

# Lab 01

---

## GEORGIA INSTITUTE OF TECHNOLOGY SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

---

ECE 4150-A Fall 2024

**Lab: Serverless Photo Gallery Application using AWS Lambda, API Gateway, S3, DynamoDB and Cognito**

---

**Professor:** Dr. Vijay Madisetti (vkm@gatech.edu)

**TA:** Samdup Choephel (schoephel3@gatech.edu)

Tuesday & Wednesday 12:00-13:00

Location: Will be announce in advance on CatchUp

### References:

- A. Bahga, V. Madisetti, "Cloud Computing Solutions Architect: A Hands-On Approach", ISBN: 978-0996025591
- <https://aws.amazon.com/documentation/>

### Due Date:

The lab report will be **due on January 31, 2024**.

---

This lab is about creating a serverless **Photo Gallery** application. **Make sure your AWS region is set for us-east-1 or N. Virginia (located at the top right of the navigation bar)**

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

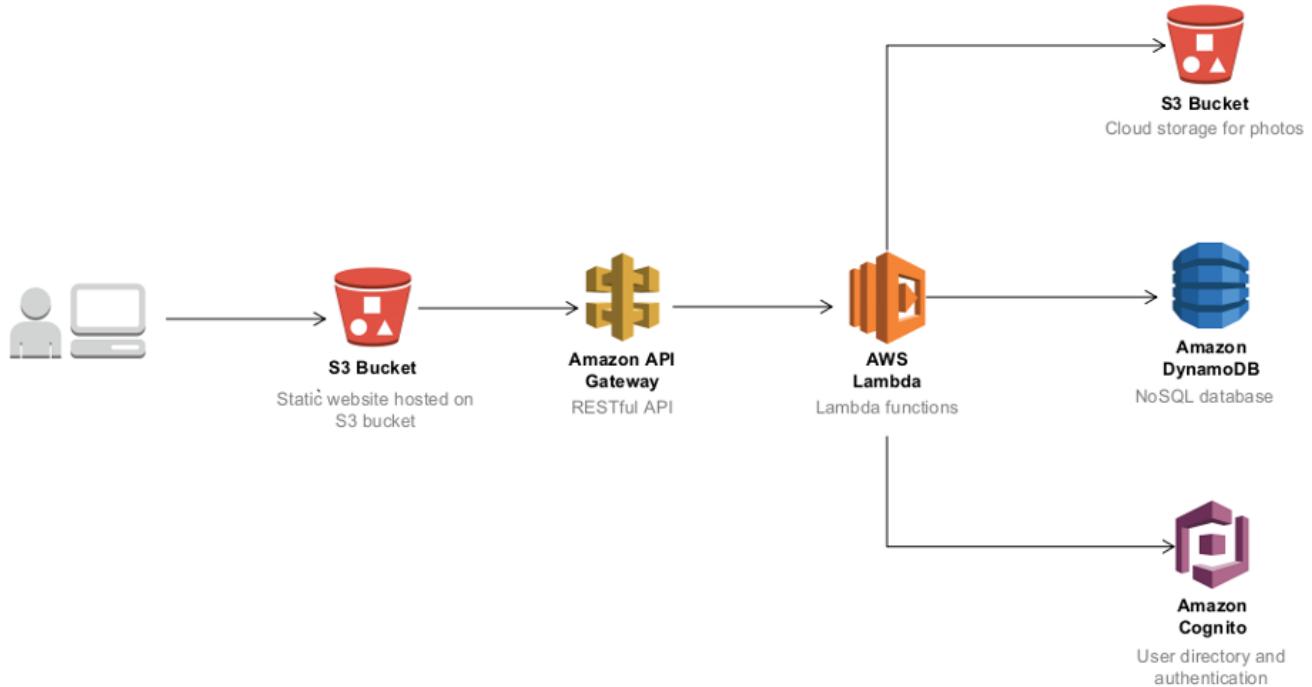


Fig. Architecture diagram of the Photo Gallery application showing AWS services used

Follow the steps below to set up 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. **Uncheck Block all public access under Bucket setting for Block Public Access section**
- The bucket name must be unique. Click here to know the rules. For the name of the bucket, use **staticwebsite-lastname-year-courseNumber** (e.g., **staticwebsite-choephel-2024-4150**)
- 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",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "S3:getObject",
            "Resource": "arn:aws:s3:::mybucketname/*"
        }
    ]
}
```

**Note:** You have to visit [here](#) again later to whitelist your static website

## 2. Create an S3 Bucket for storing photos

- Create a new S3 bucket for storing photos. **Uncheck Block all public access under Bucket setting for Block Public Access section**
- For the name of the bucket, use **photobucket-lastname-year-courseNumber** (e.g., **photobucket-choephel-2024-4150**)
- 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",  
    "Statement": [  
        {  
            "Sid": "PublicReadGetObject",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "S3:getObject",  
            "Resource": "arn:aws:s3:::mybucketname/photos/*"  
        }  
    ]  
}
```

## Use the following script Cross-origin resource sharing (CORS)

```
[  
    {  
        "AllowedHeaders": [  
            "Authorization"  
        ],  
        "AllowedMethods": [  
            "GET"  
        ],  
        "AllowedOrigins": [  
            "*"  
        ],  
        "ExposeHeaders": []  
    }  
]
```

## 3. Create a DynamoDB table for storing records of photos

- Create a DynamoDB table named **PhotoGallery** with a partition key and a sort key named **PhotoID** and **CreationTime**, respectively, as shown below.



## Create table

### Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

#### Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.)

#### Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

String

1 to 255 characters and case sensitive.

#### Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Number

1 to 255 characters and case sensitive.

### Table settings

#### Default settings

The fastest way to create your table. You can modify these settings now or after your table has been created.

#### Customize settings

Use these advanced features to make DynamoDB work better for your needs.

### Default table settings

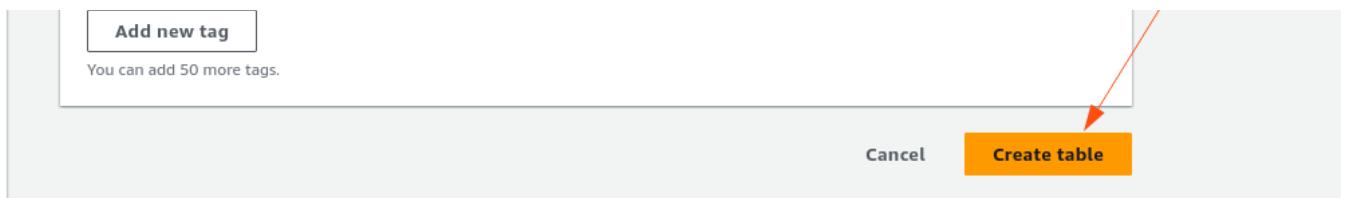
These are the default settings for your new table. You can change some of these settings after creating the table.

Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes

### Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.



#### 4. Create Cognito User Pool (You can open Cognito through the search box)

- Navigate to the AWS Cognito console
- Create a new **User Pool** called **PhotoGalleryUserPool** as shown below.

The screenshot shows the "Configure sign-in experience" step of creating a user pool. On the left, a sidebar lists steps 1 through 6. Step 1 is "Configure sign-in experience", which is currently selected and expanded. Step 2 is "Configure security requirements", Step 3 is "Configure sign-up experience", Step 4 is "Configure message delivery", Step 5 is "Integrate your app", and Step 6 is "Review and create".

**Configure sign-in experience** Info

Your app users can sign in to your user pool with a user name and password, or sign in with a third-party identity provider.

**Authentication providers**

Configure the providers that are available to users when they sign in.

**Provider types**

Choose whether users will sign in to your Cognito user pool, a federated identity provider, or both. Amazon Cognito has different pricing for federated users and user pool users. [Learn more about pricing](#)

**Cognito user pool**  
Users can sign in using their email address, phone number, or user name. User attributes, group memberships, and security settings will be stored and configured in your user pool.

**Federated Identity providers**  
Users can sign in using credentials from social identity providers like Facebook, Google, Amazon, and Apple; or using credentials from external directories through SAML or Open ID Connect. You can manage user attribute mappings and security for federated users in your user pool.

**Cognito user pool sign-in options** Info

Choose the attributes in your user pool that are used to sign in. If you select only one attribute, or you select a user name and at least one other attribute, your user can sign in with all of the selected options. If you select only phone number and email, your user will be prompted to select one of the two sign-in options when they sign up.

**User name**  
 **Email**  
 **Phone number**

**User name requirements**

**Allow users to sign in with a preferred user name**  
 **Make user name case sensitive**

**⚠️** Cognito user pool sign-in options can't be changed after the user pool has been created.

**Cancel** **Next**

aws Services Search [Alt+S] N. Virginia sam.cloud

Amazon Cognito > User pools > Create user pool

Step 1 Configure sign-in experience

Step 2 Configure security requirements

Step 3 Configure sign-up experience

Step 4 Configure message delivery

Step 5 Integrate your app

Step 6 Review and create

## Configure security requirements Info

Set up a strong password requirement in addition to multi-factor authentication to protect your app users from accidentally compromising their credentials.

### Password policy Info

Create a password policy to define the length and complexity of the passwords your users can set.

Password policy mode Info

Cognito defaults Use default password requirements.

Custom Use password requirements that you define.

Password minimum length  
8 character(s)

Password requirements

Contains at least 1 number  
Contains at least 1 special character  
Contains at least 1 uppercase letter  
Contains at least 1 lowercase letter

Temporary passwords set by administrators expire in  
7 day(s)

## Multi-factor authentication

Configure secure access to your app by enforcing multi-factor authentication (MFA) during the user sign-in process. MFA settings are applied to all app clients.

MFA enforcement Info

Require MFA - Recommended Users must provide an additional authentication factor when signing in.

Optional MFA Users can sign in with a single authentication factor, and can choose to add additional authentication factors.

No MFA Users can only sign in with a single authentication factor. This is the least secure option.

## User account recovery

Configure how users will recover their account when they forget their password. Recipient message and data rates apply.

### Self-service account recovery Info

Enable self-service account recovery - Recommended

Allow forgot-password operations in your user pool. In the hosted UI sign-in page, a "Forgot your password?" link is displayed. When this feature is not enabled, administrators reset passwords with the Cognito API.

### Delivery method for user account recovery messages Info

Select how your user pool will deliver messages when users request an account recovery code. SMS messages are charged separately by Amazon SNS. Email messages are charged separately by Amazon SES. [Learn more about pricing](#).

Email only

SMS only

Email if available, otherwise SMS

SMS if available, otherwise email

SMS if available, otherwise email, and allow a user to reset their password via SMS if they are also using it for MFA

Cancel

Previous

Next

aws Services Search [Alt+S] N. Virginia sam.cloud

Amazon Cognito > User pools > Create user pool

Step 1 Configure sign-in experience

Step 2 Configure security requirements

Step 3 Configure sign-up experience

Step 4 Configure message delivery

Step 5 Integrate your app

Step 6 Review and create

## Configure sign-up experience Info

Determine how new users will verify their identities when signing up and which attributes should be required or optional during the user sign-up flow.

