# Project 1: Serverless Photo Gallery Application using AWS Lambda, API Gateway,

# S3, DynamoDB and Cognito

**References:**

[1] A. Bahga, V. Madisetti, “Cloud Computing Solutions Architect: A Hands-On Approach”, ISBN: 978-0996025591

[2] <https://aws.amazon.com/documentation/>

**Due Date:**

The lab report will be **due on \_\_\_\_\_\_\_\_.**

This lab is about creating a serverless Photo Gallery application.

The application uses the following -

1. A web interface implemented in HTML & JS which can be served through S3 static website hosting

2. AWS API Gateway endpoints

3. Lambda functions

4. Cognito for user pool management

5. DynamoDB for storing records of photos

6. S3 for storing photos

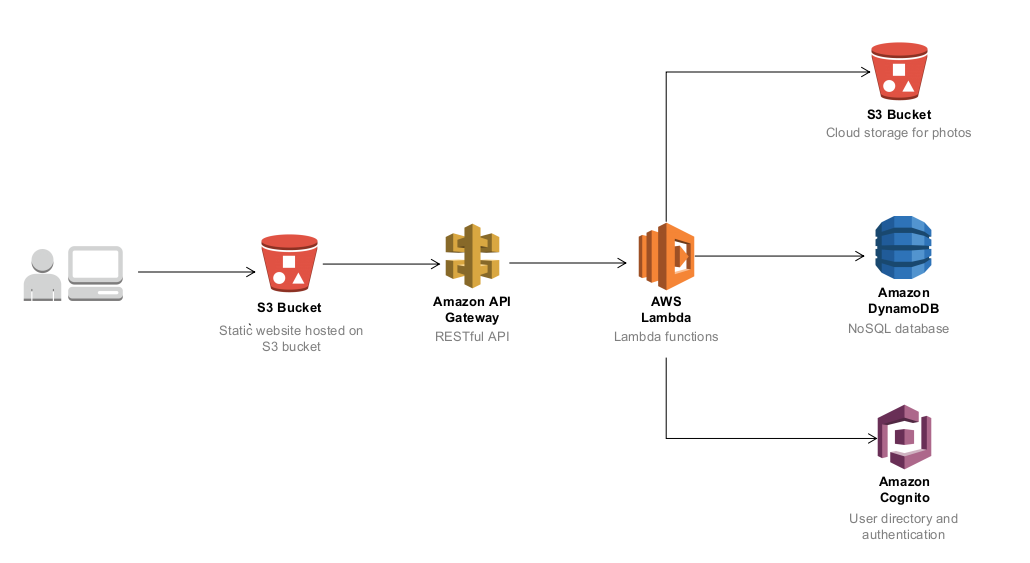


Fig. Architecture diagram of the Photo Gallery application showing AWS services used

Follow the steps below to setup the Photo Gallery application in your AWS account.

**1. Create an S3 Bucket for hosting the static website for the application**

* Create a new S3 bucket for hosting the static website and enable static website hosting for the bucket.
* Upload the files from the ‘staticwebsite’ folder in the attached files with this lab to the S3 bucket.

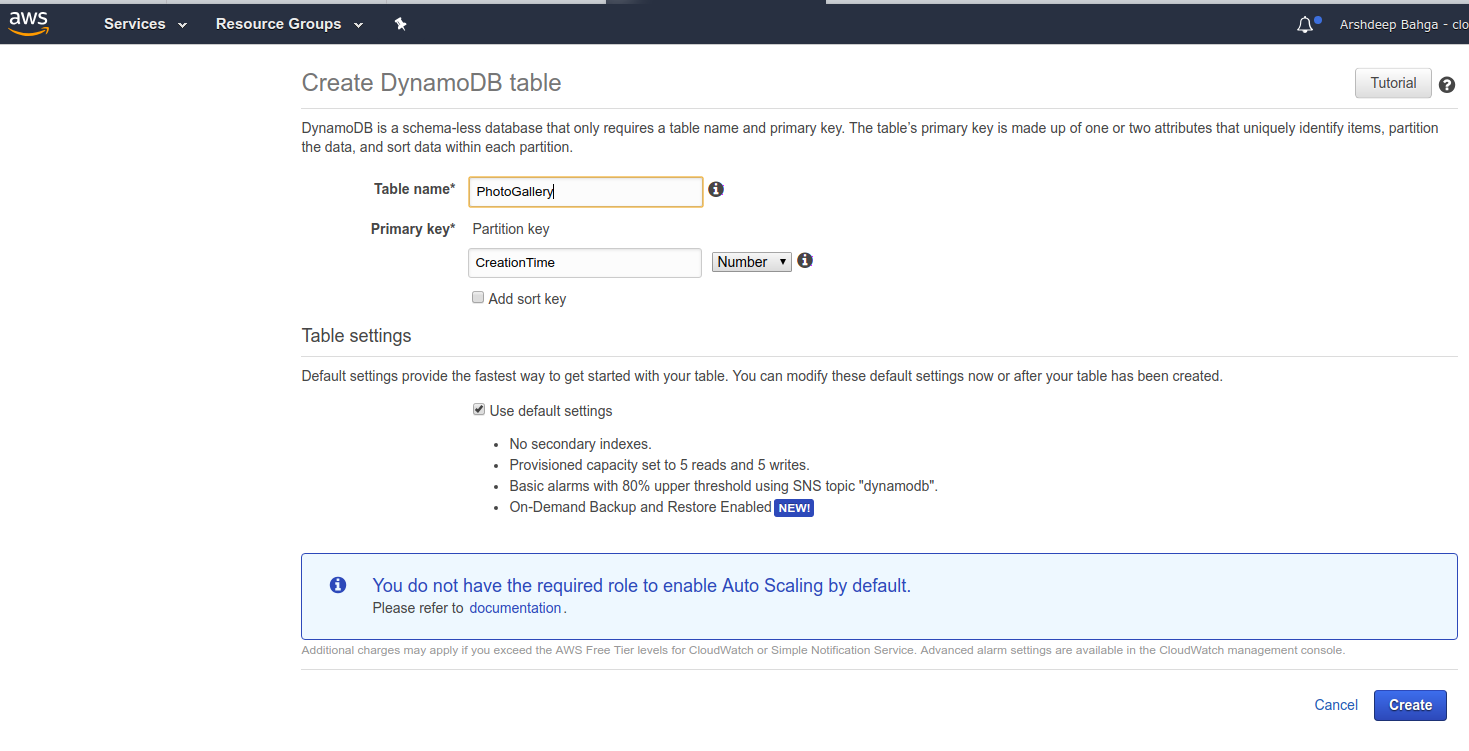
**2. Create an S3 Bucket for storing photos**

