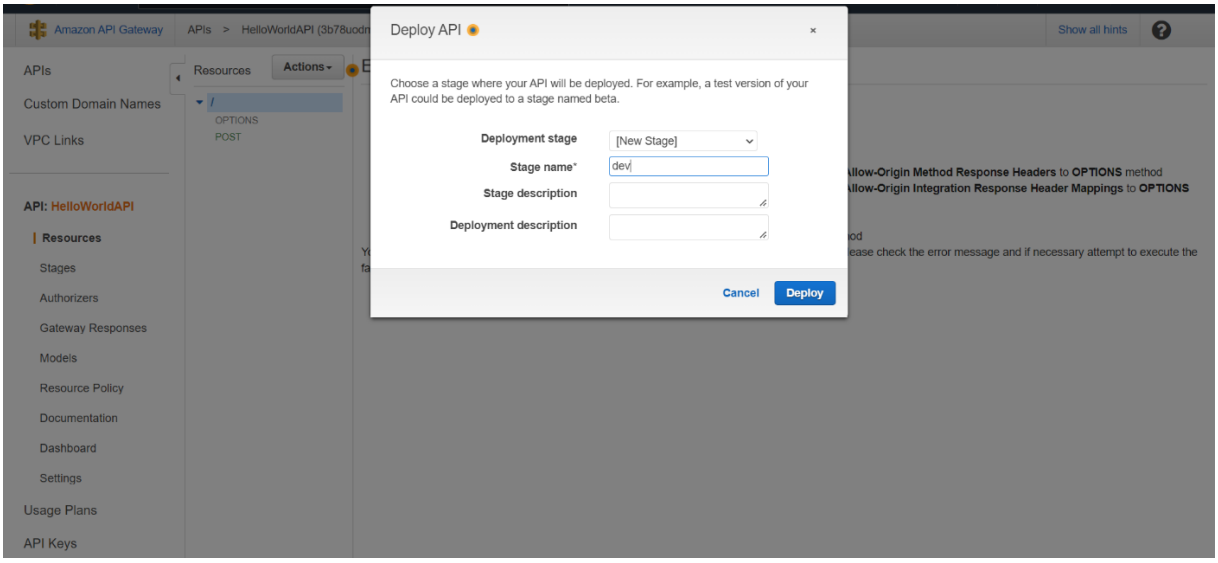
- In the left navigation pane, select Resources under API: HelloWorldAPI.
- Ensure the "/" resource is selected.
- From the Actions dropdown menu, select Create Method.
- Select POST from the new dropdown that appears, then select the checkmark.
- Select Lambda Function for the Integration type.
- Select the Lambda Region you used when making the function (or else you will see a warning box reading "You do not have any Lambda Functions in...").
- Enter HelloWorldFunction in the Lambda Function field.
- Choose the blue Save button.
- You should see a message letting you know you are giving the API you are creating permission to call your Lambda function. Choose the OK button.
- With the newly created POST method selected, select Enable CORS from the Action dropdown menu.
- Leave the POST checkbox selected and choose the blue Enable CORS and replace existing CORS headers button.



