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

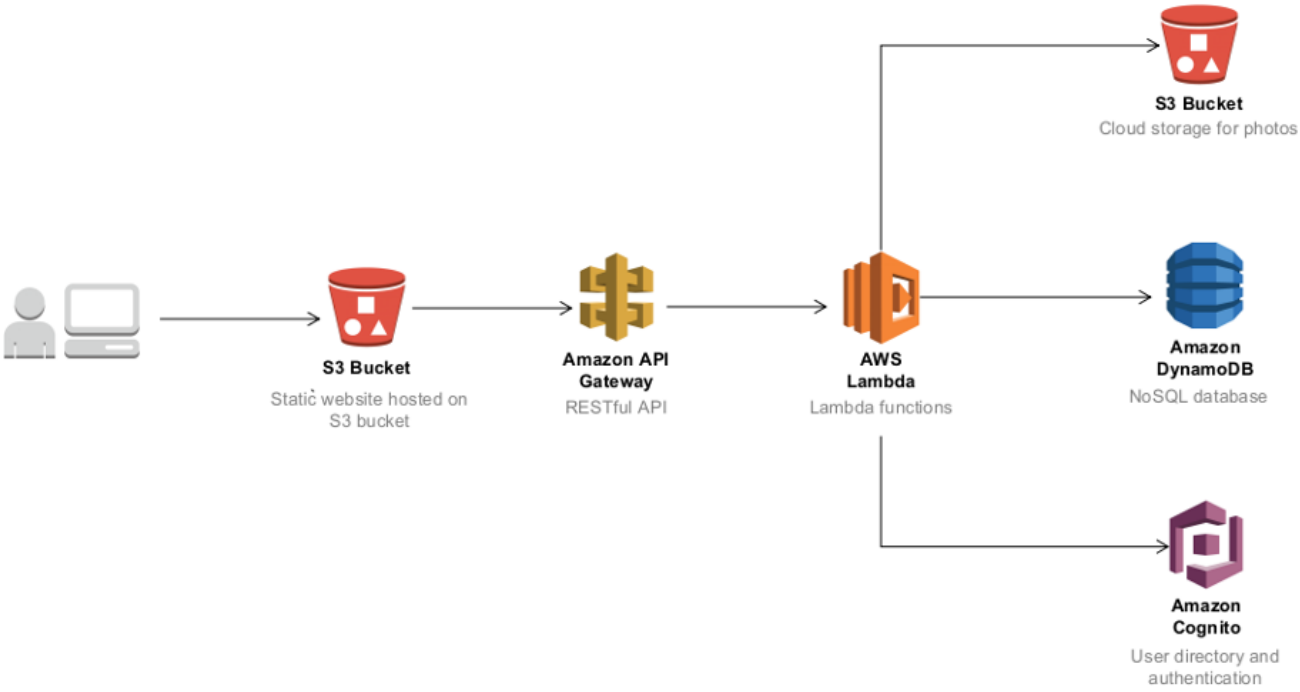


Fig. Architecture diagram of the Photo Gallery application showing AWS services used

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.

PhotoGallery

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.

PhotoID | 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.

CreationTime | 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

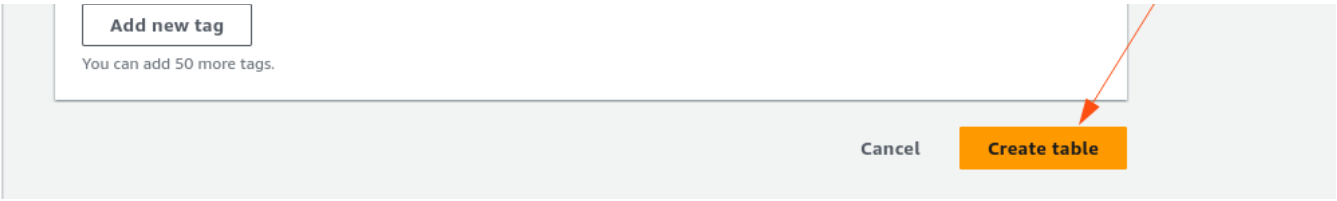
These are the default settings for your new table. You can change some of these settings after creating the table.

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

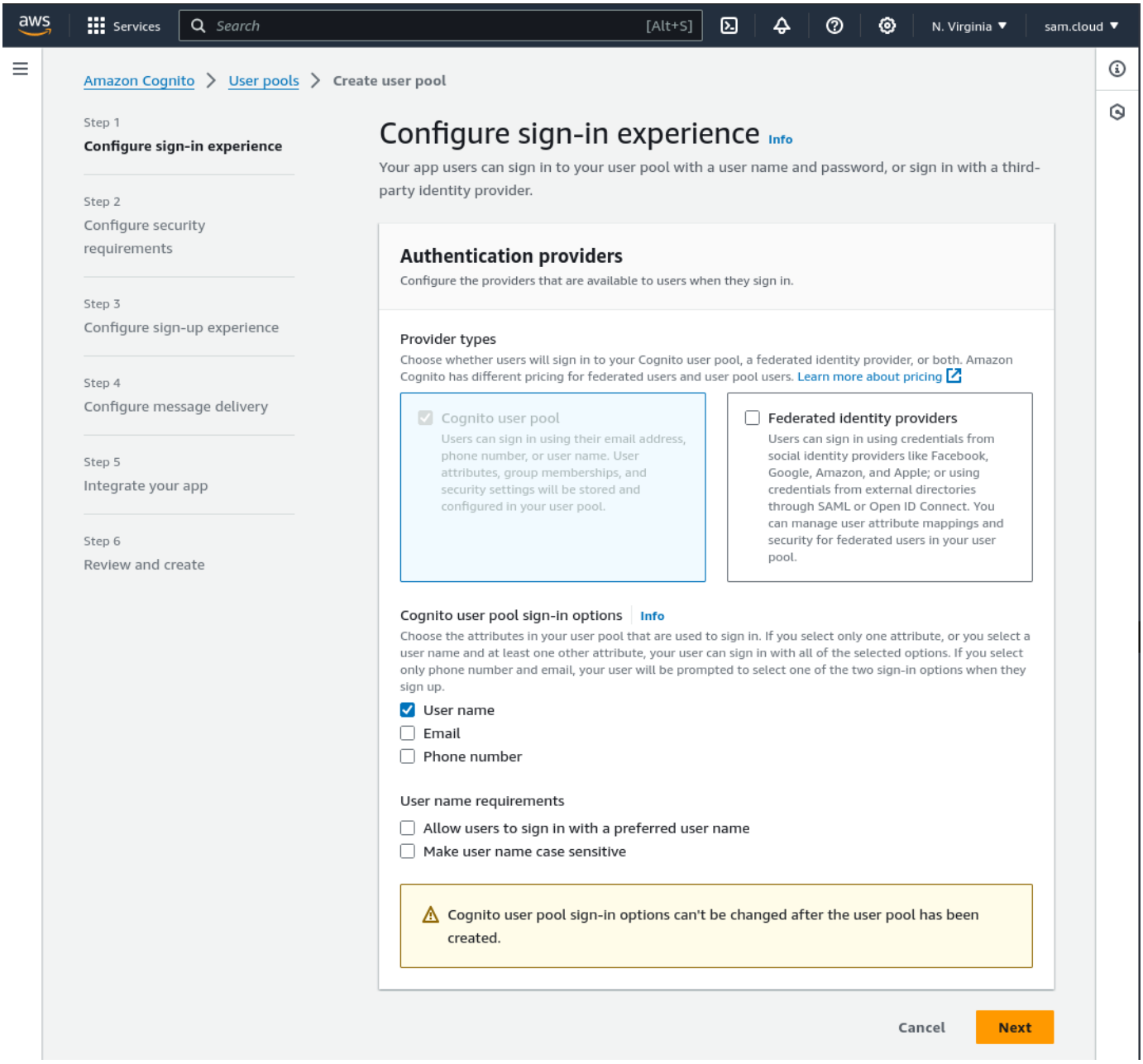
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