* Create a new S3 bucket for storing photos.
* Create a folder named ‘photos’ in this bucket.
* Add a bucket policy below to enable public access to the photos uploaded. Replace ‘mybucketname’ with the name of the S3 bucket created.

|  |
| --- |
| {  "Version": "2012-10-17",  "Id": "Policy1538026169421",  "Statement": [  {  "Sid": "Stmt1538026165732",  "Effect": "Allow",  "Principal": "\*",  "Action": "s3:GetObject",  "Resource": "arn:aws:s3:::mybucketname/photos/\*"  }  ]  } |

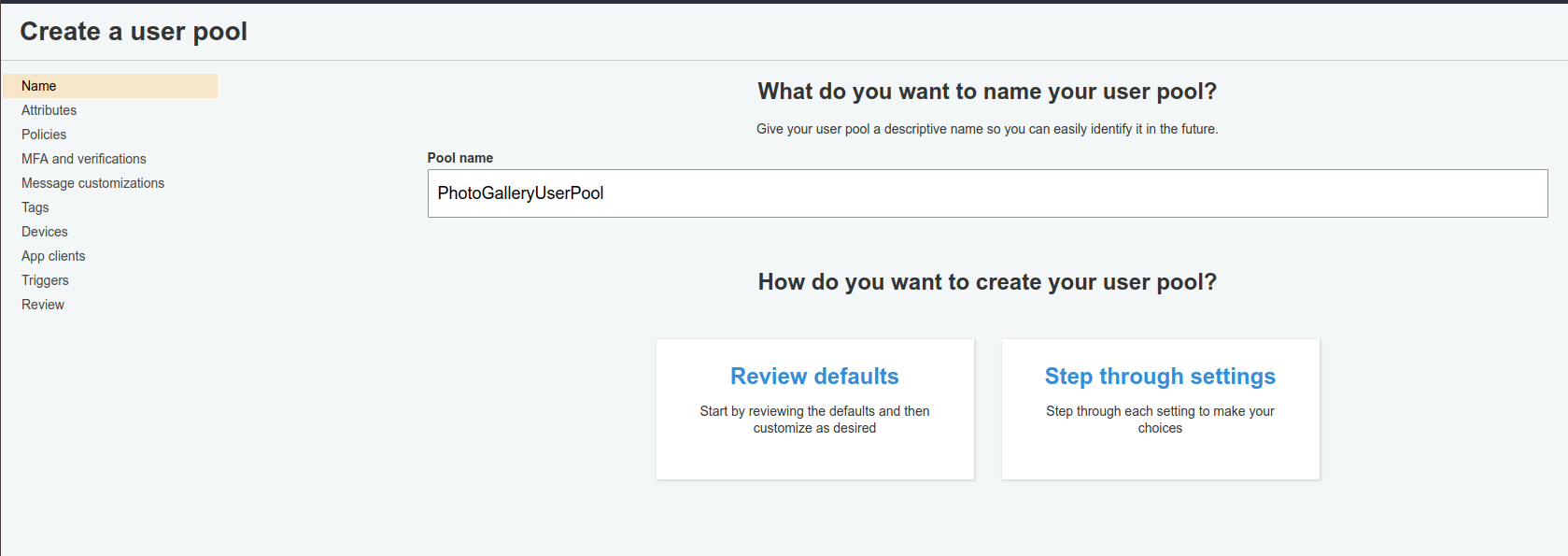
1. **Create a DynamoDB table for storing records of photos**

* Create a DynamoDB table named ‘PhotoGallery’ with sort key named ‘CreationTime’ as shown below.