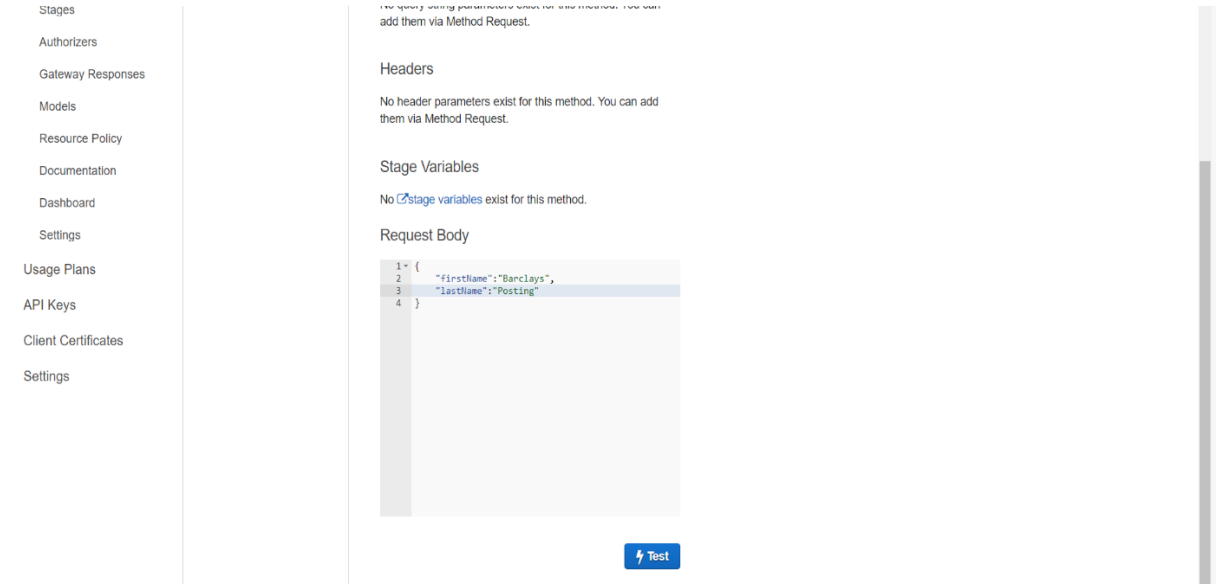
- You should see a message asking you to confirm method changes. Choose the blue Yes, replace existing values button.

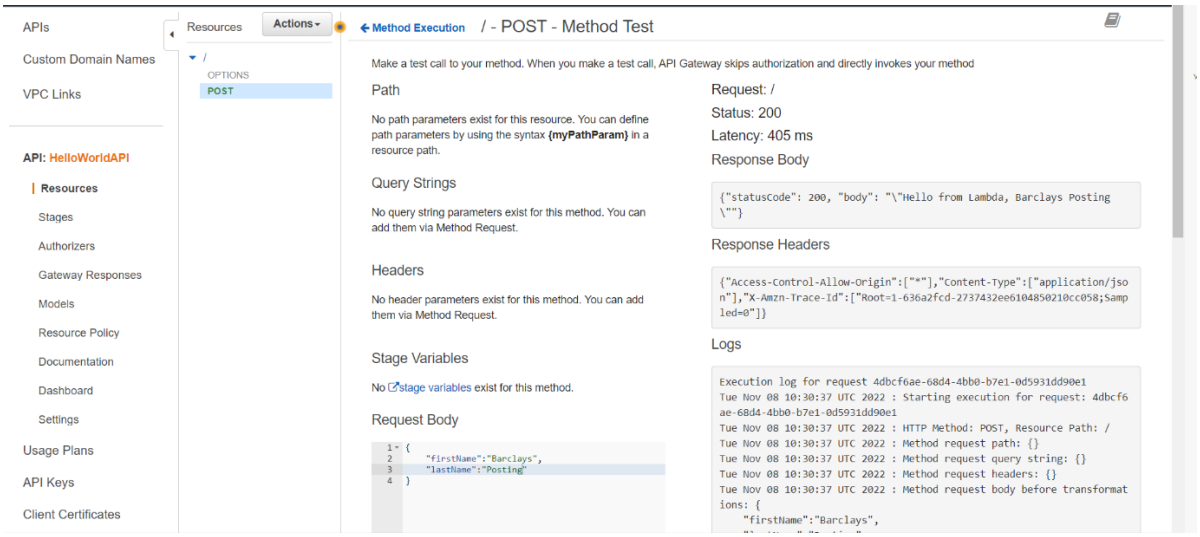


- In the Actions dropdown list, select Deploy API.
- Select [New Stage] in the Deployment stage dropdown list.
- Enter dev for the Stage Name.
- Choose Deploy.
- Copy and save the URL next to Invoke URL