### Self-service sign-up Info

Choose whether new users of your app can register for an account themselves.

**Self-registration Info**

**Enable self-registration**  
Display a "Sign up" link on the sign-in page in the hosted UI, and allow the use of public APIs to create new user accounts. When this feature is not enabled, federation and administrative API operations create user profiles.

ⓘ If you activate user sign-up in your user pool, anyone on the internet can sign up for an account and sign in to your apps. Don't enable self-registration in your user pool until you want to open your app to public sign-up. [Learn more](#)

### Attribute verification and user account confirmation

Choose between Cognito-assisted and self-managed user attribute verification and account confirmation. Only verified attributes can be used for sign-in, account recovery, and MFA. A user account must be confirmed either by attribute verification, or user pool administrator confirmation, before a user is allowed to sign in.

#### Cognito-assisted verification and confirmation Info

**Allow Cognito to automatically send messages to verify and confirm - Recommended**  
Cognito sends a verification message with a code that the user must enter. For new users, this will verify the attribute and confirm their account. When this feature is not enabled, administrative API operations and Lambda triggers verify and confirm users.

**Attributes to verify Info**  
Choose the user contact attribute that Cognito will send a verification message to. Recipient message and data rates apply when you use SMS.

**Send SMS message, verify phone number**  
Verify with SMS to allow users to use their phone number for sign-in, MFA, and account recovery. SMS messages are charged separately by Amazon SNS.

**Send email message, verify email address**  
Verify with email to allow users to use their email address for sign-in, MFA, and account recovery. Email messages are charged separately by Amazon SES.

**Send SMS message if phone number is available, otherwise send email message**  
You must build custom code when you want to verify both email and phone numbers at user account creation.

#### Verifying attribute changes Info

**Keep original attribute value active when an update is pending - Recommended**  
When you update the value of an email or phone number attribute, your user must verify the new value. Until they verify the new value, they can receive messages and sign in with the original value. If you don't turn on this feature, your user can't sign in with that attribute before they verify the new value.

**Active attribute values when an update is pending Info**  
Choose the attributes that you want to keep active when an update to their value is pending. Your users can receive messages and sign in with the original attribute value until they verify the new value.

**Email address**

### Required attributes Info

Choose the attributes that are required when a new user is created. Cognito assigns all users a set of standard attributes based on the OpenID Connect (OIDC) standard.

Required attributes based on previous selections  
**email**

Additional required attributes

⚠ Required attributes can't be changed once this user pool has been created.

**Custom attributes - optional**  
Personalize the sign-up experience by adding up to 50 custom attributes. Custom attribute names can't be changed after a user pool has been created.

[Cancel](#)[Previous](#)[Next](#)

aws | Services |  [Alt+S]

Amazon Cognito > User pools > Create user pool

Step 1 [Configure sign-in experience](#)

Step 2 [Configure security requirements](#)

Step 3 [Configure sign-up experience](#)

Step 4 [Configure message delivery](#)

Step 5 Integrate your app

Step 6 Review and create

## Configure message delivery Info

Amazon Cognito uses Amazon SES and Amazon SNS to send email and SMS messages to your app users. Messages may incur additional SES and SNS costs.

### Email

Configure how your user pool sends email messages to users.

Email provider [Info](#)

**Send email with Amazon SES - Recommended**  
Send emails using an Amazon SES verified identity in your account. We recommend this option for higher email volume and production workloads.

**Send email with Cognito**  
Use Cognito's default email address as a temporary start for development. You can use it to send up to 50 emails a day.

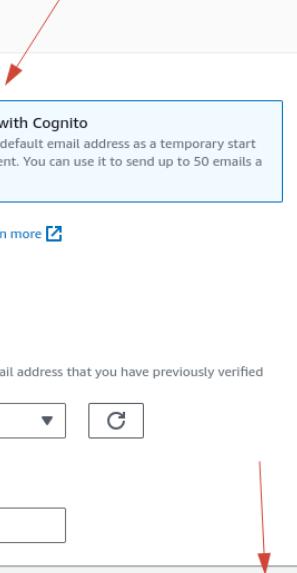
You must have configured a verified sender with [Amazon SES](#) to use the SES feature. [Learn more](#)

SES Region [Info](#)  
**US East (Ohio)**

FROM email address [Info](#)  
By default "no-reply@verificationemail.com" will be used. You can also choose a different email address that you have previously verified with Amazon SES.  
 [C](#)

REPLY-TO email address - optional [Info](#)  
If you set an invalid reply-to address, sending restrictions may be imposed on your account.

[Cancel](#) [Previous](#) [Next](#)



AWS Services Search [Alt+S]

Amazon Cognito > User pools > Create user pool

Step 1 Configure sign-in experience

Step 2 Configure security requirements

Step 3 Configure sign-up experience

Step 4 Configure message delivery

Step 5 Integrate your app

Step 6 Review and create

## Integrate your app Info

Set up app integration for your user pool with Cognito's built-in authentication and authorization flows.

### User pool name

Create a friendly name for your user pool.

User pool name  User pool names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + = . , @ -

**⚠ Your user pool name can't be changed once this user pool is created.**

### Hosted authentication pages

Choose whether to use Cognito's Hosted UI and OAuth 2.0 server for user sign-up and sign-in flows.

Use the Cognito Hosted UI Build hosted sign-up, sign-in, and OAuth 2.0 service endpoints in Amazon Cognito. When this feature is not enabled, use Cognito API operations to perform sign-up and sign-in.

### Initial app client

Configure an app client. App clients are single-app platforms in your user pool that have permissions to call unauthenticated API operations. A user pool can have multiple app clients.

#### App type Info

Select an app type and we will automatically populate common default settings. You can add additional app clients after the user pool is created.

**Public client** A native, browser or mobile-device app. Cognito API requests are made from user systems that are not trusted with a client secret.

**Confidential client** A server-side application that can securely store a client secret. Cognito API requests are made from a central server.

**Other** A custom app. Choose your own grant, auth flow, and client-secret settings.

#### App client name Info

Enter a friendly name for your app client.

MyCloudApp App client names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + = . , @ -

**⚠ You cannot change or remove a client secret after you allow Amazon Cognito to generate it for your app client.**

#### Client secret Info

Choose whether your app client will have a client secret. Client secrets are used by the server-side component of an app to authorize API requests. Using a client secret can prevent a third party from impersonating your client.

Generate a client secret

**Don't generate a client secret**

**► Advanced app client settings** We have populated suggested authentication flows, OAuth 2.0 grant types, and OIDC scopes based on the selections you made earlier.

App client name | [Info](#)  
Enter a friendly name for your app client.

Client secret | [Info](#)  
Choose whether your app client will have a client secret. Client secrets are used by the server-side component of an app to authorize API requests. Using a client secret can prevent a third party from impersonating your client.

Generate a client secret  
 Don't generate a client secret

**⚠** You cannot change or remove a client secret after you allow Amazon Cognito to generate it for your app client.

**▼ Advanced app client settings**  
We have populated suggested authentication flows, OAuth 2.0 grant types, and OIDC scopes based on the selections you made earlier.

**Authentication flows** | [Info](#)  
Choose authentication flows that your app will support. Refresh token authentication is always enabled. We have populated options based on your app type.

Select authentication flows

ALLOW\_REFRESH\_TOKEN\_AUTH X Refresh token based authentication  
ALLOW\_USER\_SRP\_AUTH X SRP (secure remote password) protocol based authentication

ALLOW\_ADMIN\_USER\_PASSWORD\_AUTH X Username password auth for admin APIs for authentication  
ALLOW\_CUSTOM\_AUTH X Lambda trigger based custom authentication

ALLOW\_USER\_PASSWORD\_AUTH X User name and password authentication

Authentication flow session duration | [Info](#)  
3 minutes  
Must be between 3 and 15 minutes.

Refresh token expiration | [Info](#)  
30 days 0 minutes  
Must be between 60 minutes and 10 years.

Enter “App client name” as **MyCloudApp**. Click on **Next**.

Go through the **Review and Create** carefully as the step is irreversible.

Create the User Pool by clicking “**Create Pool**”.

After creating the pool, save the **Pool ID** for later; we will use this identifier for our serverless function.

Amazon Cognito | [Services](#) | [Search](#) | [Alt+S]

Amazon Cognito > User pools > PhotoGalleryUserPool

**PhotoGalleryUserPool** | [Info](#) | [Delete user pool](#)

**User pool overview**

User pool name: PhotoGalleryUserPool  
User pool ID: **us-east-2\_OW8m9YlQh** (highlighted with a red box)  
ARN: arn:aws:cognito-idp:us-east-2:975050089819:userpool/us-east-2\_OW8m9YlQh  
Estimated number of users: 0

Created time: January 17, 2024 at 10:19 EST  
Last updated time: January 17, 2024 at 10:19 EST

**Getting started**

Users | Groups | Sign-In experience | **Sign-up experience** | Messaging | **App Integration** (highlighted with a red box) | User pool properties

**Configuration for all app clients**  
Domain and resource server settings for the user pool. All app clients that enable the Hosted UI use the user pool domain. All app clients can authorize access to user pool resource servers.

Save the values you get from 2,3 and 5 to somewhere like a notepad

**Domain Info**  
Configure a domain for your Hosted UI and OAuth 2.0 endpoints. You must choose a domain if you need Cognito to create Hosted UI authentication endpoints. If you have an existing domain, you must delete it before assigning a new one.

**Cognito domain**  
Domain

**Custom domain**  
Domain

**Actions ▾**

**Resource servers (0) Info**  
Configure resource servers. A resource server is a remote server that authorizes access based on OAuth 2.0 scopes in an access token.

**Search resource servers by name or ID**

**Resource server name** ▲ **Resource server identifier** ▾ **Custom scopes**

No resource servers

**Create resource server**

**App client defaults**  
Hosted UI customization and advanced security settings for the user pool. You can customize the Hosted UI and advanced security in app clients to override the defaults.

**Hosted UI customization Info**  
Customize the hosted sign-up and sign-in pages to match your app's style and branding by uploading your own logo and customized CSS.

**Edit**

**Logo** | **Custom CSS**

**Advanced security Info**  
Configure advanced security features, including Cognito's automatic responses to suspicious user activity. Advanced security adds cost to your bill. [See pricing](#)

**Status**  **Disabled**

**Enable**

**App client list**  
The app clients that integrate your apps with your user pool. Configure client overrides to user pool default configurations, and configure Amazon Pinpoint analytics.

**App clients and analytics (1) Info**  
Configure an app client. App clients are the user pool authentication resources attached to your app. Select an app client to configure the permitted authentication actions for an app.

**Search app clients by name or ID**

**App client name** ▾ **Client ID** ▾

**MyCloudApp** | **6jj4i884qns9fc13khf03u50r** 5

From **App Clients and analytics**, Under the new app client (**MyCloudApp**), copy and save the **App client id**; we will use this identifier for our serverless function.