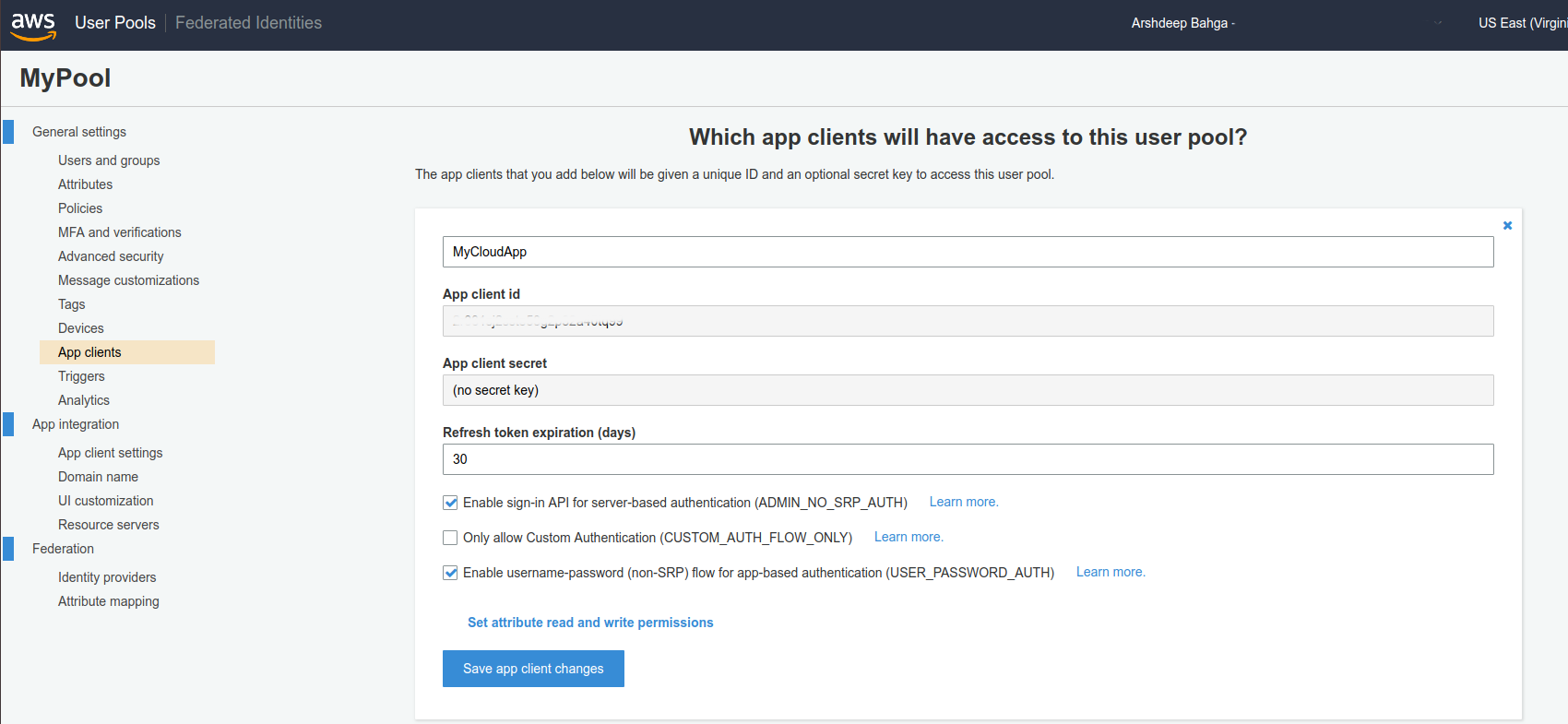


**4. Create Cognito User Pool**

* Navigate to Cognito console at: <https://console.aws.amazon.com/cognito/users/>
* Create a new User Pool as shown below.

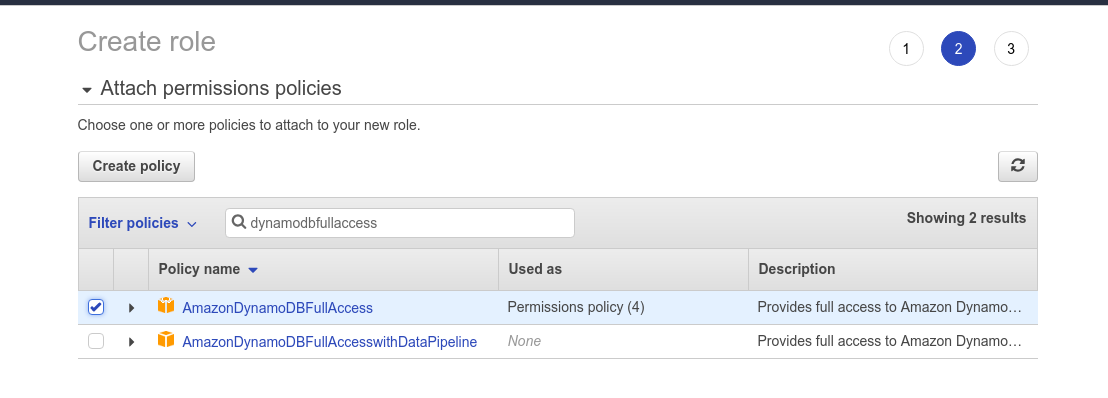
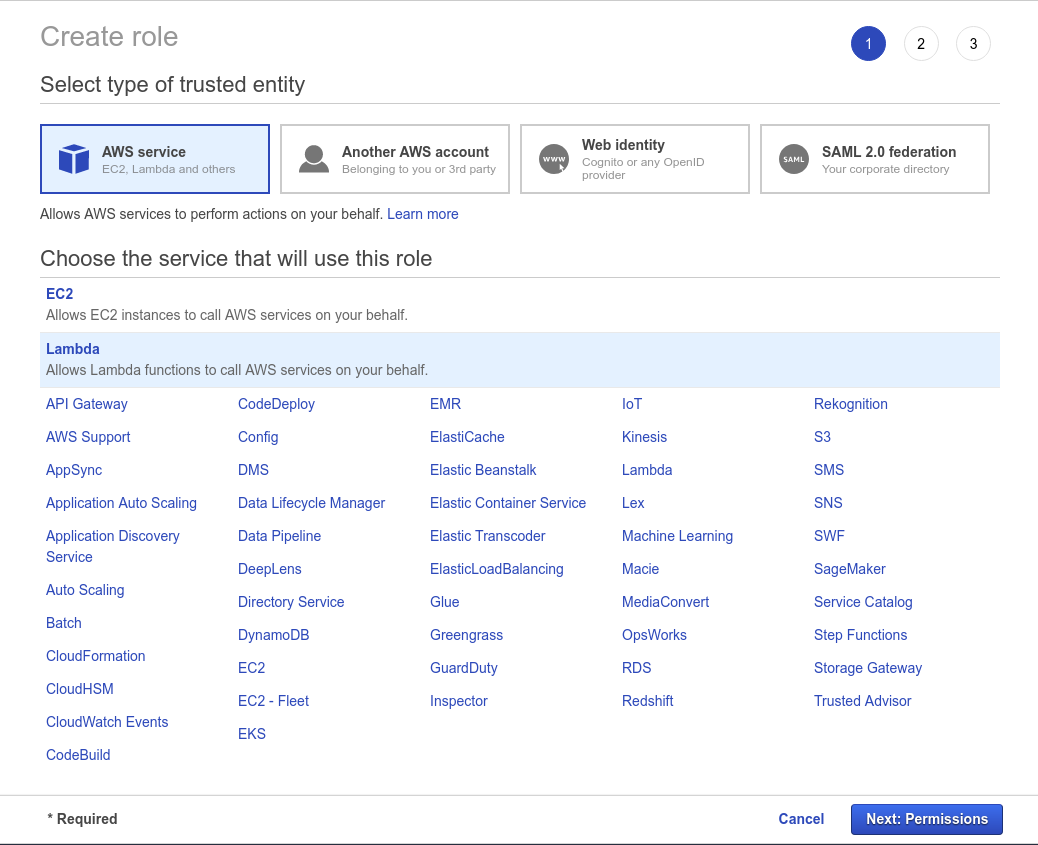