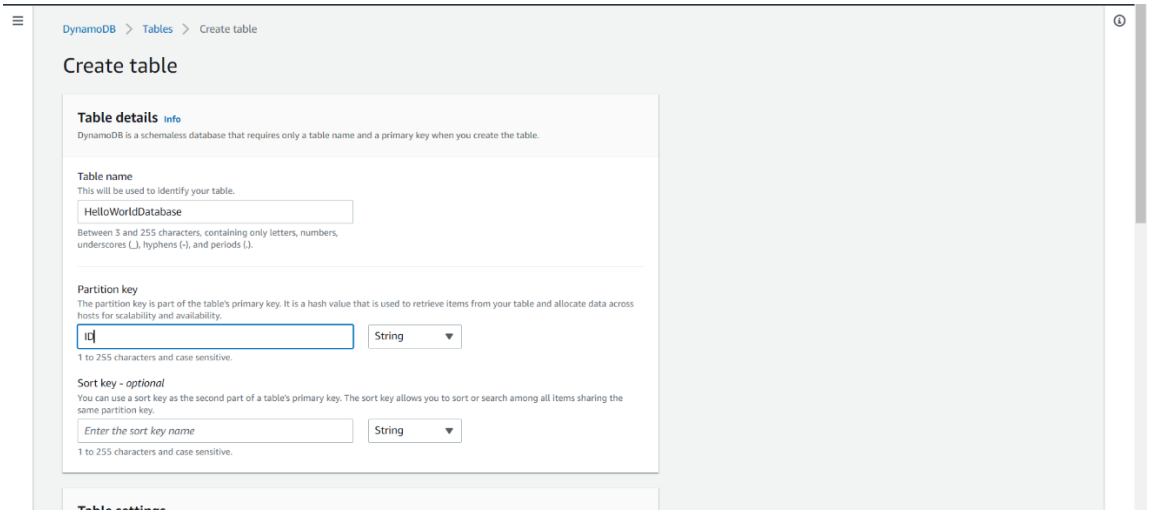
- In the left navigation pane, select Resources.
- The methods for our API will now be listed on the right. Choose POST.
- Choose the small blue lightning bolt.
- Paste the following into the Request Body field:
- Choose the blue Test button.
- On the right side, you should see a response with Code 200.
- We need to build and tested an API that calls our Lambda function.