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.



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.

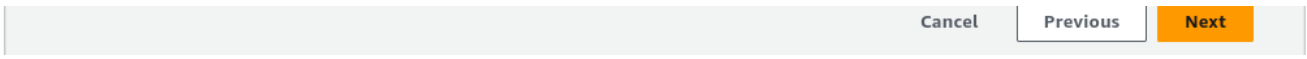
Self-service account recovery Info

- [x] 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

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

- [x] 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

- [x] 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

- [x] 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.

- ( ) Phone number
- (•) Email address
- ( ) Phone number and 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

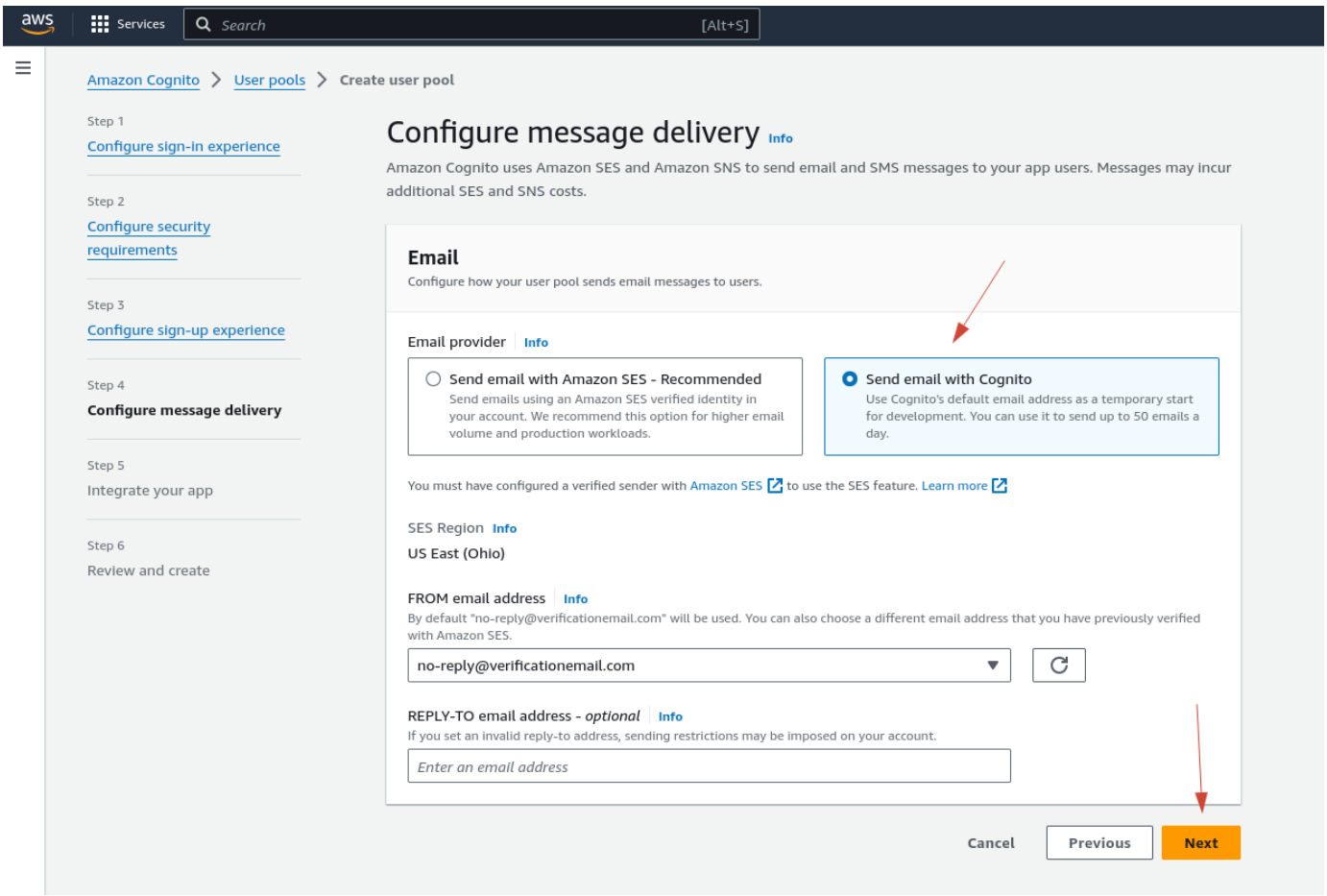
Additional required attributes

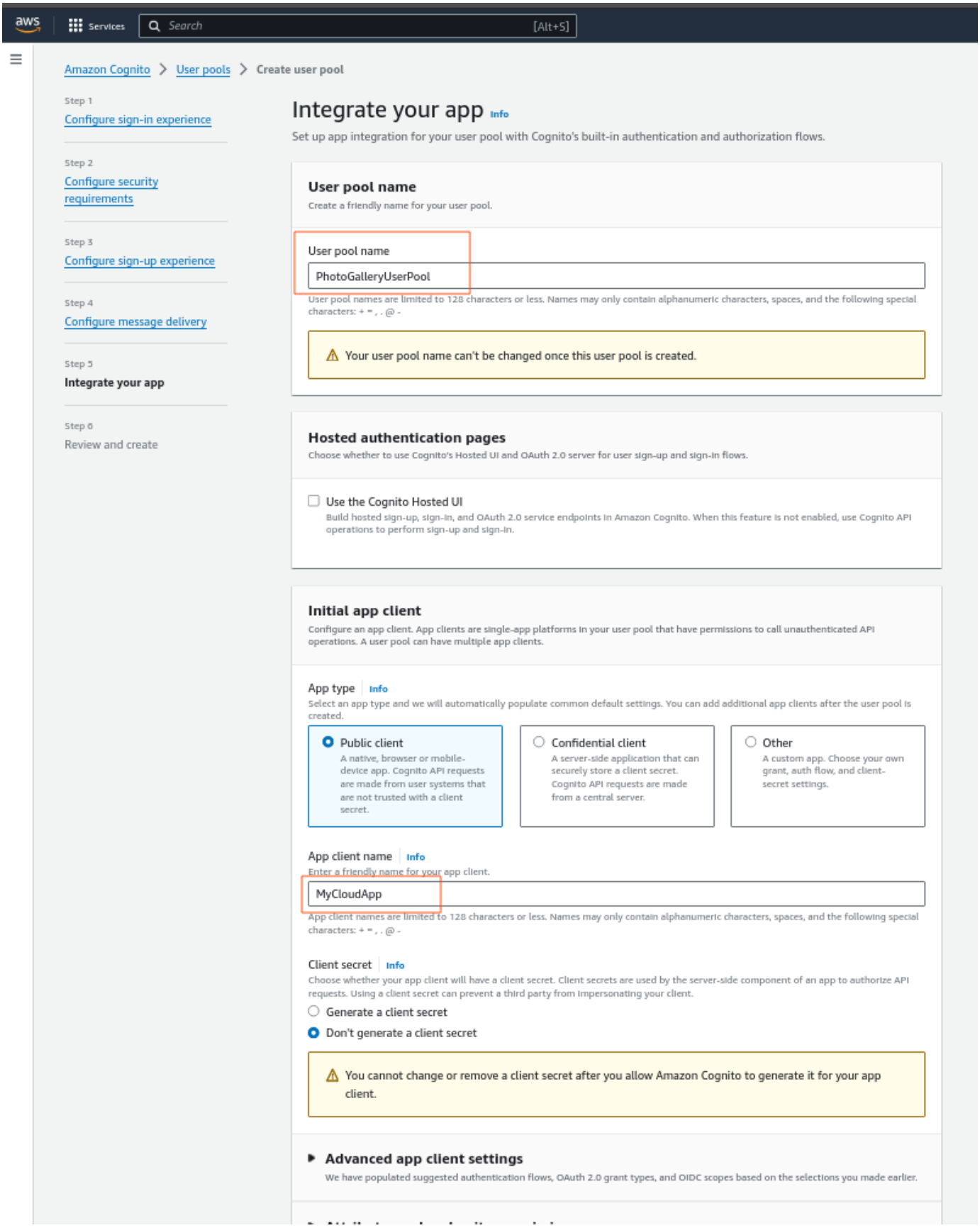
Select attributes

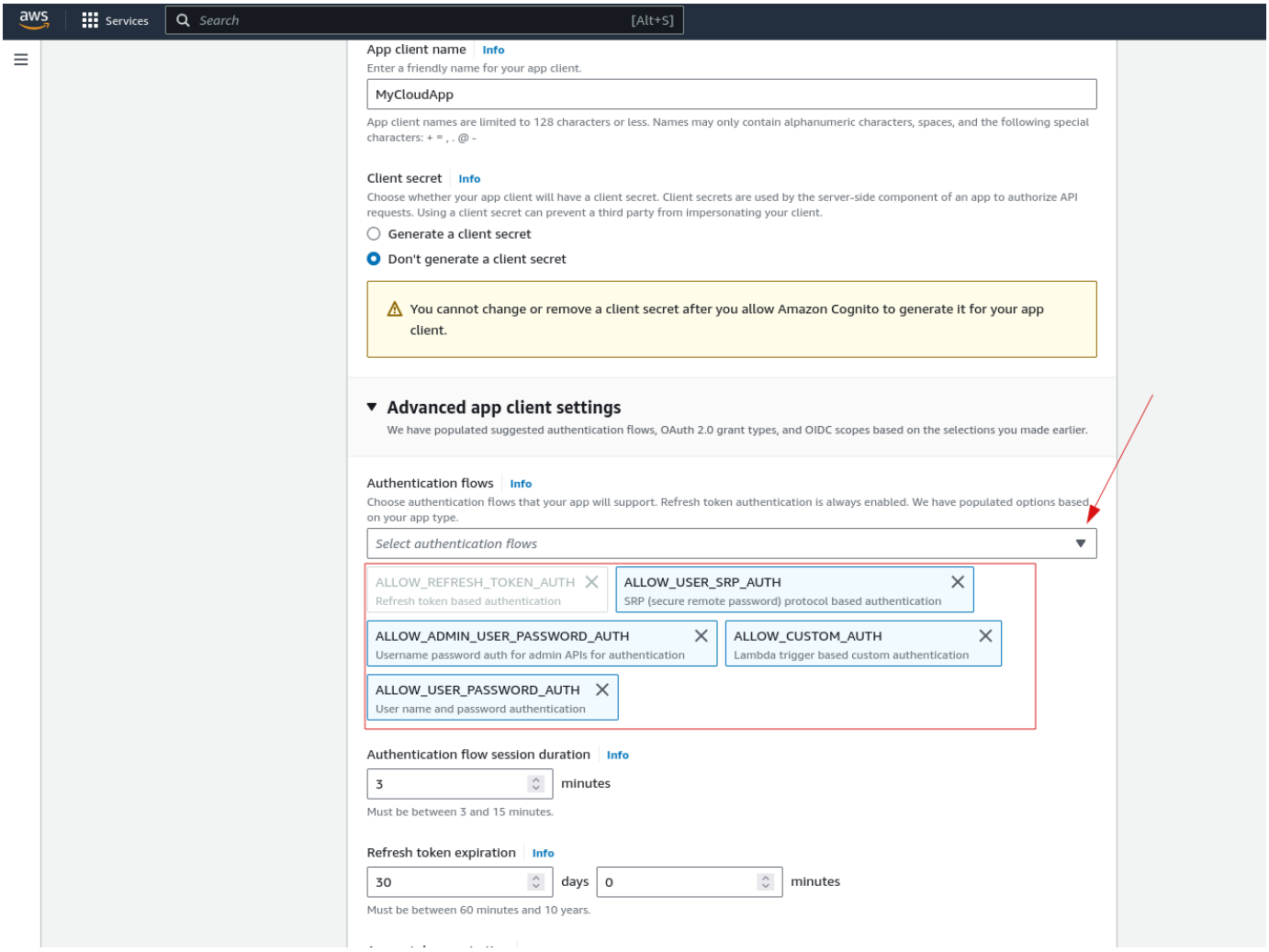
> ⚠ Required attributes can't be changed once this user pool has been created.