## 5. Create IAM Roles

- In the AWS IAM console, goto roles and create a new IAM role called '**lambda\_photogallery\_role**' for Lambda service, as shown below. Attach policy called **AmazonDynamoDBFullAccess** to this role. Then attach policy called **AmazonCognitoPowerUser** to this role.

**Select trusted entity**

**Trusted entity type**

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**  
Lambda

Choose a use case for the specified service.  
Use case  
 Lambda Allows Lambda functions to call AWS services on your behalf.

**Next**

**Add permissions**

**Permissions policies (1/907)**

Choose one or more policies to attach to your new role.

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonCognitoPowerUser	AWS managed	Provides administrative access to existin...
<input type="checkbox"/> AmazonEC2ContainerRegistryPowerUser	AWS managed	Provides full access to Amazon ECR Cont...
<input type="checkbox"/> AmazonElasticContainerRegistryPublicPowerUser	AWS managed	Provides full access to Amazon ECR Publ...
<input type="checkbox"/> AWSCodeCommitPowerUser	AWS managed	Provides full access to AWS CodeCommit...
<input type="checkbox"/> AWSDataPipeline_PowerUser	AWS managed	Provides full access to Data Pipeline, list ...
<input type="checkbox"/> AWSKeyManagementServicePowerUser	AWS managed	Provides access to AWS Key Managemen...
<input type="checkbox"/> PowerUserAccess	AWS managed - job function	Provides full access to AWS services and ...

**Set permissions boundary - optional**

**Next**

Screenshot of the AWS IAM 'Create role' wizard.

**Step 1: Select trusted entity**

**Role details**

**Role name:** lambda\_photogallery\_role

**Description:** Allows Lambda functions to call AWS services on your behalf.

**Step 1: Select trusted entities**

**Trust policy:**

```
1 [ { "Version": "2012-10-17", "Statement": [ 2 { "Effect": "Allow", "Action": [ 3 "sts:AssumeRole" ], "Principal": { 4 "Service": [ 5 "lambda.amazonaws.com" ] } } ] } ]
```

**Step 2: Add permissions**

**Permissions policy summary**

Policy name	Type	Attached as
<a href="#">AmazonCognitoPowerUser</a>	AWS managed	Permissions policy

**Step 3: Add tags**

**Add tags - optional**

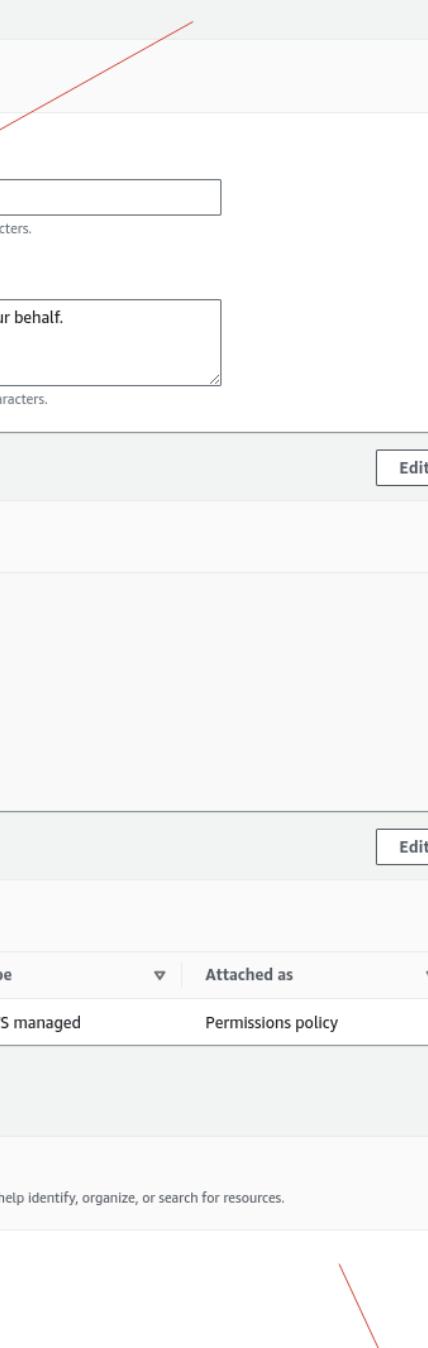
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

**Add new tag**

You can add up to 50 more tags.

**Create role**



Screenshot of the AWS IAM Role Management interface showing the 'lambda\_photogallery\_role' configuration.

**Summary**

Creation date: January 17, 2024, 10:34 (UTC-05:00)

Last activity: -

ARN: arn:aws:iam::975050089819:role/lambda\_photogallery\_role

Maximum session duration: 1 hour

**Permissions**

Permissions policies (1) [Info](#)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name: AmazonCognitoPowerUser

Type: AWS managed

Attached entities: 1

**Permissions boundary** (not set)

**Generate policy based on CloudTrail events**

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

Generate policy

No requests to generate a policy in the past 7 days.

A red arrow points to the 'Attach policies' button in the 'Permissions policies' section.

A red arrow points to the 'Add permissions' button at the top right of the page.

The right sidebar shows a list of notebooks:

- File Edit View Go
- NOTEBOOKS +
- All notes 0 3
- 2022 TSP 4
- AD
- Backup Solution
- Bash Cheat sheet
- Bash for cyber sec
- Bash Training 5
- Binary Exploitation
- BMC 3
- Bug Bounty Note
- CEH Practical 15
- cheatsheet 2
- Conflicts (attachm
- CyberSec 2
- DNS network setu
- Docker 3
- Documentation fo
- Fixing VMWare Iss
- Free Labs 1
- GATECH NOTES
- AMA Assignme
- CCN Corner 8
- CS-6260-A Cryp

Created remote items: 2  
Updated remote items: 1  
Completed: 17/01/2024 10:35 (9s)

Synchronize

Screenshot of the 'Add permissions' dialog for the 'lambda\_photogallery\_role'.

Attach policy to lambda\_photogallery\_role

Current permissions policies (1)

Other permissions policies (1/906)

Filter by Type: All types

Search: AmazonDynamoDBFullAccess

Policy name: AmazonDynamoDBFullAccess

Type: AWS managed

Description: Provides full access to Amazon DynamoD...

Cancel Add permissions

A red arrow points to the search bar in the 'Other permissions policies' section.

A red arrow points to the 'Add permissions' button at the bottom right.

aws | Services | Search [Alt+S] | Global ▾ | cloudtenzin ▾

**Identity and Access Management (IAM)**

IAM > Roles > lambda\_photogallery\_role

### lambda\_photogallery\_role Info

Allows Lambda functions to call AWS services on your behalf.

**Edit** | **Delete**

Summary	
Creation date	January 17, 2024, 10:34 (UTC-05:00)
ARN	arn:aws:iam::975050089819:role/lambda_photogallery_role
Last activity	-
Maximum session duration	1 hour

**Permissions** | Trust relationships | Tags | Access Advisor | Revoke sessions

**Permissions policies (2) Info**

You can attach up to 10 managed policies.

Filter by Type

Policy name	Type	Attached entities
AmazonCognitoPowerUser	AWS managed	1
AmazonDynamoDBFullAccess	AWS managed	1

**Permissions boundary (not set)**

**Generate policy based on CloudTrail events**

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

**Generate policy**

No requests to generate a policy in the past 7 days.

- Create another IAM role called '**apiS3PutGet-Role**' to upload the photos to the S3 Bucket service, as shown below. Attach policy called **AmazonAPIGatewayPushToCloudWatchLogs** to this role. Then attach policy called **AmazonS3FullAccess** to this role.

## 6. Review Code

- Under the `lambda-functions` directory, open `login.py`, `signup.py`, and `confirmemail.py`; update the `<USER_POOL_ID>` and the `<CLIENT_ID>`, in lines 6 and 7, respectively, with the ones you just created.

```

FOLDERS          < >    login.py
lab-files
  api-gateway-definition
  lambda-functions
    addphoto.py
    confirmemail.py
    getphoto.py
    getphotos.py
    login.py
    search.py
    signup.py
  staticwebsite

1 import json
2 import boto3
3 from botocore.exceptions import ClientError
4
5 REGION="us-east-1"
6 USER_POOL_ID=""
7 CLIENT_ID=""
8
9 cognitoclient = boto3.client('cognito-idp', region_name=REGION)
10
11 def lambda_handler(event, context):
12     username=event['body-json']['username']
13     password=event['body-json']['password']
14     result=False

```

```

FOLDERS          < >    login.py           &nbsp; signup.py
lab-files
  api-gateway-definition
  lambda-functions
    addphoto.py
    confirmemail.py
    getphoto.py
    getphotos.py
    login.py
    search.py
    signup.py
  staticwebsite

1 import json
2 import boto3
3 from botocore.exceptions import ClientError
4
5 REGION="us-east-1"
6 USER_POOL_ID=""
7 CLIENT_ID=""
8
9 cognitoclient = boto3.client('cognito-idp', region_name=REGION)
10
11 def lambda_handler(event, context):
12     username=event['body-json']['username']
13     password=event['body-json']['password']
14     name=event['body-json']['name']
     result=False

```

```

1 import json
2 import boto3
3 from botocore.exceptions import ClientError
4
5 REGION="us-east-1"
6 USER_POOL_ID=<USER_POOL_ID>
7 CLIENT_ID=<CLIENT_ID>
8
9 cognitoclient = boto3.client('cognito-identity', region_name=REGION)
10
11 def lambda_handler(event, context):
12     username=event['body-json']['username']
13     code=event['body-json']['code']
14     result=False

```

## 7. Create Lambda functions

- We are going to create seven serverless functions under the AWS Lambda console. Each function follow the same process to deploy them. Please, follow each of the screenshots below.

### A. User signup (using source code in signup.py file provided):

- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery\_signup**), attached the role previously created (**lambda\_photogallery\_role**) to this function, and then click “**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.