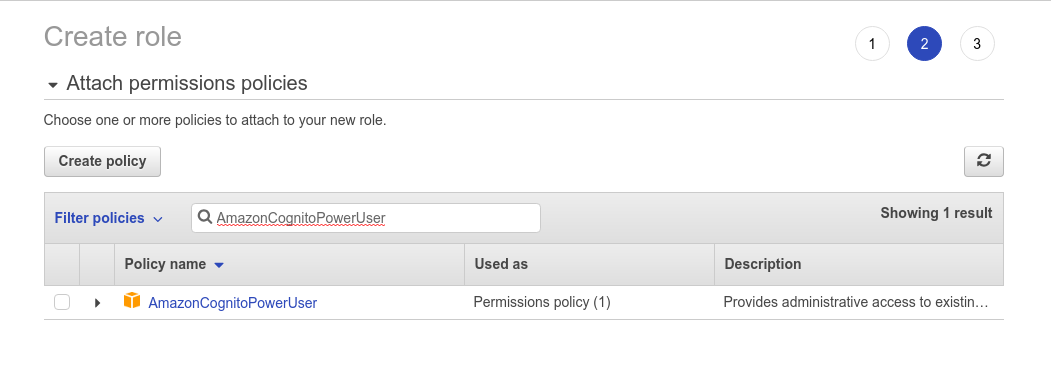
Once the User Pool is created, create a new App client for the user pool as shown below. Note down the App client ID.



**5. Create IAM Roles**

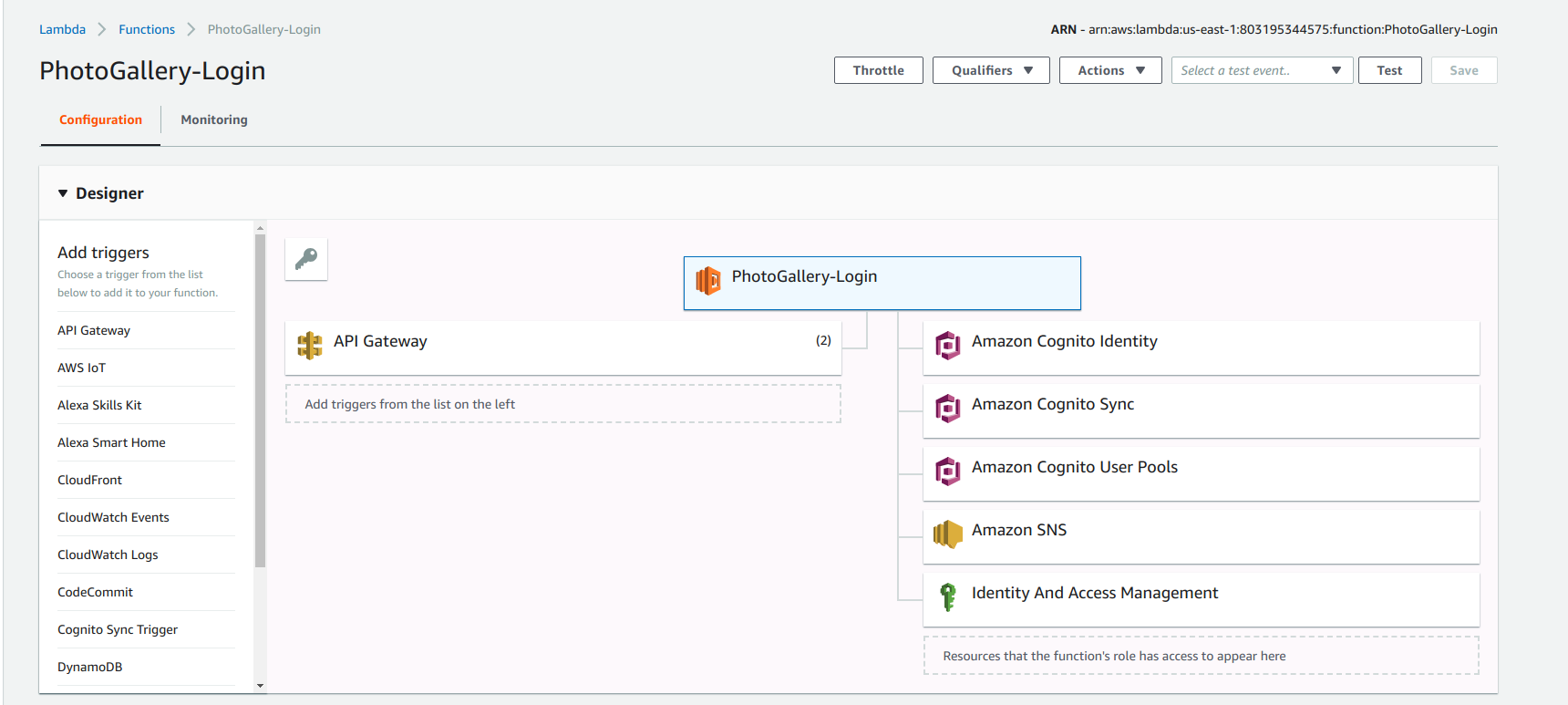
* From AWS IAM console create a new IAM role named ‘lambda\_photogallery\_role’ for Lambda service as shown below. Attach policy named AmazonDynamoDBFullAccess to this role. Then attach policy named AmazonCognitoPowerUser to this role.