▸ 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
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
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.
no-reply@verificationemail.com
REPLY-TO email address - optional Info
If you set an invalid reply-to address, sending restrictions may be imposed on your account.
Enter an email address
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
PhotoGalleryUserPool
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: + = , . @ -
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.

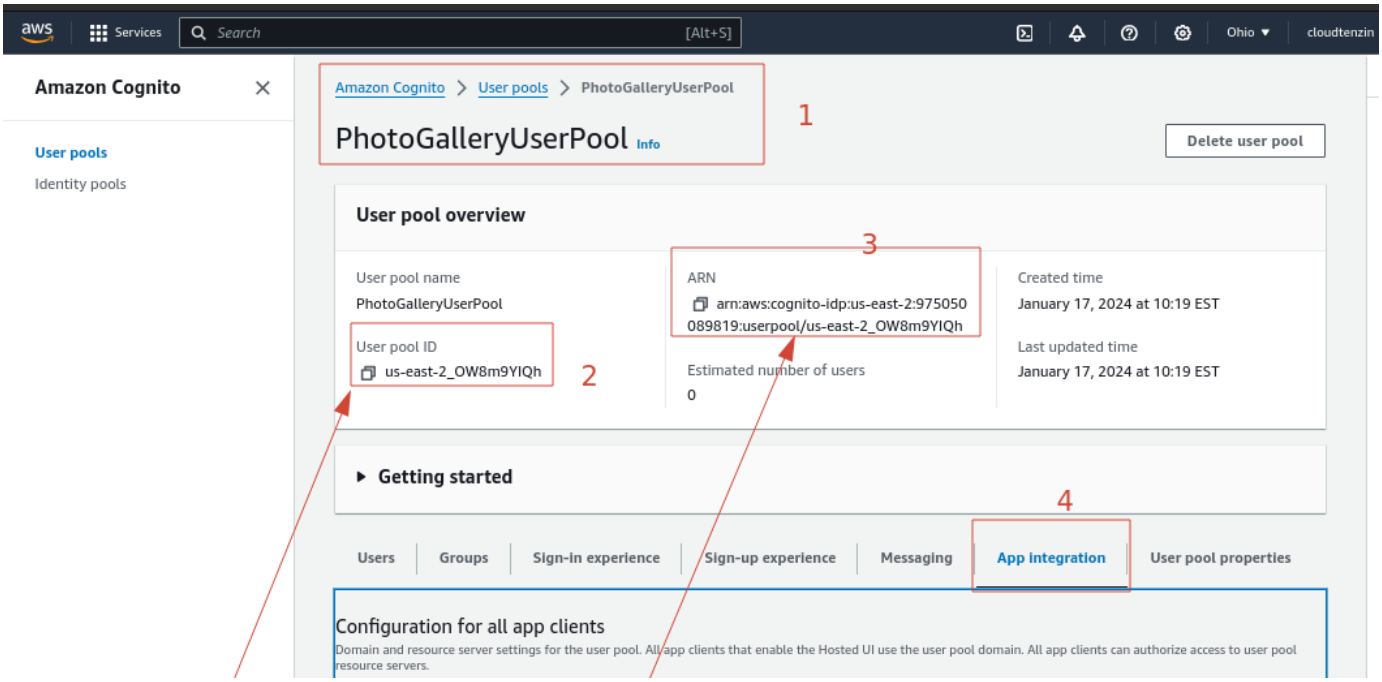


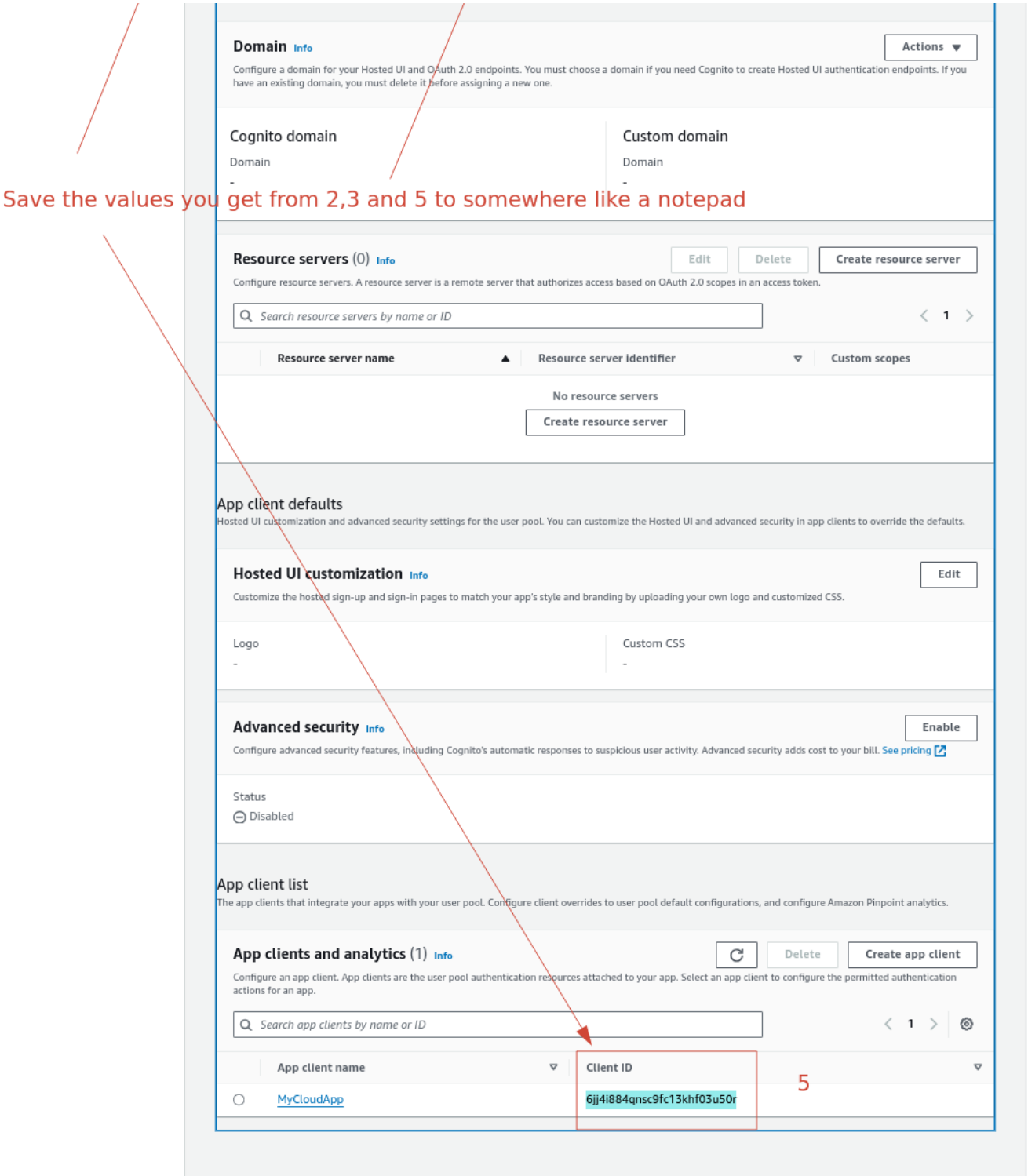
Enter “App client name” as **MyCloudApp**. Click on **Next**.

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.