**Basic information**

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

Runtime Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.  
Python 3.8

Architecture Choose the instruction set architecture you want for your function code.  
x86\_64

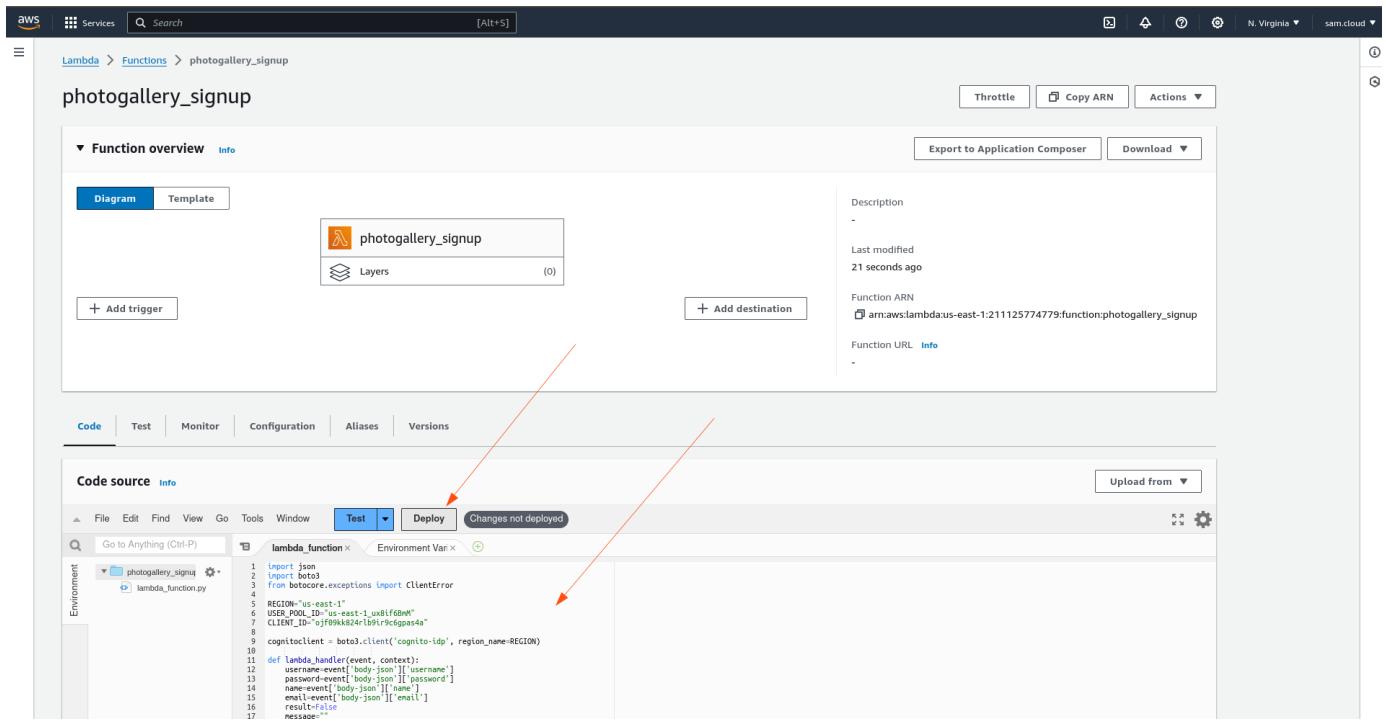
Permissions By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

**Execution role** Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

Use an existing role  
 Create a new role with basic Lambda permissions  
 Create a new role from AWS policy templates

Existing role Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
lambda\_photogallery\_role

- After creating the function, under “Function Code” copy and paste the code provided.



- Click “Deploy” to release the function

**Follow the same image examples as above.**

B. Confirming user email after signup (using source code in confirmemail.py file provided):

- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery\_confirmemail**), attached the role previously created (**lambda\_photogallery\_role**) to this function, and then click “**Create function**”
- After creating the function, under “**Function Code**” copy and paste the code provided.
- Click “**Deploy**” to release the function

C. User login (using source code in login.py file provided):

- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery\_login**), attached the role previously created (**lambda\_photogallery\_role**) to this function, and then click “**Create function**”
- After creating the function, under “**Function Code**” copy and paste the code provided.
- Click “**Deploy**” to release the function

D. Getting details of all photos (using source code in getphotos.py file provided):

- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery\_getphotos**), attached the role previously created (**lambda\_photogallery\_role**) to this function, and then click “**Create function**”
- After creating the function, under “**Function Code**” copy and paste the code provided.
- Click “**Deploy**” to release the function

E. Getting details of a specific photo (using source code in getphoto.py file provided):

- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery\_getphoto**), attached the role previously created (**lambda\_photogallery\_role**) to this function, and then click “**Create function**”
- After creating the function, under “**Function Code**” copy and paste the code provided.
- Click “**Deploy**” to release the function

F. Adding a photo (using source code in addphoto.py file provided):

- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery\_addphoto**), attached the role previously created (**lambda\_photogallery\_role**) to this function, and then click “**Create function**”
- After creating the function, under “**Function Code**” copy and paste the code provided.
- Click “**Deploy**” to release the function

G. Searching photos (using source code in search.py file provided):

- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery\_search**), attached the role previously created (**lambda\_photogallery\_role**) to this function, and then click “**Create function**”
- After creating the function, under “**Function Code**” copy and paste the code provided.
- Click “**Deploy**” to release the function

## 8. Create a new API from API Gateway

- Go to the AWS API Gateway Console. Once you enter the console, it will provide you with an option to create an Example API with some resources and method; disregard that option and select “**New API**”. Create a new API called **PhotoGalleryAPI** with the resources, methods, and integration shown below.

AWS Services Search [Alt+S]

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**  
PhotoGalleryAPI

**Description - optional**

**API endpoint type**  
Regional

Cancel **Create API**

You have to prepare resources and methods for these items.

Resource	Method	Integration Request
/signup	POST	Lambda Function: photogallery_signup
/confirmemail	POST	Lambda Function: photogallery_confirmemail
/login	POST	Lambda Function: photogallery_login
/photos	POST	Lambda Function: photogallery_addphoto
/photos	GET	Lambda Function: photogallery_getphotos
/photos/{id}	GET	Lambda Function: photogallery_getphoto
/uploadphoto	PUT	S3 Bucket: photobucket-corporan-2021-4813
/search	POST	Lambda Function: photogallery_search

For **/signup**, a POST request a lambda function:

- Create a new Resource

Resource details

Proxy resource [Info](#)  
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path: /

Resource name: signup

CORS (Cross Origin Resource Sharing) [Info](#)  
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel Create resource

- Give a name to the resource and make sure that “**Enable API Gateway CORS**” is checked
- Once the resource is created, select the resource and create a new Method under that resource.

API Gateway

APIs  
Custom domain names  
VPC links

API: PhotoGalleryAPI  
Resources  
Stages  
Authorizers  
Gateway responses  
Models  
Resource policy  
Documentation  
Dashboard  
API settings

Usage plans  
API keys  
Client certificates  
Settings

Resources

Create resource

Resource details

Path: /signup  
Resource ID: ym1zdf

Method type	Integration type	Authorization
OPTIONS	Mock	None
POST	Lambda	None

1. select resource

2. create method

- Select POST as the method for this resource

Screenshot of the AWS API Gateway 'Create method' page. The page shows the 'Method details' section with various integration types. A red arrow points to the 'Method type' dropdown, with the text '1. choose POST or GET or PUT, mentioned in the walkthrough'. Another red arrow points to the 'arn:aws:lambda:us-east-1:211125774779:function:photogallery\_signup' field, with the text '2. Choose function name'. A third red arrow points to the 'Create method' button at the bottom right, with the text '3. Click here'.

Method details

Method type

Select a method type

Integration type

- Lambda function  
Integrate your API with a Lambda function.  

- HTTP  
Integrate with an existing HTTP endpoint.  

- Mock  
Generate a response based on API Gateway mappings and transformations.  


- AWS service  
Integrate with an AWS Service.  

- VPC link  
Integrate with a resource that isn't accessible over the public internet.  


Lambda proxy Integration  
Send the request to your Lambda function as a structured event.

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 ▾

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

Default timeout  
The default timeout is 29 seconds.

Cancel **Create method**

- Select the method created and attach the lambda function as the integration type.
- After saving the integration type, select “**Integration Request**” under the method. Under the “**Mapping Template**”, click “**Edit**” and “**Add Mapping Template**”. Add “**application/json**” to the mapping. Select “**Method Request passthrough**” and click “**Save**”.

aws Services Search [Alt+S] N. Virginia sam.cloud

### API Gateway

APIs Custom domain names VPC links

**API: PhotoGalleryAPI**

- Resources**
  - Stages
  - Authorizers
  - Gateway responses
  - Models
  - Resource policy
  - Documentation
  - Dashboard
  - API settings
- Usage plans
- API keys
- Client certificates
- Settings

**Resources**

Create resource / signup - POST - Method execution

ARN: arn:aws:execute-api:us-east-1:211125774779:sqses598ld/\* /POST/signup Resource ID: ym1zdf

Method request → Integration request → Integration response ← Method response ← Integration response

Client Lambda integration

Method request | **Integration request** | Integration response | Method response | Test

**Integration request settings**

Integration type: Lambda Region: us-east-1

Lambda proxy integration: False Lambda function: photogallery\_signup

Input passthrough: When there are no templates defined (recommended) Timeout: Default (29 seconds)

URL path parameters (0)

URL query string parameters (0)

HTTP headers (0)

Mapping templates (1)

application/json

1. Choose Integration request

2. click here to edit

aws Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Edit integration request

### Edit integration request

**Method details**

1.

Integration type:

- Lambda function**: Integrate your API with a Lambda function. (selected)
- HTTP**: Integrate with an existing HTTP endpoint.
- Mock**: Generate a response based on API Gateway mappings and transformations.

**AWS service**

**VPC link**

Integrate with an AWS Service.



Integrate with a resource that isn't accessible over the public internet.



Lambda proxy integration  
Send the request to your Lambda function as a structured event.

**Lambda function**  
Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 ▾ Q arn:aws:lambda:us-east-1:211125774779:function:phot X

**Execution role**  
arn:aws:iam::myAccount:role/myRole

**Credential cache**  
Do not add caller credentials to cache key ▾

Default timeout  
The default timeout is 29 seconds.

**Request body passthrough**

When there are no templates defined (recommended)  
 When no template matches the request content-type header  
 Never

**URL path parameters**

**URL query string parameters**

**URL request headers parameters**

**Mapping templates**

**Content type**  
application/json ▾ Remove

**Generate template**  
Method request passthrough ▾

**Template body**

```

1 ## See https://docs.aws.amazon.com/apigateway/latest
   /developerguide/api-gateway-mapping-template-reference
   .html
2 ## This template will pass through all parameters
   including path, querystring, header, stage variables,
   and context through to the integration endpoint via the
   body/payload
3 #set($allParams = $input.params())
4 {
5   "body-json" : $input.json('$'),
6   "params" : {
7     #foreach($type in $allParams.keySet())
8       #set($params = $allParams.get($type))
9       "$type" : {
10         #foreach($paramName in $params.keySet())
11           "#set($paramName" : "$util.escapeJavaScript($params.get
12             ($paramName))"
13           "#if($foreach.hasNext),#end
14         #end
15       "#if($foreach.hasNext),#end
16     #end
17   },
18   "stage-variables" : {

```

Use VTL templates to create your mapping template. [Learn more](#)

Add mapping template

Cancel **Save**

2.  
3.  
4.  
5.  
6.  
7.

aws Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598Id) > Edit method response

## Edit method response Info

**Response details**

HTTP status code **1**  
200

Header name **2**  
Header name  
Access-Control-Allow-Origin

Add header

Remove

Response body **3**

Content type  
application/json

Model  
Empty

Add model

Remove

Cancel **4** Save

This screenshot shows the 'Edit method response' dialog in the AWS API Gateway console. It displays settings for a 200 HTTP status code response. The 'Header name' 'Access-Control-Allow-Origin' is listed. The 'Content type' is set to 'application/json'. The 'Model' dropdown is set to 'Empty'. At the bottom, there are 'Cancel' and 'Save' buttons, with a red arrow pointing to the 'Save' button.

aws Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598Id) > Edit integration response