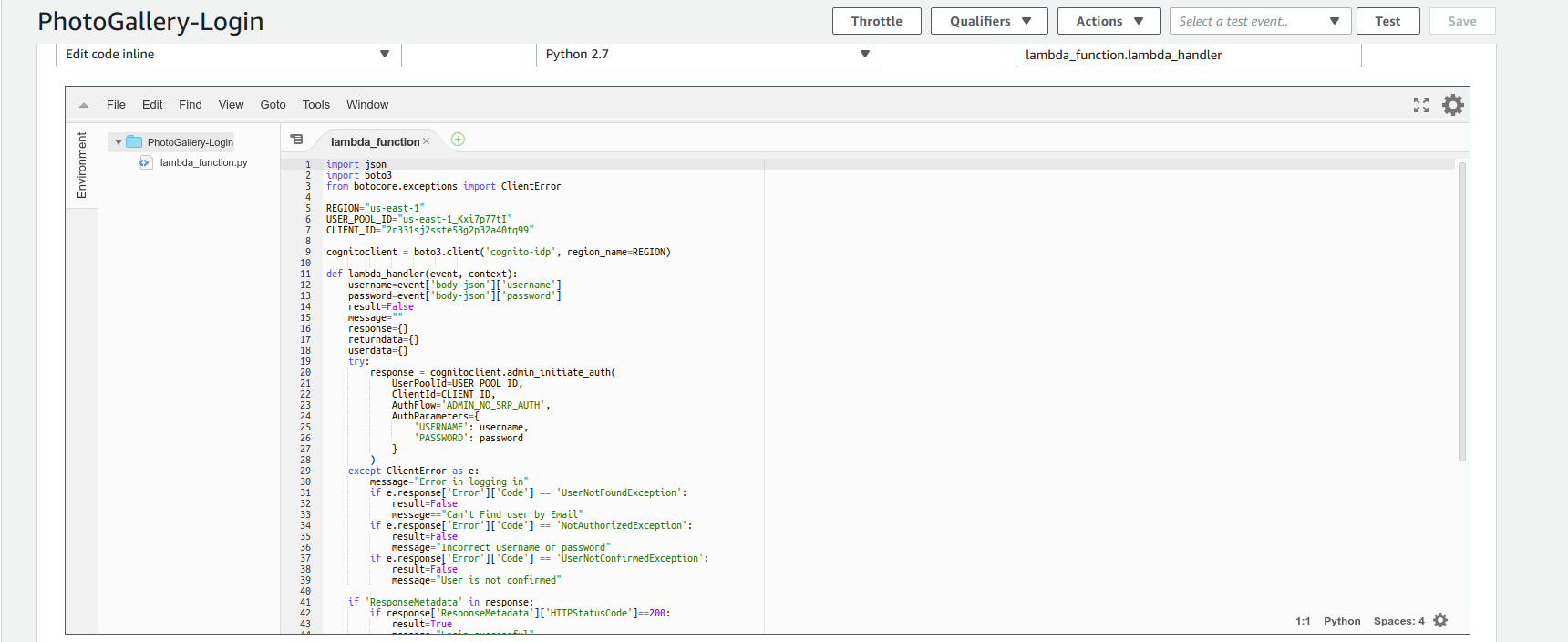


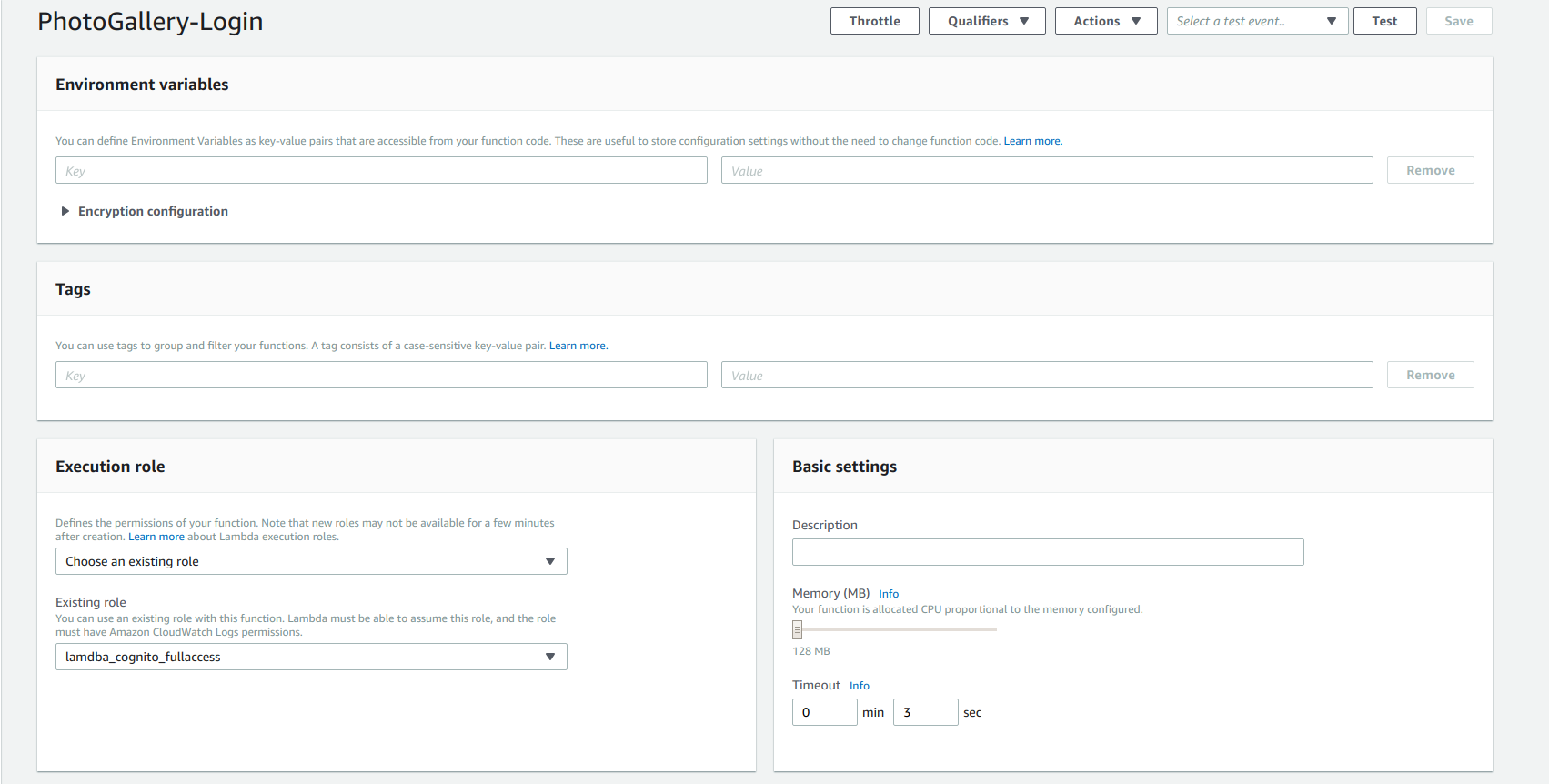


**6. Create Lambda functions**

* From AWS Lambda console create Lambda function for user signup (using source code in signup.py file provided).
* From AWS Lambda console create Lambda function for confirming user email after signup (using source code in confirmemail.py file provided).
* From AWS Lambda console create Lambda function for user login (using source code in login.py file provided).
* From AWS Lambda console create Lambda function for getting details of all photos (using source code in getphotos.py file provided).