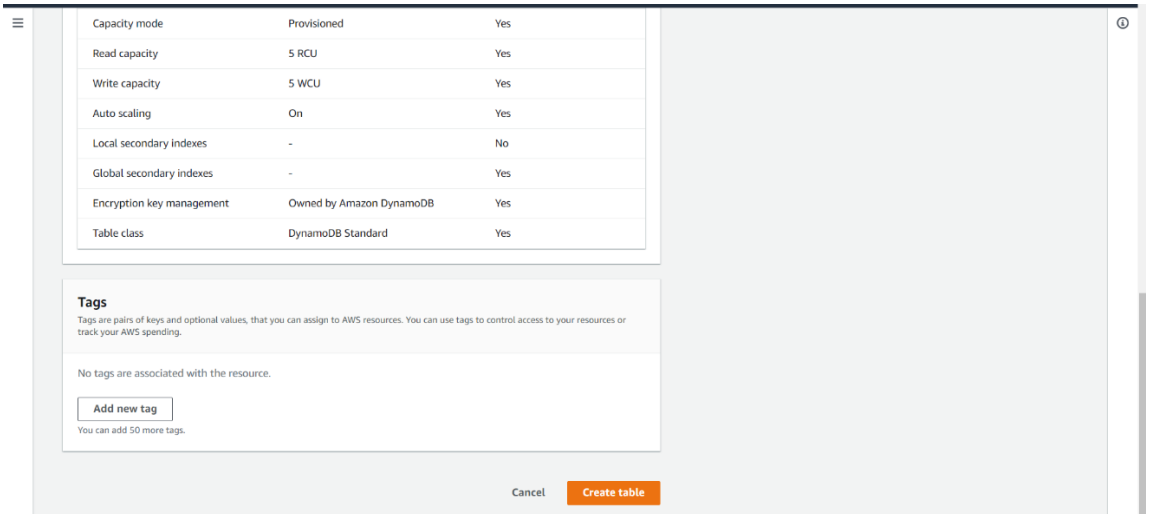


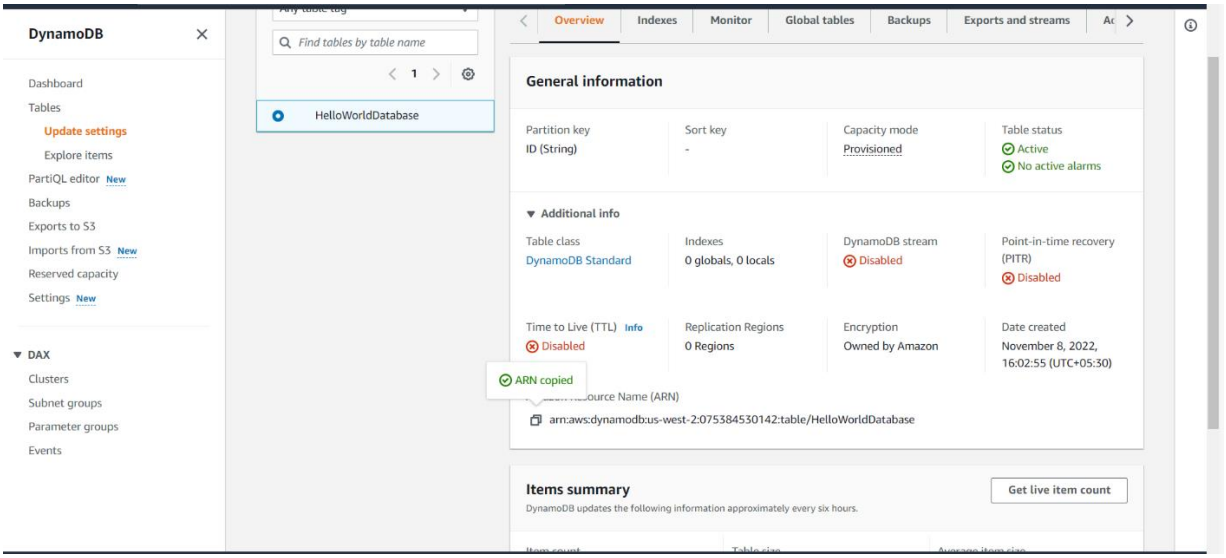
5. 

Create a Data Table

- Log in to the Amazon DynamoDB console.
- Make sure you create your table in the same Region in which you created the web app in the previous module. You can see this at the very top of the page, next to your account name.
- Choose the orange Create table button.
- Under Table name, enter HelloWorldDatabase.
- In the Partition key field, enter ID. The partition key is part of the table's primary key.
- Leave the rest of the default values unchanged and choose the orange Create table button.
- In the list of tables, select the table name, HelloWorldDatabase.
- In the General information section, show Additional info by selecting the down arrow.
- Copy the Amazon Resource Name (ARN). You will need it later in this module.