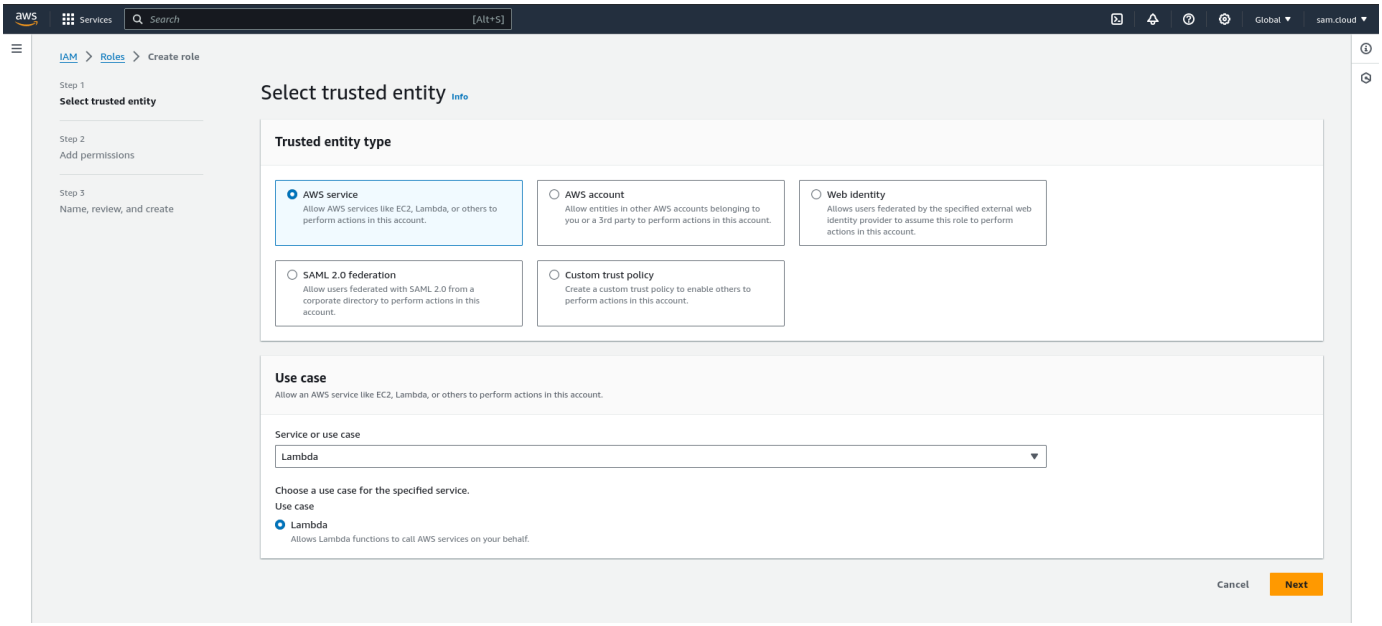


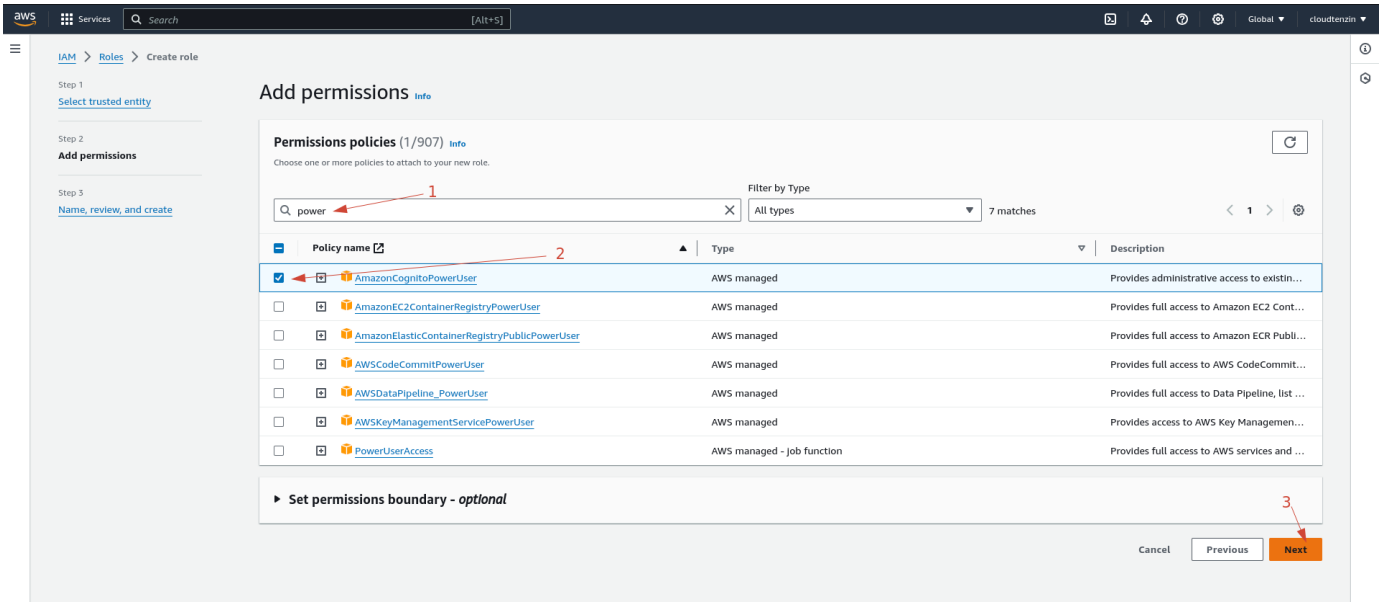


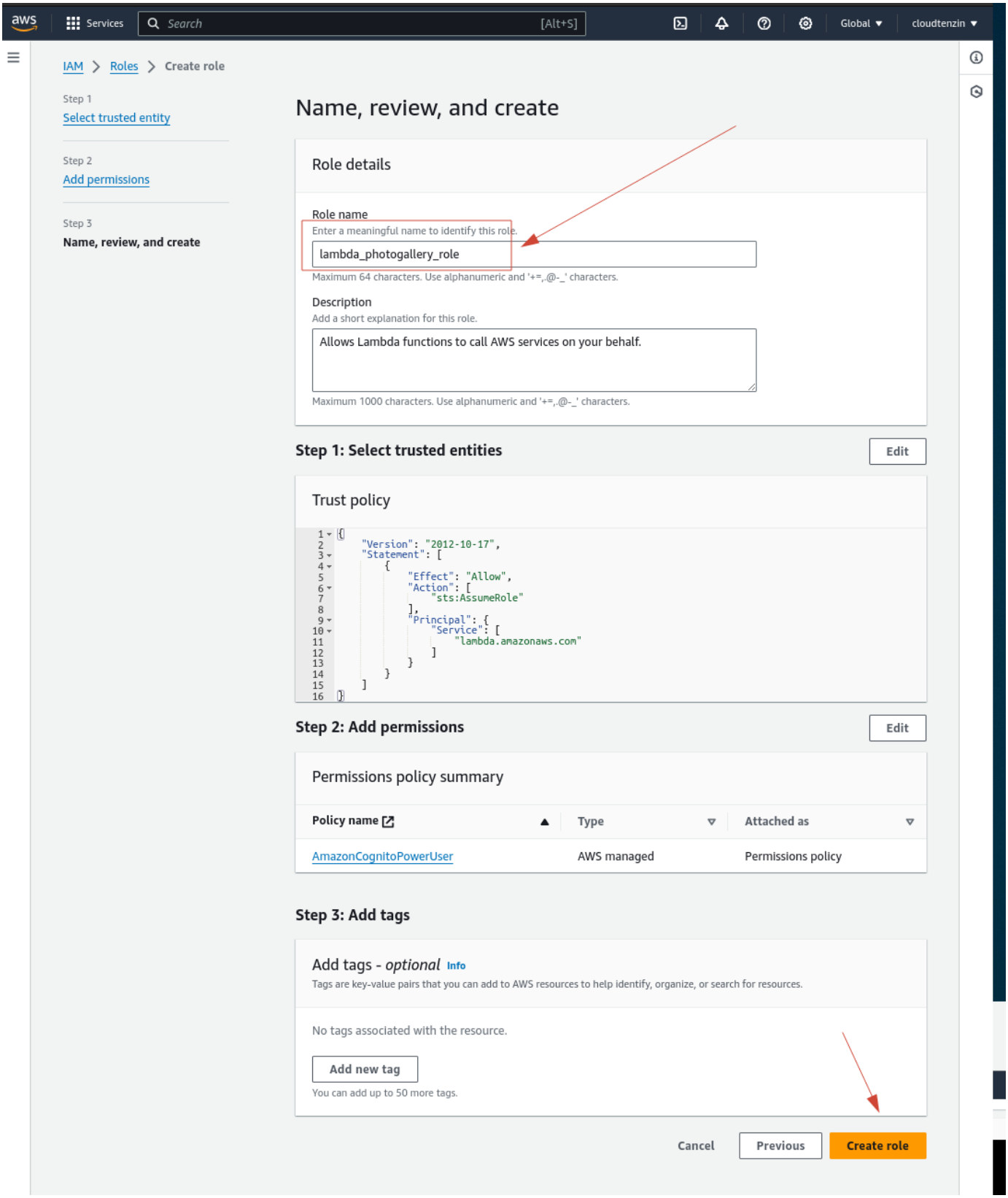
From **App Clients and analytics**, Under the new app client (**MyCloudApp**), copy and save the **App client id**; we will use this identifier for our serverless function.