* From AWS Lambda console create Lambda function for getting details of a specific photo (using source code in getphoto.py file provided).
* From AWS Lambda console create Lambda function for adding a photos (using source code in addphoto.py file provided).
* From AWS Lambda console create Lambda function for searching photos (using source code in search.py file provided).

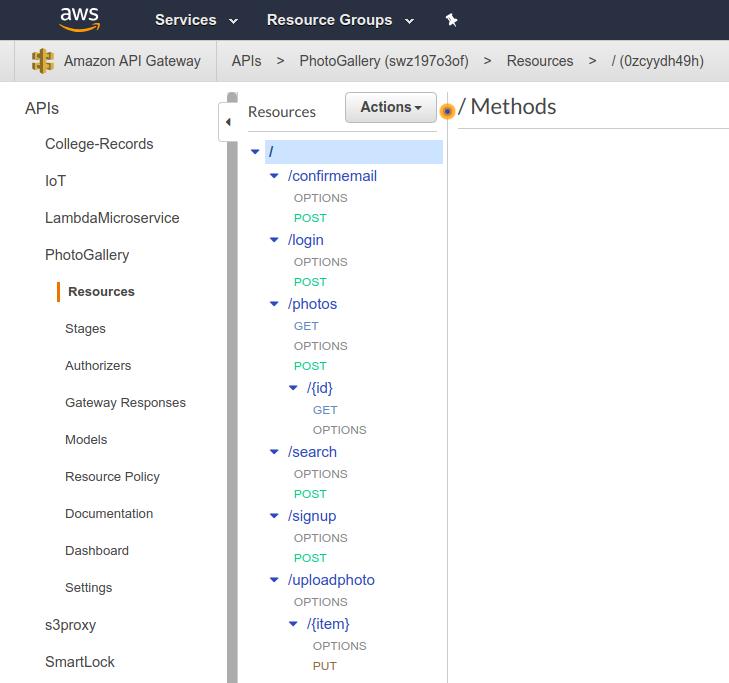






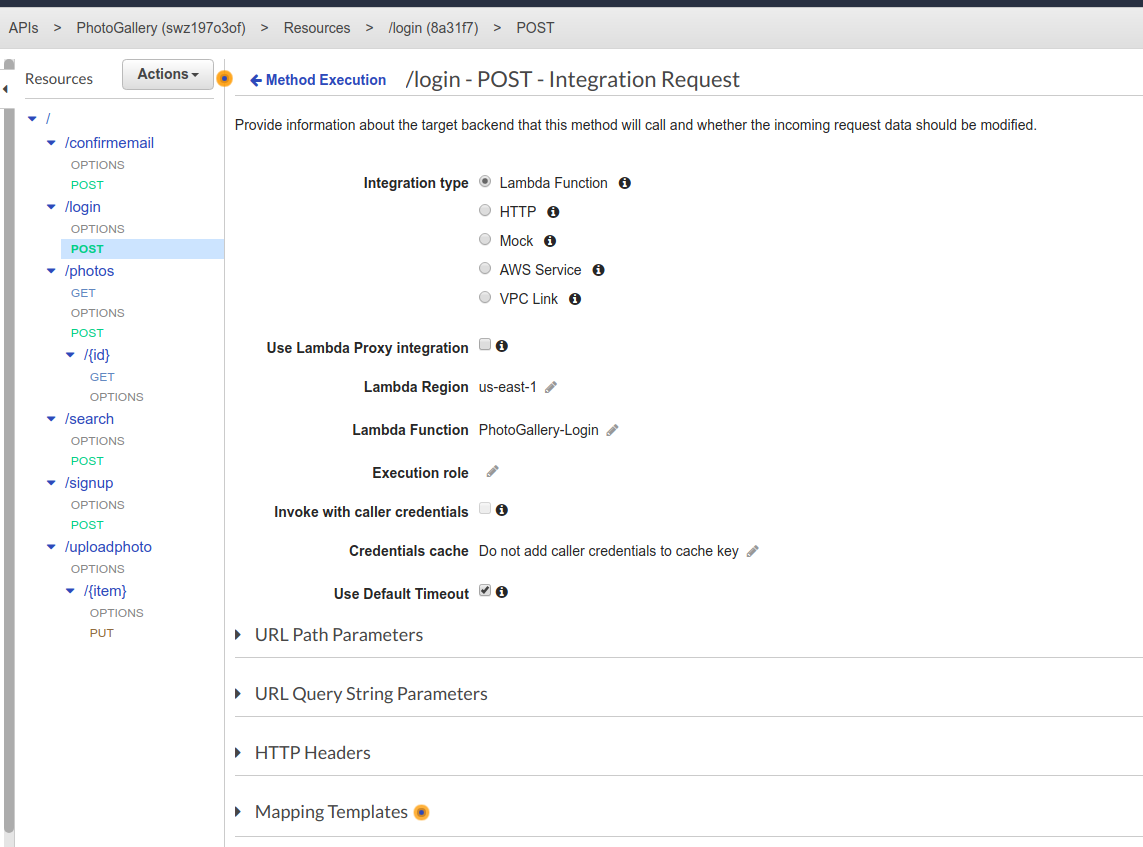
**7. Create a new API from API Gateway**