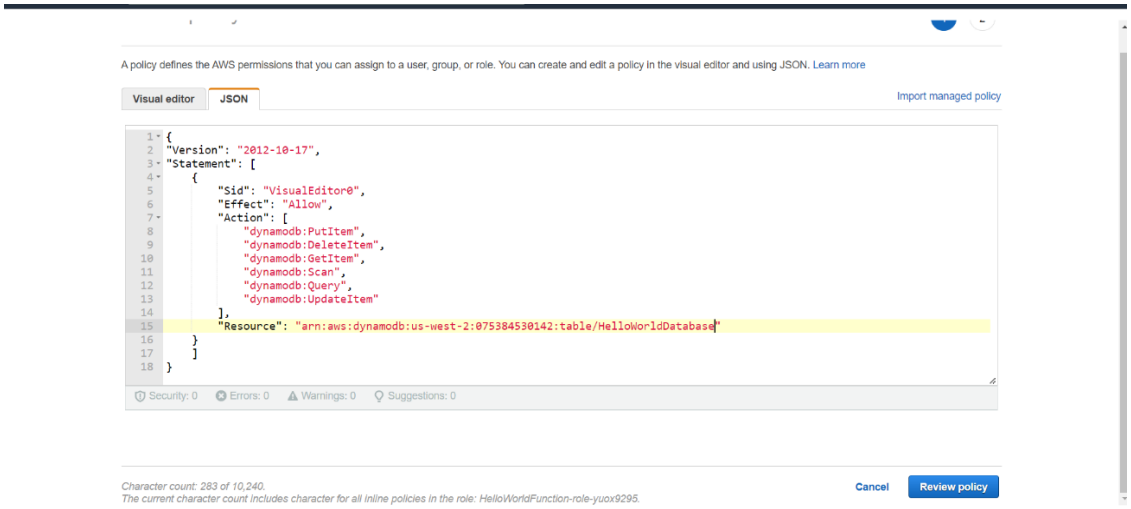


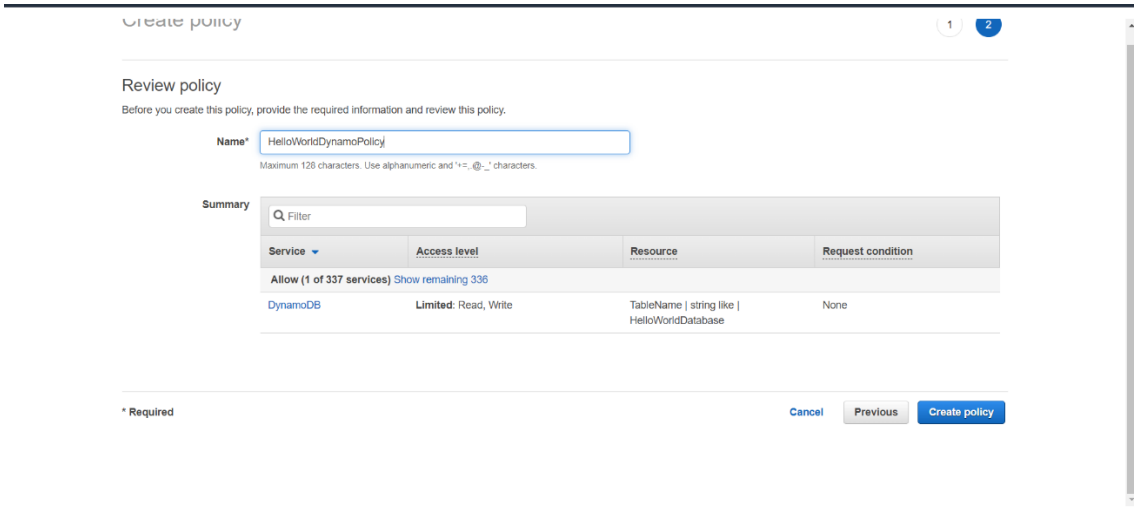




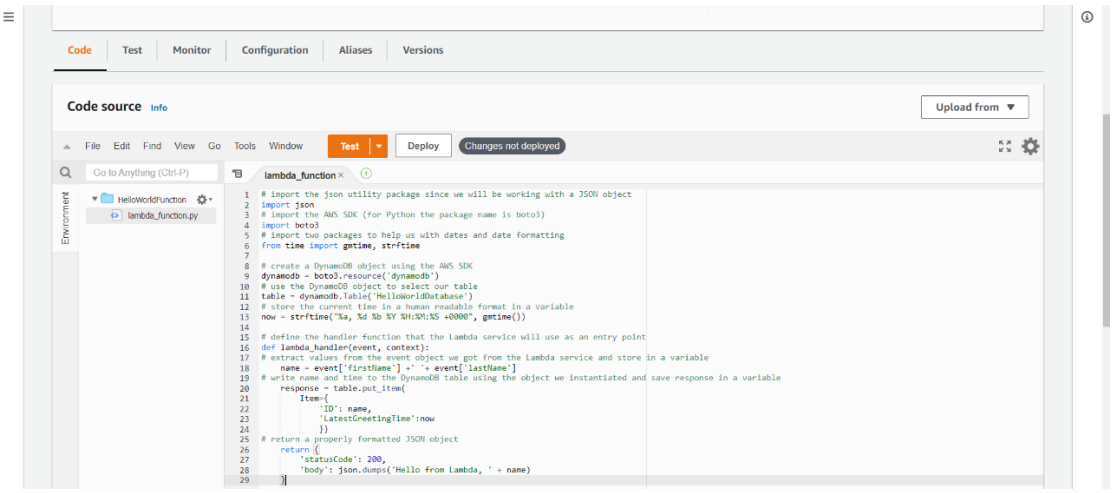
- Now that we have a table, let's edit our Lambda function to be able to write data to it. In a new browser window, open the AWS Lambda console.
- Select the function we created in module two (if you have been using our examples, it will be called HelloWorldFunction). If you don't see it, check the Region dropdown in the upper right next to your name to ensure you're in the same Region you created the function in.
- We'll be adding permissions to our function so it can use the DynamoDB service, and we'll be using AWS Identity and Access Management (IAM) to do so.
- Select the Configuration tab and select Permissions from the right side menu.
- In the Execution role box, under Role name, choose the link. A new browser tab will open.
- In the Permissions policies box, open the Add permissions dropdown and select Create inline policy.
- Select the JSON tab.