## Edit integration response Info

**Response details**

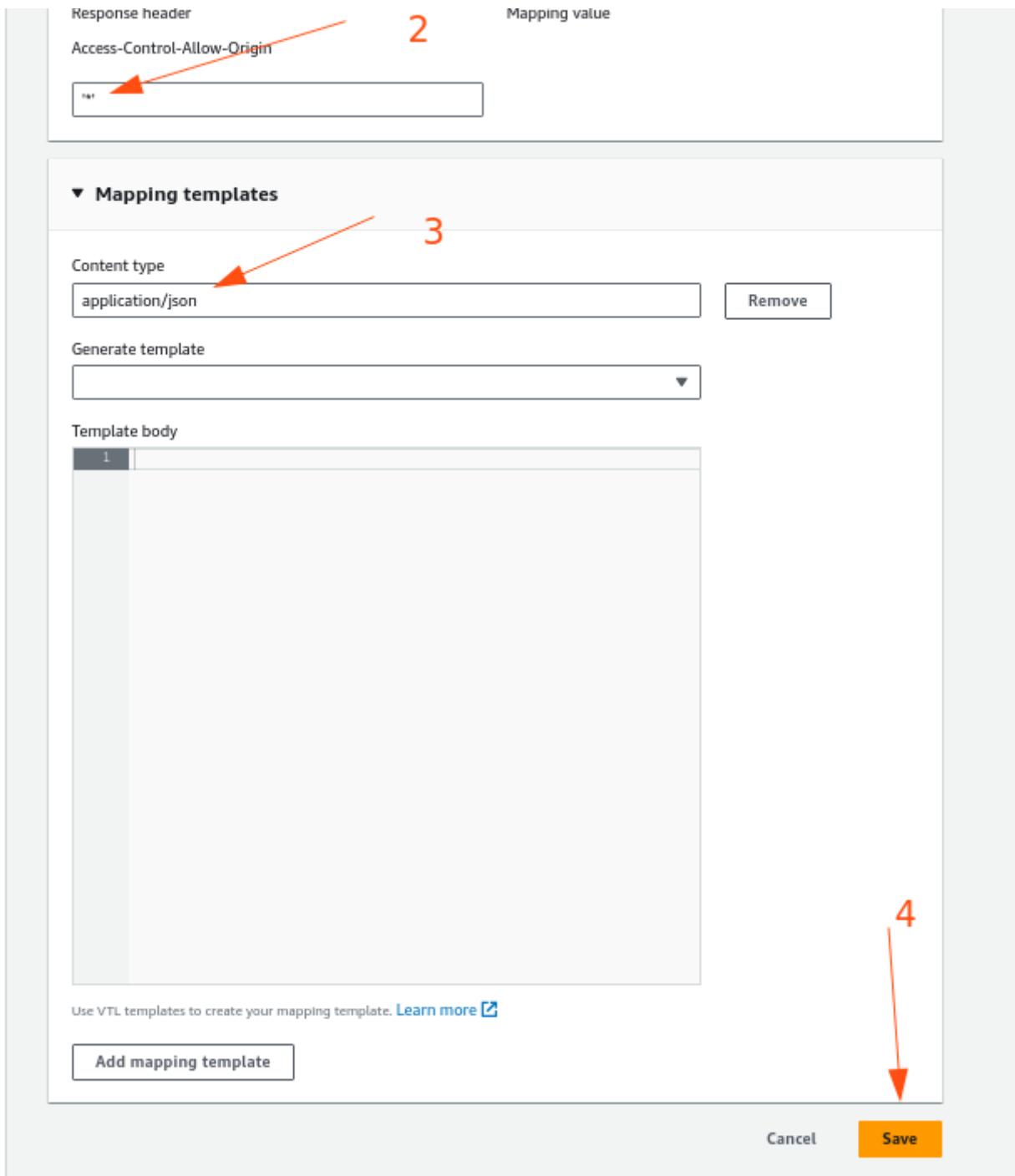
Lambda error regex | Info

Method response status code **1**  
200

Content handling | Learn more ↗  
Passthrough

Header mappings Info

This screenshot shows the 'Edit integration response' dialog in the AWS API Gateway console. It displays settings for a 200 method response. The 'Content handling' dropdown is set to 'Passthrough'. At the bottom, there is a 'Header mappings' section.



To create another resource, click “/“ under the Resources column

Repeat the same process for /confirmemail, /login, and /search

For / , a GET request a lambda function:

- Create a new **Resource** under the **Actions** dropdown button. Give a name to the resource and make sure that “**Enable API Gateway CORS**” is checked. Set the mapping template for all the method created below.

aws Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Create resource

## Create resource

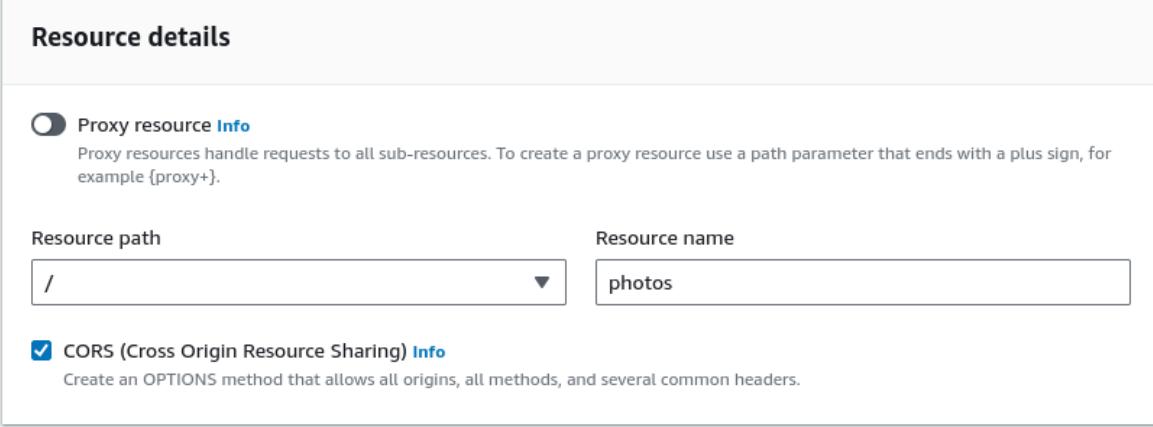
**Resource details**

Proxy resource [Info](#)  
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path: / Resource name: photos

CORS (Cross Origin Resource Sharing) [Info](#)  
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel **Create resource**



- Once the resource is created, select the resource and create a new **Method** under that resource.
- Select **GET** as the method for this resource. Select the method created and attach the lambda function as the integration type. (**getphotos**)

aws services Search [Alt+S] N. Virginia sam.c

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Edit integration request

## Edit integration request

Method details
1

Lambda function
 HTTP
 Mock

AWS service
 VPC link

Lambda proxy integration
2 **arn:aws:lambda:us-east-1:211125774779:function:photogallery\_getphotos**

Send the request to your Lambda function as a structured event.

Lambda function
3

Provide the Lambda function name or alias. You can also provide an ARN from another account.
  X

Execution role

Credential cache

Default timeout
The default timeout is 29 seconds.

Request body passthrough
4

When there are no templates defined (recommended)
5

When no template matches the request content-type header
6

Never
7

▶ URL path parameters
8

▶ URL query string parameters
9

▶ URL request headers parameters
10

▼ Mapping templates
11

Content type
application/json
12

Generate template
13

Method request passthrough
14

```

1 ## See https://docs.aws.amazon.com/apigateway/latest
   /developerguide/api-gateway-mapping-template-reference
   .html
2 ## This template will pass through all parameters
   including path, querystring, header, stage variables,
   and context through to the integration endpoint via the
   body/payload
3 #set($allParams = $input.params())
4 {
5   "body-json" : $input.json('$'),
6   "params" : {
7     #foreach($type in $allParams.keySet())
8       #set($param = $allParams.get($type))
9     #if($param.key == 'stageVariables')
10      #foreach($paramName in $param.value.keySet())
11        #set($paramName = "$util.escapeJavaScript($param.value.get($paramName))")
12        #if($foreach.hasNext),#end
13      #end
14    #if($foreach.hasNext),#end
15    #end
16  },
17  "stage-variables" : (
18    #foreach($paramName in $param.value.keySet())
19      #set($paramName = "$util.escapeJavaScript($param.value.get($paramName))")
20      #if($foreach.hasNext),#end
21    #end
22  )
23 }

```

Use VTL templates to create your mapping template. Learn more
15

Add mapping template
16

Cancel
Save
17

- Under the /photos resource, create a new **POST** method. Select the method created and attach the lambda function as the integration type. (**addphoto**)
- After saving the integration type, select “**Integration Request**” under both get and post methods. Under the “**Mapping Template**”, choose “**When there are no templates defined (recommended)**” under the Request body passthrough. Under the **Content-Type**, click “**Add mapping**” and add “**application/json**” to the mapping. Finally, select “**Method Request passthrough**” and click “**Save**”.
- Under the /photos resource, create a new resource for **/photos/{id}**, as shown below. Under the new created resource, create a new **GET** method. Select the method created and attach the lambda function (getphoto) as the integration type.

The screenshot shows the AWS API Gateway interface for creating a new resource. The top navigation bar includes the AWS logo, Services, a search bar, and a [Alt+S] key shortcut. Below the navigation, the breadcrumb trail shows: API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Create resource. The main title is "Create resource". A section titled "Resource details" contains the following fields:

- Proxy resource Info**: An unchecked radio button option.
- Proxy resource Info**: A tooltip explaining proxy resources handle requests to all sub-resources.
- Resource path**: A dropdown menu containing the value "/photos/".
- Resource name**: A text input field containing the value "{id}".
- CORS (Cross Origin Resource Sharing) Info**: A checked checkbox option.
- CORS (Cross Origin Resource Sharing) Info**: A tooltip explaining it creates an OPTIONS method allowing all origins, methods, and headers.

At the bottom right of the dialog are two buttons: "Cancel" and "Create resource", with "Create resource" being highlighted in orange.

aws | Services | Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598Id) > Create method

## Create method

**Method details**

Method type: GET

Integration type:

- Lambda function: Integrate your API with a Lambda function. (selected)
- HTTP: Integrate with an existing HTTP endpoint.
- Mock: Generate a response based on API Gateway mappings and transformations.
- AWS service: Integrate with an AWS Service.
- VPC link: Integrate with a resource that isn't accessible over the public internet.
- Lambda proxy integration: Send the request to your Lambda function as a structured event.

Lambda function: Provide the Lambda function name or alias. You can also provide an ARN from another account. (us-east-1)

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

Default timeout: The default timeout is 29 seconds.

**Create method**

- After saving the integration type, select “**Integration Request**”, click **Edit**. Choose “**When there are no templates defined (recommended)**” under the Request body passthrough, under the “**Mapping Template**”. Under the Content-Type, add “**application/json**” to the mapping. Finally, Under the Generate template dropdown, select “**Method Request passthrough**” and click “**Save**”.
- Finally, Create a new **Resource** at the root (“/”) to upload the photos to the S3 bucket. Give a name to the resource and make sure that “**Enable API Gateway CORS**” is checked

aws Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Create resource

## Create resource

**Resource details**

**Proxy resource** [Info](#)  
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path: / Resource name: uploadphoto

**CORS (Cross Origin Resource Sharing)** [Info](#)  
Create an OPTIONS method that allows all origins, all methods, and several common headers.