* Create a new API named ‘PhotoGallery’ from API Gateway console with resources and methods as shown below.

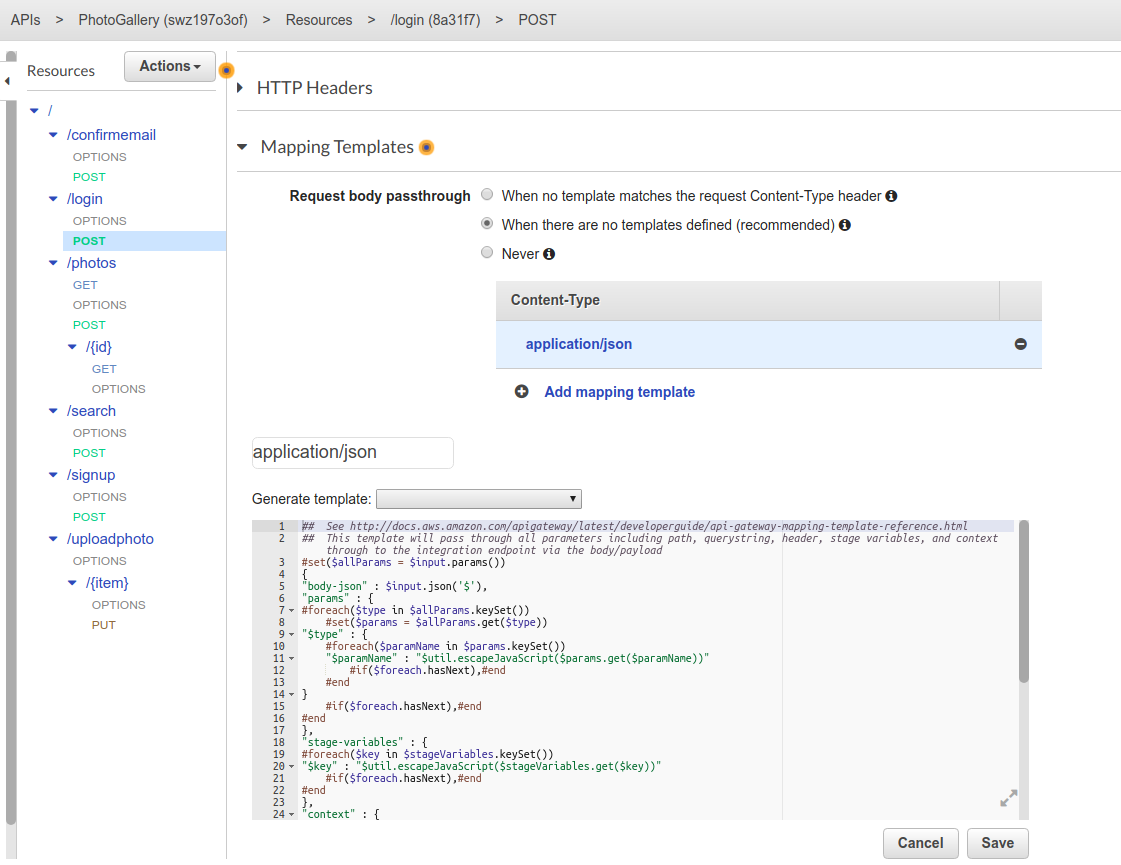


|  |  |  |
| --- | --- | --- |
| **Resource** | **Method** | **Integration Request** |
| /login | POST | Lambda function for signup |
| /signup | POST | Lambda function for signup |
| /confirmemail | POST | Lambda function for confirm email |
| /photos | GET | Lambda function for getting photos |
| /photos | POST | Lambda function for adding a photo |
| /photos/{id} | GET | Lambda function for getting details of a specific photo |
| /uploadphoto/{item} | PUT | AWS Service - S3 |
| /search | POST | Lambda function for searching photos |