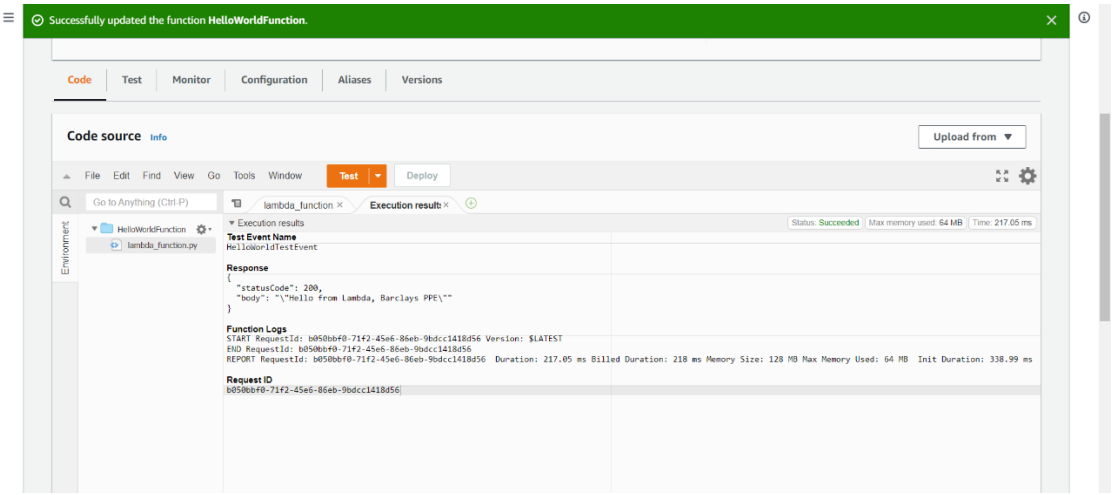
- Paste the following policy in the text area, taking care to replace your table's ARN in the Resource field in line 15:
- This policy will allow our Lambda function to read, edit, or delete items, but restrict it to only be able to do so in the table we created.
- Choose the blue Review Policy button.
- Next to Name, enter HelloWorldDynamoPolicy.
- Choose the blue Create Policy button.
- You can now close this browser tab and go back to the tab for your Lambda function.