[Cancel](#) [Create resource](#)

Within the same resource, create another resource as shown below.

aws Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Create resource

## Create resource

**Resource details**

**Proxy resource** [Info](#)  
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path: /uploadphoto/ Resource name: {item}

**CORS (Cross Origin Resource Sharing)** [Info](#)  
Create an OPTIONS method that allows all origins, all methods, and several common headers.

[Cancel](#) [Create resource](#)

Under the new created resource, create a new **PUT** method.

Screenshot of the AWS API Gateway 'Edit integration request' page. Red numbered arrows point to specific fields:

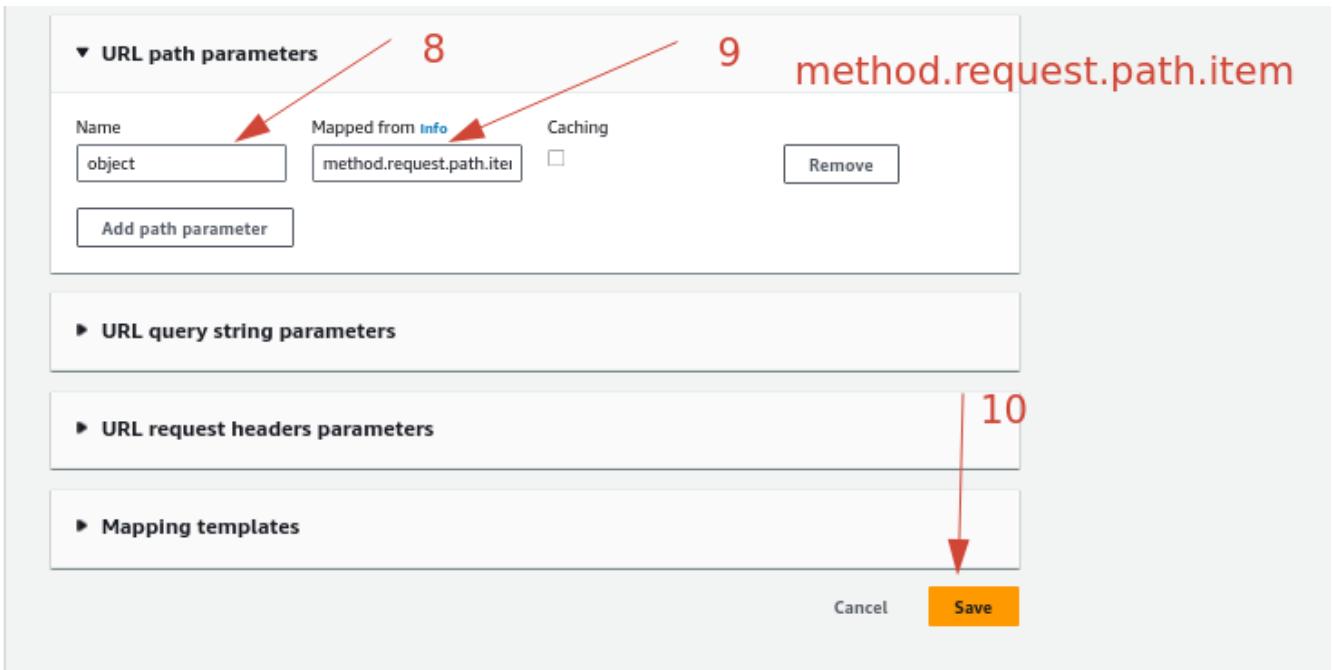
1. Integration type: AWS service (selected)
2. AWS Region: us-east-1
3. AWS service: Simple Storage Service (S3)
4. HTTP method: PUT
5. Path override - optional: photobucket-choephel-2024-4150/photos/{object}
6. Execution role: arn:aws:iam::211125774779:role/apiS3PutGet-Role
7. Content handling: Passthrough

Request body passthrough settings:

- When no template matches the request content-type header
- When there are no templates defined (recommended)
- Never

Warning message in a callout box:

**⚠️** If you set the request body passthrough to **When no templates matches** the request content-type header, API Gateway will pass all request payloads directly to the endpoint without transformation, and will transform any matches for the incoming content type. To secure your integration, select **When there are no templates defined (recommended)**.



- Select the method created and select the **AWS Region** where the S3 bucket was created (in this case, is **us-east-1** or N. Virginia). Under **AWS Service**, choose **Simple Storage Service (S3)**. Under **Action Type**, select “**Use path override**” and add the path override; in this case, it is the path where the photos are going to be stored in the bucket (**photobucket-choephel-2024-4150/photos/{object}**). Attach the execution role previously for the S3 bucket. Click “**Save**” to set the configuration.

Once the method is created, select the **Integration Request** under the method and create a mapping configuration as shown above.

AWS Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Edit method response

## Edit method response Info

1

**Response details**

HTTP status code **2**  
200

Header name **3**  
Header name  
Access-Control-Allow-Origin

Add header

Remove

Response body **4**

Content type application/json Model Empty

Add model

Remove

Cancel Save

AWS Services Search [Alt+S]

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Edit integration response

## Edit integration response Info

1

**Response details**

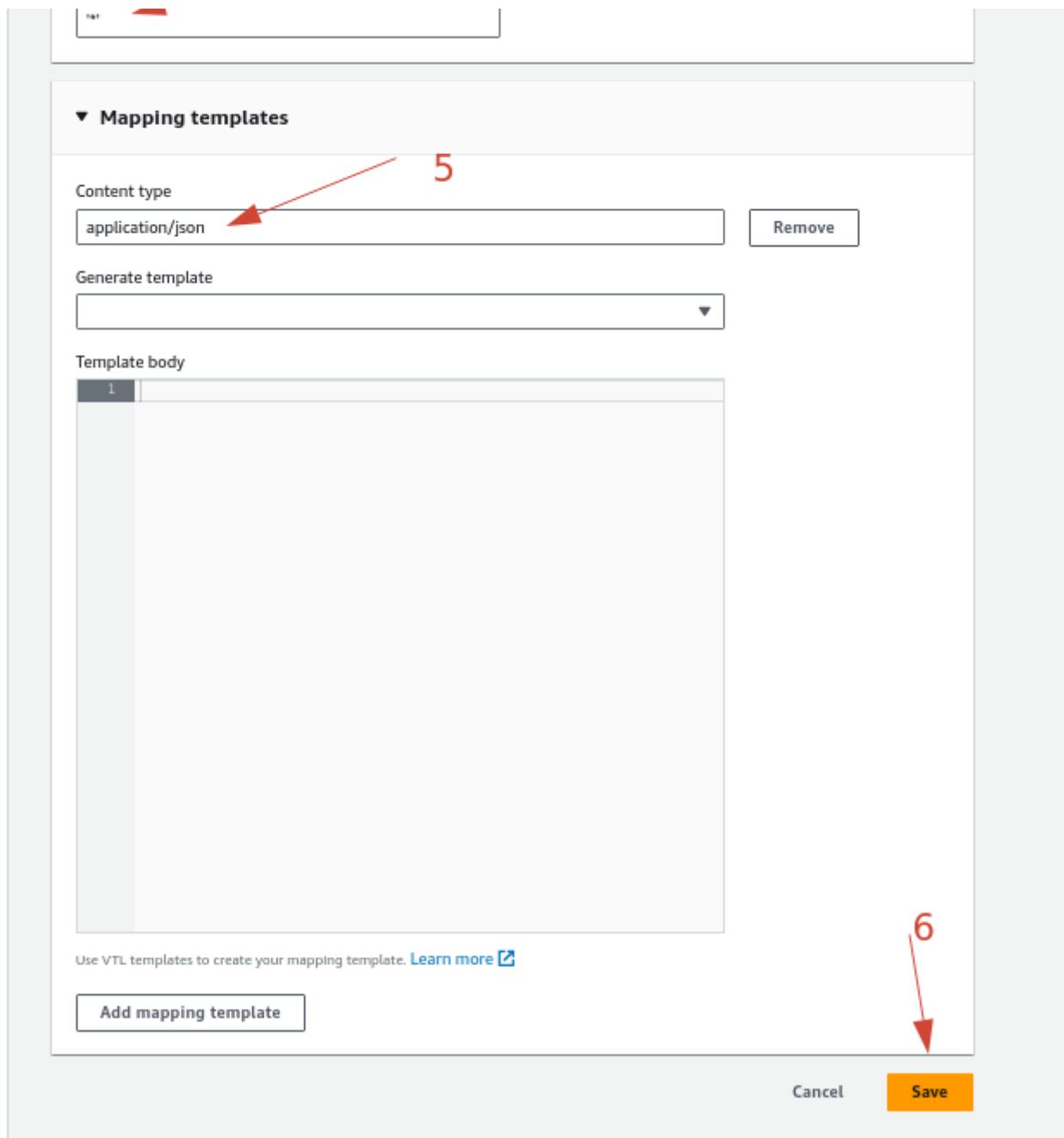
HTTP status regex Info

Method response status code **2**  
200

Content handling Learn more **3**  
Passthrough

**Header mappings** Info

Response header	Mapping value
Access-Control-Allow-Origin <b>4</b>	



- To allow images to be uploaded to the S3 bucket, we need to add the binary media types. For this case, we are going to allow **JPEG**, **JPG**, and **PNG** to be uploaded to the bucket. To do that, select **API Setting** in the left column and add the media types under **Binary Media Types**, as shown below

**API settings**

**API details**

API name PhotoGalleryAPI	Default endpoint Active <a href="https://sqses5981d.execute-api.us-east-1.amazonaws.com">https://sqses5981d.execute-api.us-east-1.amazonaws.com</a>
Description -	API key source Header
API endpoint type Regional	Content encoding Inactive

**Binary media types (3)**  
Add or remove media types that contain binary data.

Media type	<b>image/jpeg</b>	<b>Manage media types</b>
	image/png	< 1 >
	image/jpg	▼

**Tags (0)**  
Find resources

Key	▲	Value
No tags		
No tags associated with the resource.		
<b>Manage tags</b>		

- Make sure to **enable CORS** for the resources. To do that, select the resources and click the **Actions** dropdown and select “Enable CORS”.