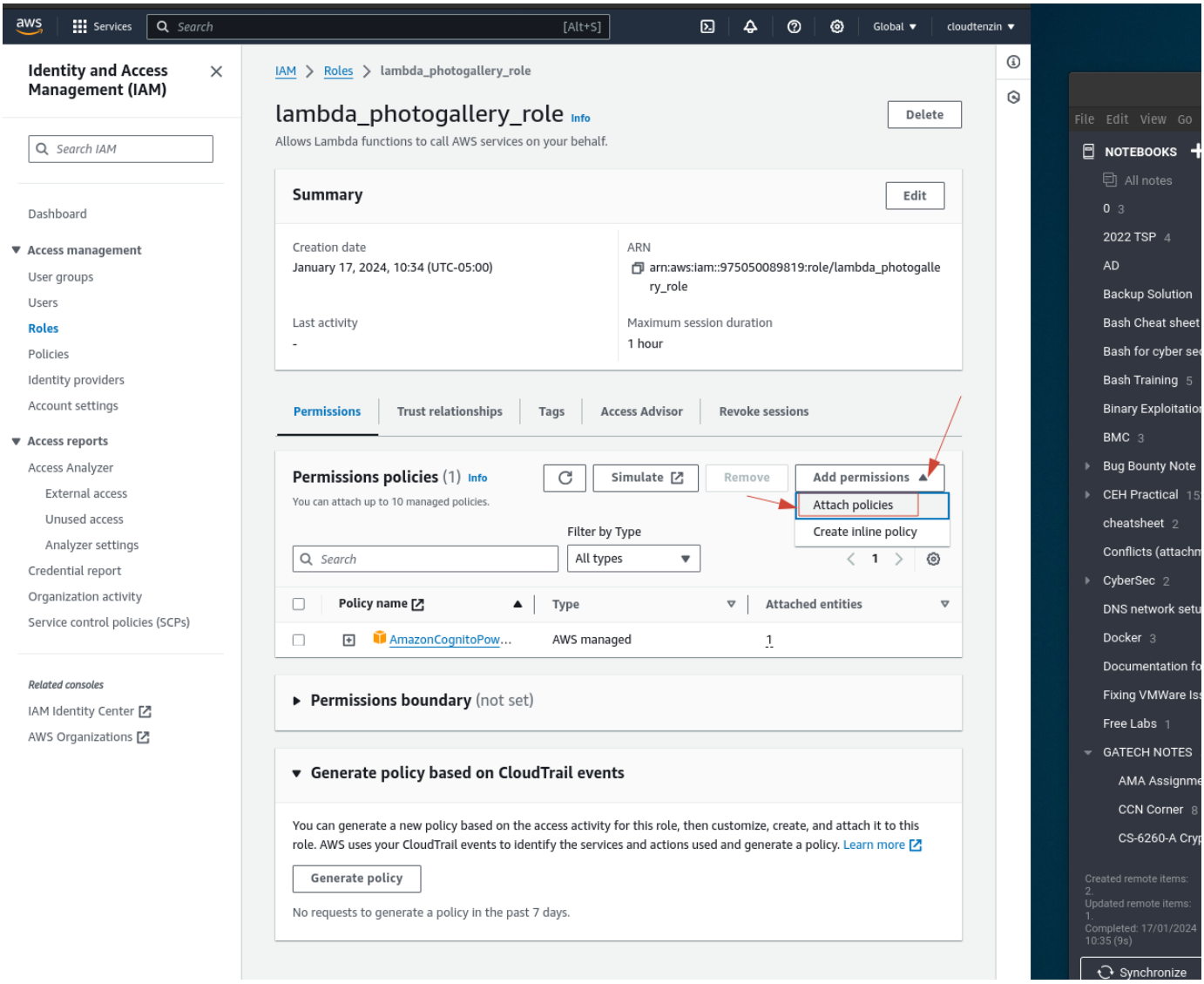
5. Create IAM Roles

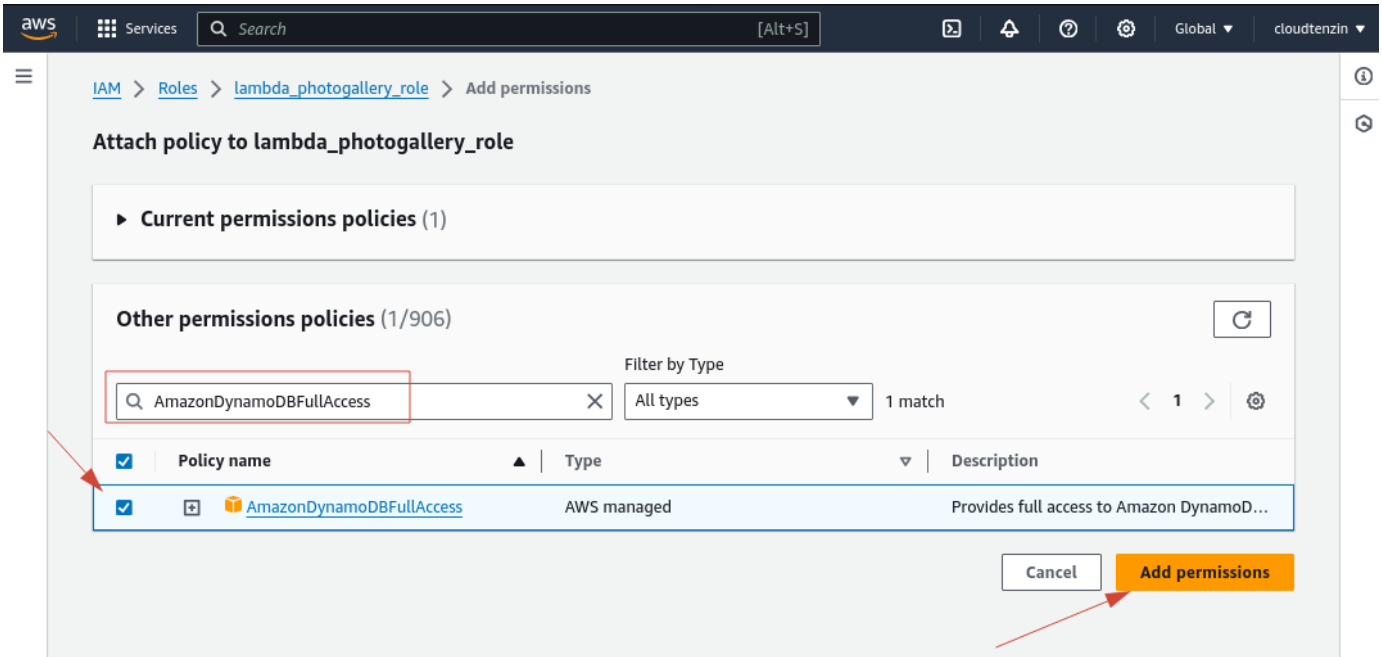
- In the AWS IAM console, goto roles and create a new IAM role called '**lambda_photogallery_role**' for Lambda service, as shown below. Attach policy called **AmazonDynamoDBFullAccess** to this role. Then attach policy called **AmazonCognitoPowerUser** to this role.