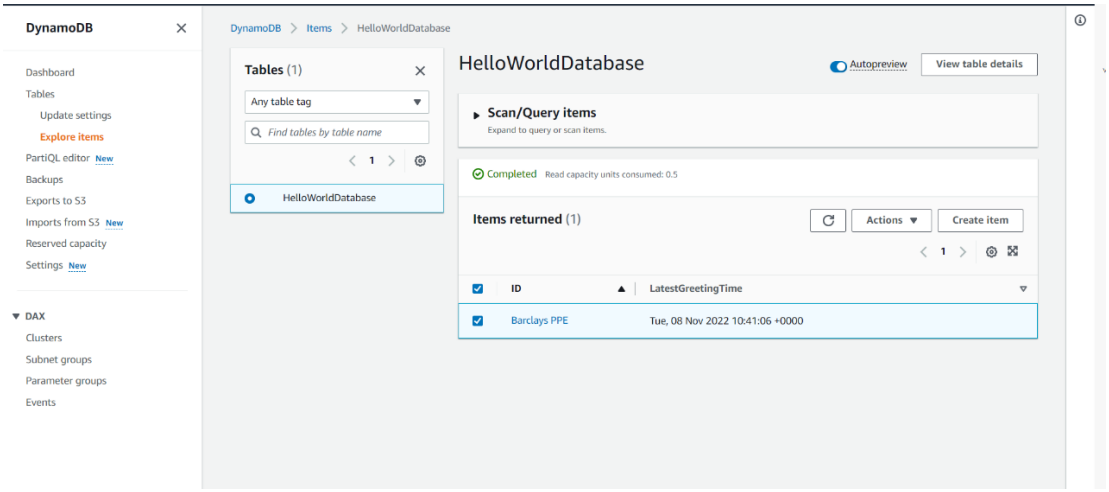




- Select the Code tab and select your function from the navigation pane on the left side of the code editor.
- Replace the code for your function with the following:
- Choose the Deploy button at the top of the code editor.