**Resources**

**Resource details**

Path /confirmemail	Resource ID omj1se		
<b>Methods (2)</b>			
Method type	Integration type	Authorization	API key
<a href="#">OPTIONS</a>	Mock	None	Not required
<a href="#">POST</a>	Lambda	None	Not required

**3. repeat the process to all the resources**

API Gateway > APIs > Resources - PhotoGalleryAPI (sqses5981d)

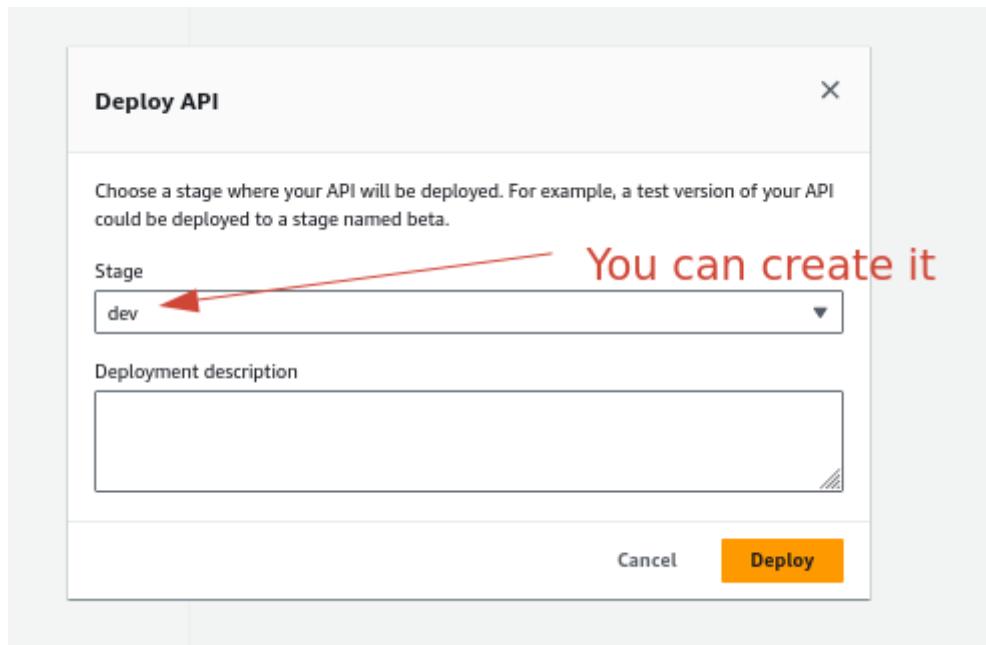
**Resources**

Path: /confirmemail

Methods (2)

Method type	Integration type	Authorization	API key
OPTIONS	Mock	None	Not required
POST	Lambda	None	Not required

- To deploy the API, click “Deploy API”, as shown below



- Create a new stage for deployment and called it “**dev**” as shown below
- Copy and save the API URL; will use this URL to access the resources from the website.

## 9. Updating the website

Add some resources to script used for the website. Update the **process.js** file under the staticwebsite/assets directory. Add the **<API\_URL>** and **<PHOTOGALLERY\_S3\_BUCKET\_NAME>** in lines 1 and 2, respectively, as shown below

```

DERS
for students
lambda-functions
  addphoto.py
  confirmemail.py
  getphoto.py
  getphotos.py
  login.py
  search.py
  signup.py
staticwebsite
  assets
    css
    css-rtl
    img
    plugins
    sass
    scripts
  process.js

```

```

1 const API_URL = "<API_URL>";
2 const PHOTOGALLERY_S3_BUCKET_URL = "<PHOTOGALLERY_S3_BUCKET_URL>";
3
4 function clearSession() {
5   sessionStorage.clear();
6   location.href='login.html';
7 };
8
9 $.urlParam = function(name){
10   var results = new RegExp('[\?&]' + name + '=([^&#]*)').exec(window.location.href);
11   return results[1] || 0;
12 }
13
14 function processLogin() {
15   var username = $("#username").val();
16   var password = $("#password").val();
17
18   var datadir = {
19     username: username,
20     password: password
21   };
22
23   $.ajax({

```

After making the updates, upload the files from the ‘staticwebsite’ folder to the S3 bucket for the website (eg. **staticwebsite-choephel-2024-4150**).

aws | Services | Search [Alt+S]

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**. For folder upload

**For file upload**

**Files and folders (369 Total, 21.1 MB)**

All files and folders in this table will be uploaded.

<input type="checkbox"/>	Name	Folder	Type	Size
<input type="checkbox"/>	animate.css	assets/css/	text/css	72.3 KB
<input type="checkbox"/>	grey.css	assets/css/theme/	text/css	73.3 KB
<input type="checkbox"/>	base.css	assets/css/theme/	text/css	73.3 KB
<input type="checkbox"/>	gold.css	assets/css/theme/	text/css	73.3 KB
<input type="checkbox"/>	teal.css	assets/css/theme/	text/css	73.3 KB
<input type="checkbox"/>	red.css	assets/css/theme/	text/css	73.3 KB
<input type="checkbox"/>	purple.css	assets/css/theme/	text/css	73.3 KB
<input type="checkbox"/>	dark.css	assets/css/theme/	text/css	117.7 KB
<input type="checkbox"/>	blue-grey.css	assets/css/theme/	text/css	73.4 KB
<input type="checkbox"/>	blue.css	assets/css/theme/	text/css	73.3 KB

**Find by name**

1 2 3 4 5 6 7 ... 37 >

**Destination** Info

Destination  
<s3://staticwebsite-choephel-2024-4150-new>

▶ **Destination details**  
Bucket settings that impact new objects stored in the specified destination.

▶ **Permissions**  
Grant public access and access to other AWS accounts.

▶ **Properties**  
Specify storage class, encryption settings, tags, and more.

**upload** ↓

Cancel **Upload**

aws Services Search [Alt+S]

Amazon S3 > Buckets staticwebsite-choephel-2024-4150-new

### staticwebsite-choephel-2024-4150-new Info Publicly accessible

Objects Properties Permissions Metrics Management Access Points

**Objects (9) Info**  
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

1. search for index.html  
2. click on it

Name	Type	Last modified	Size	Storage class
index.html	html	January 17, 2024, 13:05:14 (UTC-05:00)	2.9 KB	Standard

aws Services Search [Alt+S]

Amazon S3 Buckets staticwebsite-choephel-2024-4150 Info Publicly accessible

Objects Properties Permissions Metrics Management Access Points

**Bucket overview**

AWS Region: US East (N. Virginia) us-east-1 | Amazon Resource Name (ARN): arn:aws:s3:::staticwebsite-choephel-2024-4150 | Creation date: January 14, 2024, 01:30:40 (UTC-05:00)

**Bucket Versioning**  
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning: Disabled | Multi-factor authentication (MFA) delete: Enabled

**Tags (0)**  
You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

**Default encryption** Info  
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type: Server-side encryption with Amazon S3 managed keys (SSE-S3)  
Bucket Key: When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS. [Learn more](#)

**Enabled**

**Intelligent-Tiering Archive configurations (0)**  
Enable objects stored in the Intelligent-Tiering storage class to tier down to the Archive Access tier or the Deep Archive Access tier which are optimized for objects that will be rarely accessed for long periods of time. [Learn more](#)

**Server access logging**  
Log requests for access to your bucket. Use [CloudWatch](#) to check the health of your server access logging. [Learn more](#)

Server access logging: Disabled

**AWS CloudTrail data events** Info  
Configure CloudTrail data events to log Amazon S3 object-level API operations in the CloudTrail console. [Learn more](#)

**Event notifications (0)**  
Send a notification when specific events occur in your bucket. [Learn more](#)

**Amazon EventBridge**  
For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. [Learn more](#) or [see EventBridge pricing](#)

Send notifications to Amazon EventBridge for all events in this bucket: Off

Transfer acceleration

Transfer acceleration  
Disabled

Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. [Learn more](#)

Object Lock  
Disabled

Requester pays

When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. [Learn more](#)

Requester pays  
Disabled

**Static website hosting**

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting  
Enabled  
Hosting type  
Bucket hosting  
Bucket website endpoint  
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://staticwebsite-cheopel-2024-4150.s3-website-us-east-1.amazonaws.com>

## 10. 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.
- Confirm the user's email.
- Goto login page and login using the user created above.
- Goto add photos page and add a new photo.
- Add a more photos and try browsing and searching for photos.

## Challenges:

- Competition of the main functionalities of the website as described in the instructions (**80 points**)
- Allow users to delete a picture in their own gallery (**10 Points**)
- Allow users to update the title, description and tags of the pictures after creation (**10 Points**)

## Deliverables:

- A video that shows the functions of your website, including user signup/login/logout, upload a picture, view pictures under the same tag. If you have completed the challenge tasks, also be sure to include the following example: a deletion of a picture, search for a picture, update of a picture's information, etc.. You can also show the database to prove that the picture has been deleted or modified.

## Miscellaneous Information

## Hints on Debugging Process:

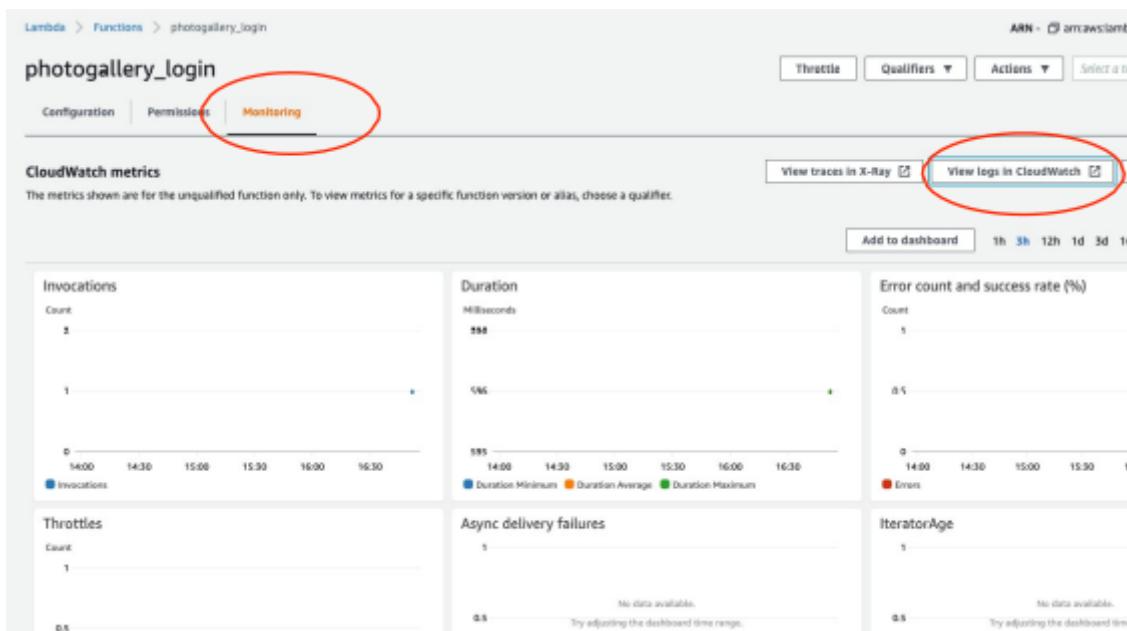
- Test your code on localhost before uploading to cloud

```
[11:20] (/>w<)/ ~ $ cd /Users/jolinchen/Desktop/Spring_2021/TA_ECE_4150/Lab1/staticwebsite
[11:20] (/>w<)/ ~/Desktop/Spring_2021/TA_ECE_4150/Lab1/staticwebsite $ http-server
Starting up http-server, serving .
Available on:
  http://127.0.0.1:8080
  http://10.0.0.12:8080
Hit CRTL-C to stop the server
```

- Go to the root folder of the staticwebsite, type in command line http-server to set up HTTP server for testing. You can find the installation information of http-server at: <https://www.npmjs.com/package/http-server>. Then open the url as indicated in your terminal.