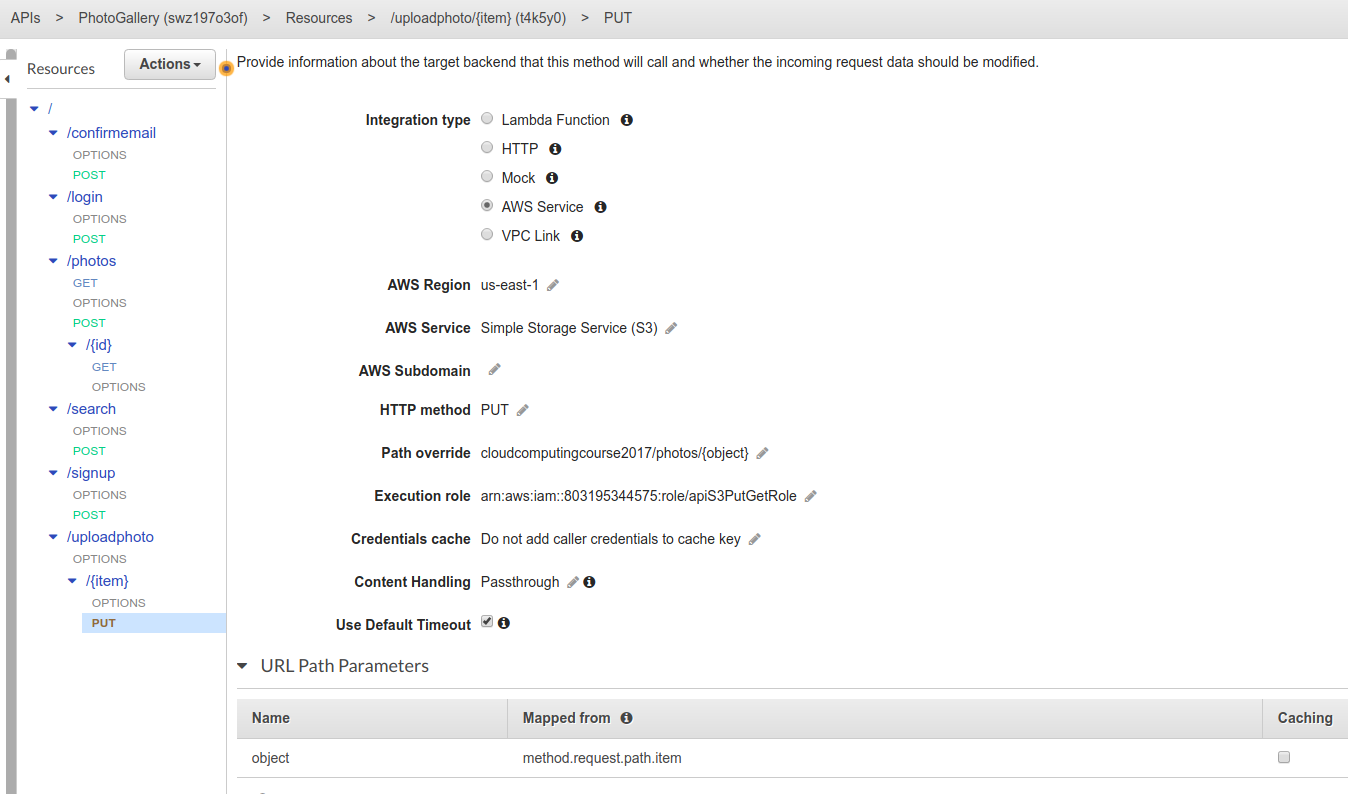
Use following settings for creating POST methods for login, signup, confirmemail and photos resources.



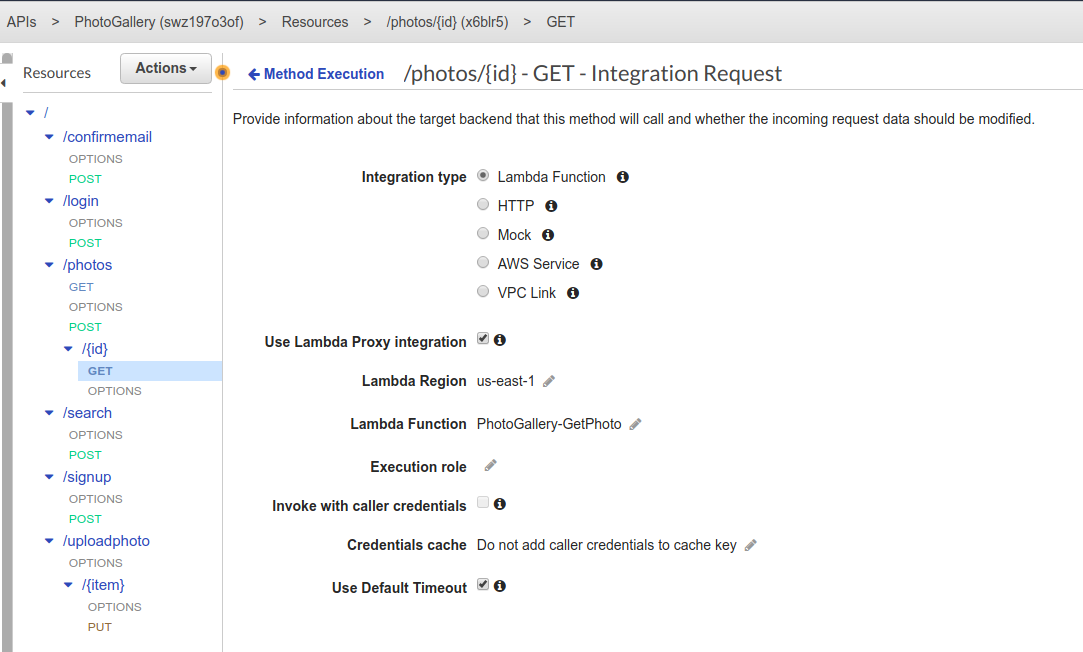
Add mapping template for content type application/json as shown below.



Use following settings for creating PUT method for /uploadphoto/{item} resource. In path override replace ‘cloudcomputhingcourse2017’ with the name of the bucket you created for storing photos. Add URL path parameter named ‘object’ as shown.



Use following settings for creating GET method for /photos/{id} resource.



**8. Access the Photo Gallery application in a browser**

* Access the URL of the static website for photo gallery application hosted on S3.
* Goto signup page and create a new user as shown in demo video.
* Confirm the user’s email as shown in demo video
* Goto login page and login using the user created above.
* Goto add photos page and add a new photo as shown in demo video.
* Add a more photos and try browsing and searching for photos.

**PROJECT 1 Submission:**

1. Watch the video and complete the steps (except ‘search’ photo functionality) – 80%
2. Write a lambda function (preferably in Python) to search photos – 20%