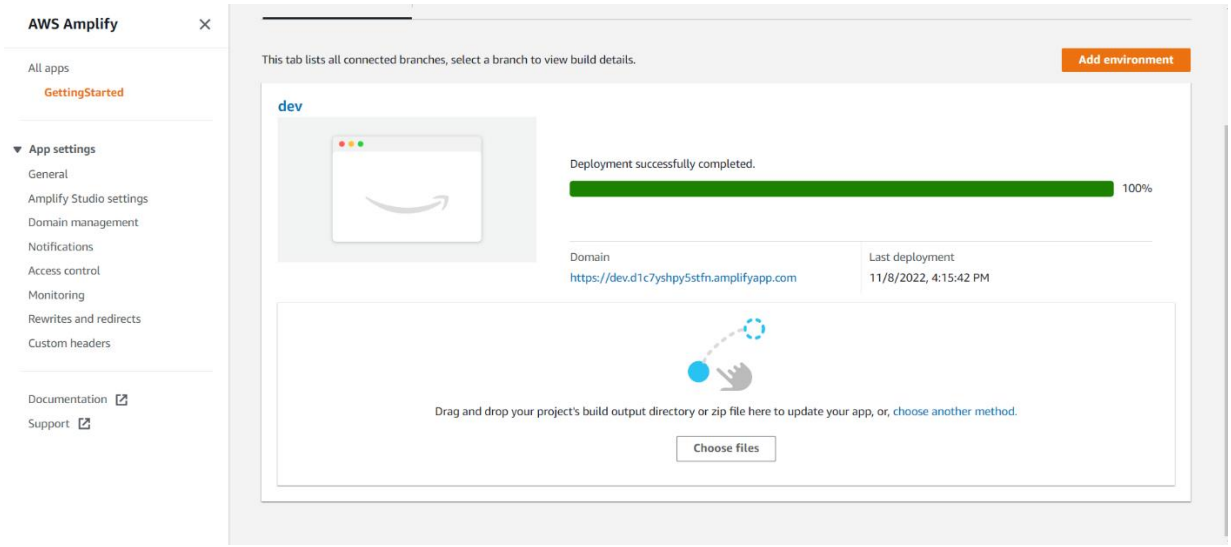
6. Add Interactivity to Your Web App

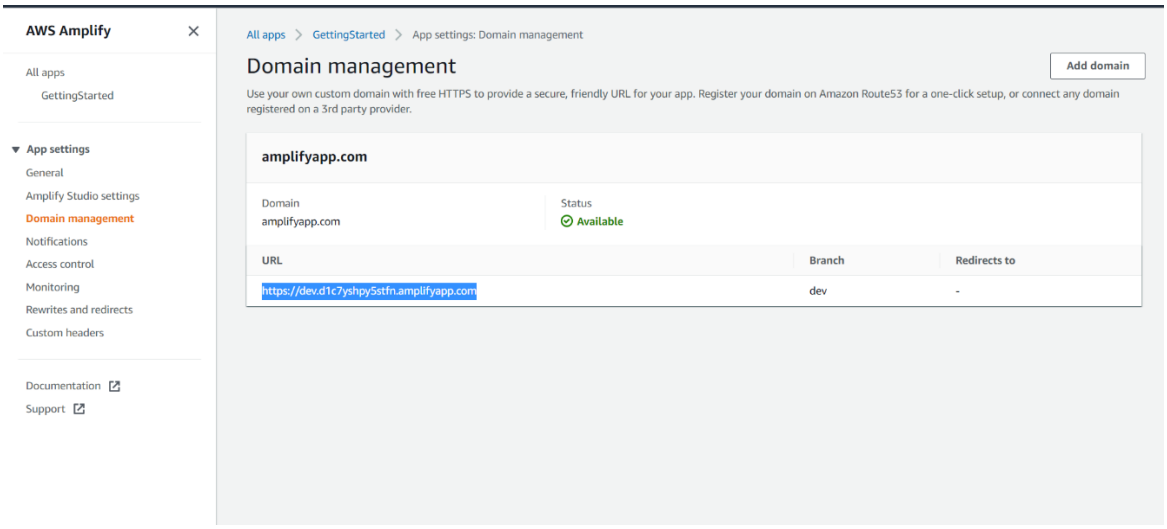
- Open the index.html file you created in module one.
- Replace the existing code with the following:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Hello World</title>
  <!-- Add some CSS to change client UI -->
  <style>
  body {
    background-color: #232F3E;
    }
  label, button {
    color: #FF9900;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 20px;
    margin-left: 40px;
    }
   input {
    color: #232F3E;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 20px;
    margin-left: 20px;
    }
  </style>
  <script>
    // define the callAPI function that takes a first name and last name as parameters
    var callAPI = (firstName,lastName)=>{
      // instantiate a headers object
      var myHeaders = new Headers();
      // add content type header to object
      myHeaders.append("Content-Type", "application/json");
      // using built in JSON utility package turn object to string and store in a variable
      var raw = JSON.stringify({"firstName":firstName,"lastName":lastName});
      // create a JSON object with parameters for API call and store in a variable
      var requestOptions = {
        method: 'POST',
        headers: myHeaders,
        body: raw,
        redirect: 'follow'
      };
      // make API call with parameters and use promises to get response
      fetch("YOUR-API-INVOKE-URL", requestOptions)
      .then(response => response.text())
      .then(result => alert(JSON.parse(result).message))
      .catch(error => console.log('error', error));
    }
  </script>
</head>
<body>
  <form>
```

```
    <label>First Name :</label>
    <input type="text" id="fName">
    <label>Last Name :</label>
    <input type="text" id="lName">
    <!-- set button onClick method to call function we defined passing input values as parameters -->
    <button type="button"
onclick="callAPI(document.getElementById('fName').value,document.getElementById('lName').value)">Call
API</button>
  </form>
</body>
</html>
```

- Make sure you add your API Invoke URL on Line 41 (from module three). Note: If you do not have your API's URL, you can get it from the API Gateway console by selecting your API and choosing stages.
- Save the file.
- ZIP (compress) only the HTML file.
- Open the Amplify console.
- Choose the web app created in module one.
- Choose the white Choose files button.
- Select the ZIP file you created in Step 5.
- When the file is uploaded, a deployment process will automatically begin. Once you see a greenbar, your deployment will be complete.




AWS Amplify
All apps
GettingStarted
App settings
General
Amplify Studio settings
Domain management
Notifications
Access control
Monitoring
Rewrites and redirects
Custom headers
Documentation
Support
All apps > GettingStarted > App settings: Domain management
Domain management
Add domain
Use your own custom domain with free HTTPS to provide a secure, friendly URL for your app. Register your domain on Amazon Route53 for a one-click setup, or connect any domain registered on a 3rd party provider.
amplifyapp.com
Domain
amplifyapp.com
Status
Available
URL
Branch
Redirects to
https://dev.d1c7yshpy5stfn.amplifyapp.com
dev
-