## 2. Use CloudWatch to monitor data flow

- This is to locate problems occurring in lambda functions. Go to your lambda function list and choose any lambda function, go to “Monitoring” —> “view logs in CloudWatch”.



You have to create a log group in order to get the logs. To so do, in your newly opened CloudWatch Management Console page, copy the link above (in this case: group: /aws/lambda/ photogallery\_login): Go back to Log groups, choose “Create log group”, in Log group name, paste the link you just copied then create:

### **✖ Log group does not exist**

The specific log group: /aws/lambda/photogallery\_login does not exist in this account or region.

## Create log group

**Log group details**

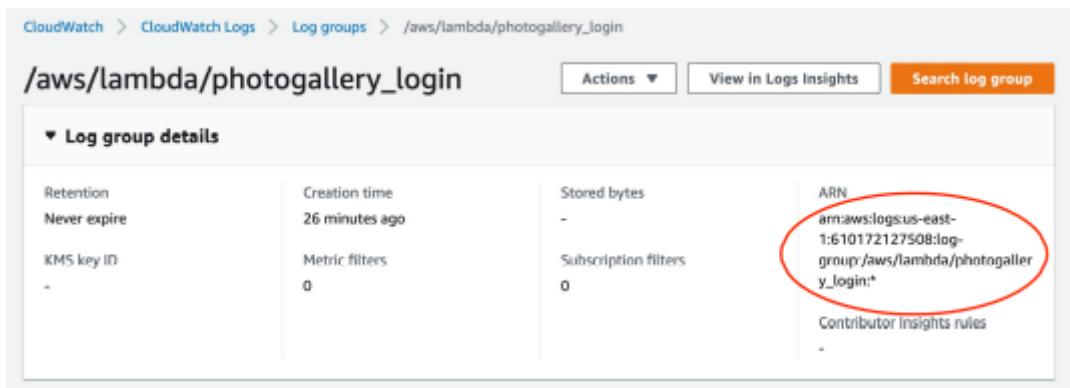
Log group name

Retention setting

KMS key ARN - optional

Cancel
Create

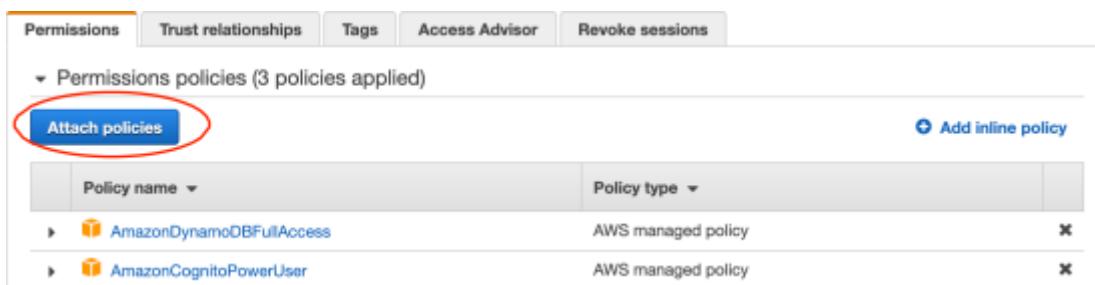
In your newly created log group, click on the dropdown button of “Log group details” and copy the ARN in it:



CloudWatch > CloudWatch Logs > Log groups > /aws/lambda/photogallery\_login

/aws/lambda/photogallery_login																			
<a href="#">Actions</a> ▾ <a href="#">View in Logs Insights</a> <a href="#" style="background-color: #ff5722; color: white; border: 1px solid #ff5722; padding: 2px 10px; border-radius: 5px; font-weight: bold;">Search log group</a>																			
<b>Log group details</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%;">Retention</td> <td style="width: 25%;">Creation time</td> <td style="width: 25%;">Stored bytes</td> <td style="width: 25%;">ARN</td> </tr> <tr> <td>Never expire</td> <td>26 minutes ago</td> <td>-</td> <td>arn:aws:logs:us-east-1:610172127508:log-group:/aws/lambda/photogallery_login:*</td> </tr> <tr> <td>KMS key ID</td> <td>Metric filters</td> <td>Subscription filters</td> <td>Contributor Insights rules</td> </tr> <tr> <td>-</td> <td>0</td> <td>0</td> <td>-</td> </tr> </table>				Retention	Creation time	Stored bytes	ARN	Never expire	26 minutes ago	-	arn:aws:logs:us-east-1:610172127508:log-group:/aws/lambda/photogallery_login:*	KMS key ID	Metric filters	Subscription filters	Contributor Insights rules	-	0	0	-
Retention	Creation time	Stored bytes	ARN																
Never expire	26 minutes ago	-	arn:aws:logs:us-east-1:610172127508:log-group:/aws/lambda/photogallery_login:*																
KMS key ID	Metric filters	Subscription filters	Contributor Insights rules																
-	0	0	-																

Now we need to grant write access to the log group. Go to IAM console and open roles. Go to lambda\_photogallery\_role you created before and choose “Attach policies”:



Permissions Trust relationships Tags Access Advisor Revoke sessions

▼ Permissions policies (3 policies applied)

**Attach policies** [Add inline policy](#)

Policy name	Policy type	
AmazonDynamoDBFullAccess	AWS managed policy	X
AmazonCognitoPowerUser	AWS managed policy	X

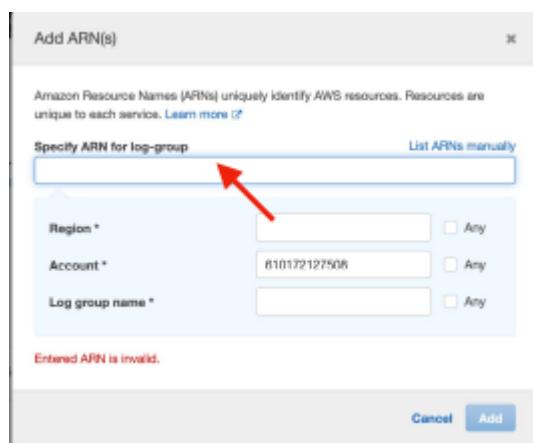
Choose “Create policy”. In Service, choose “CloudWatch Logs”. In Actions - Write, click on the dropdown list and choose as indicated below.

**Access level**[Expand all](#) | [Collapse all](#)

- ▶  List
- ▶  Read
- ▼  Write (3 selected)

<input type="checkbox"/> AssociateKmsKey <a href="#">?</a>	<input type="checkbox"/> DeleteLogStream <a href="#">?</a>	<input type="checkbox"/> PutMetricFilter <a href="#">?</a>
<input type="checkbox"/> CancelExportTask <a href="#">?</a>	<input type="checkbox"/> DeleteMetricFilter <a href="#">?</a>	<input type="checkbox"/> PutResourcePolicy <a href="#">?</a>
<input type="checkbox"/> CreateExportTask <a href="#">?</a>	<input type="checkbox"/> DeleteResourcePolicy <a href="#">?</a>	<input type="checkbox"/> PutRetentionPolicy <a href="#">?</a>
<input type="checkbox"/> CreateLogDelivery <a href="#">?</a>	<input type="checkbox"/> DeleteRetentionPolicy <a href="#">?</a>	<input type="checkbox"/> PutSubscriptionFilter <a href="#">?</a>
<input checked="" type="checkbox"/> CreateLogGroup <a href="#">?</a>	<input type="checkbox"/> DeleteSubscriptionFilter <a href="#">?</a>	<input type="checkbox"/> TagLogGroup <a href="#">?</a>
<input checked="" type="checkbox"/> CreateLogStream <a href="#">?</a>	<input type="checkbox"/> DisassociateKmsKey <a href="#">?</a>	<input type="checkbox"/> UntagLogGroup <a href="#">?</a>
<input type="checkbox"/> DeleteDestination <a href="#">?</a>	<input type="checkbox"/> PutDestination <a href="#">?</a>	<input type="checkbox"/> UpdateLogDelivery <a href="#">?</a>
<input type="checkbox"/> DeleteLogDelivery <a href="#">?</a>	<input type="checkbox"/> PutDestinationPolicy <a href="#">?</a>	
<input type="checkbox"/> DeleteLogGroup <a href="#">?</a>	<input checked="" type="checkbox"/> PutLogEvents <a href="#">?</a>	

In Resources, choose “Specific”, click on “Add ARN” in log-group and paste your copied ARN into it. After finishing everything, check on your json file, which should look like the screenshot below. Click on “Review Policy”, name your new policy as you like and create. Attach the policy to your role after creation.



```
1 {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Sid": "VisualEditor0",  
6             "Effect": "Allow",  
7             "Action": [  
8                 "logs:CreateLogStream",  
9                 "logs:CreateLogGroup",  
10                "logs:PutLogEvents"  
11            ],  
12            "Resource": "arn:aws:logs:us-east-1:610172127508:log-group:/aws/lambda/photogallery_login  
13                :*"  
14        }  
15    ]  
}
```

Now as you enter your log group of chosen lambda function, you should be able to monitor and examine all the activities.

/aws/lambda/photogallery\_login

Actions ▾ View in Logs Insights Search log group

► Log group details

Log streams Metric filters Subscription filters Contributor Insights

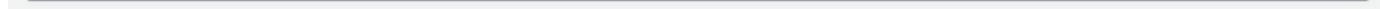
Log streams (1)

Filter log streams or try prefix search

Clear Delete Create log stream Search all

Last event time

Log stream 2021/01/25/[&LATEST]7d58d95e577b46c18c418c01a75b4246 2021-01-25 12:53:53 (UT...)



CloudWatch > CloudWatch Logs > Log groups > /aws/lambda/photogallery\_login >  
2021/01/25/[&LATEST]7d58d95e577b46c18c418c01a75b4246

Try CloudWatch Logs Insights Try Logs Insights X

CloudWatch Logs insights allows you to search and analyze your logs using a new, purpose-built query language. To learn more, read the AWS blog or visit our documentation.

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns.

View as text Actions Create Metric Filter

Filter events Clear 1m 30m 1h 12h Custom

Timestamp Message

No older events at this moment. Retry

2021-01-25T12:53:52.475-05:00 START RequestId: eb9ad9ca-18fe-47dc-9a5d-21f0493136f2 Version: \$LATEST

2021-01-25T12:53:53.052-05:00 END RequestId: eb9ad9ca-18fe-47dc-9a5d-21f0493136f2

2021-01-25T12:53:53.052-05:00 REPORT RequestId: eb9ad9ca-18fe-47dc-9a5d-21f0493136f2 Duration: 576.11 ms Billed Durat...

No newer events at this moment. Auto retry paused. Resume