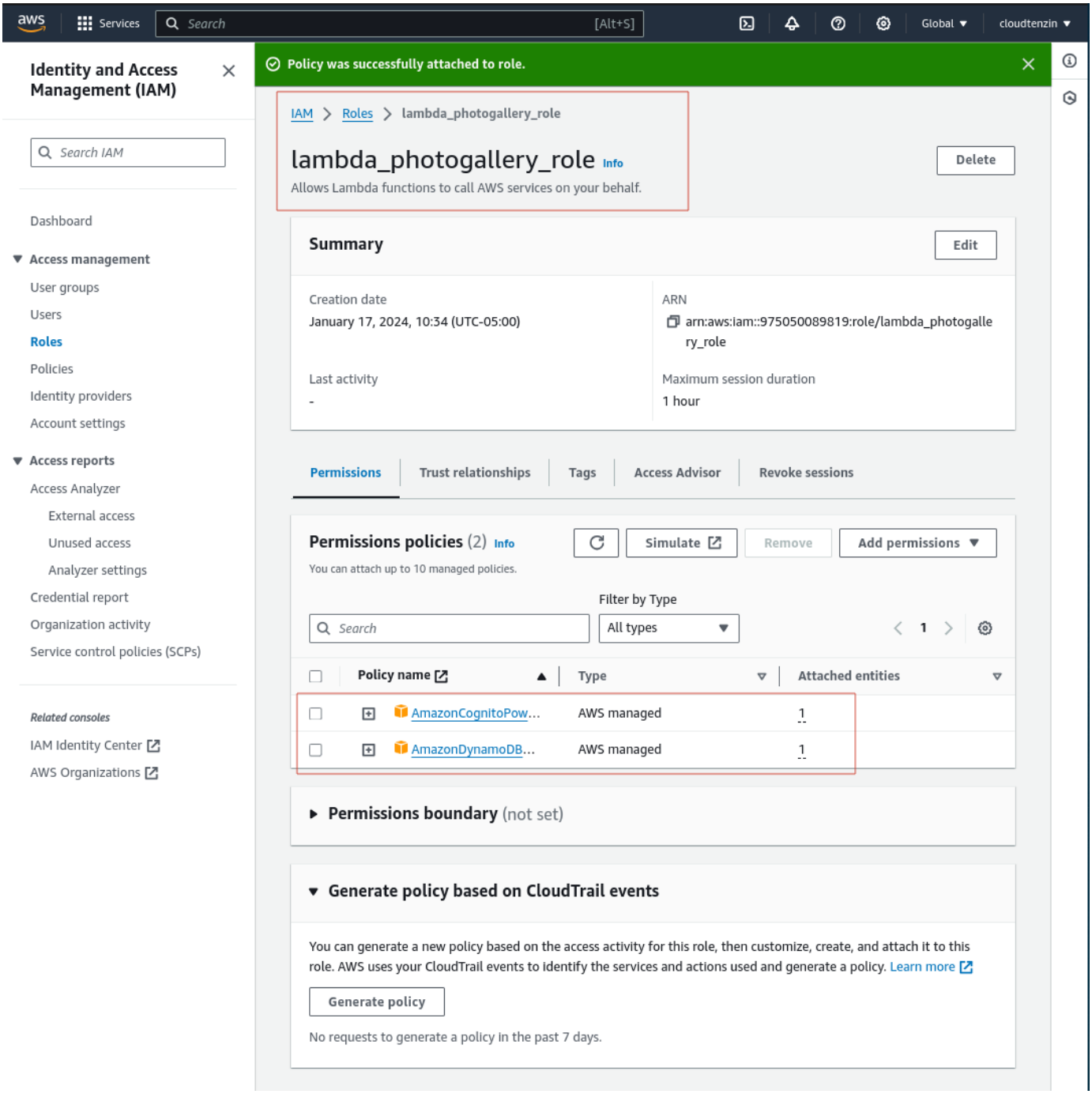

Services
Search
[Alt+S]
Global
sam.cloud
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Select trusted entity Info
Trusted entity type
AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
AWS account
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.
Cancel
Next


Services
Search
[Alt+S]
Global
cloudtenzin
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions Info
Permissions policies (1/907) Info
Choose one or more policies to attach to your new role.
1
power
Filter by Type
All types
7 matches
Policy name
Type
Description
2
AmazonCognitoPowerUser
AWS managed
Provides administrative access to existin...
AmazonEC2ContainerRegistryPowerUser
AWS managed
Provides full access to Amazon EC2 Cont...
AmazonElasticContainerRegistryPublicPowerUser
AWS managed
Provides full access to Amazon ECR Publi...
AWSCodeCommitPowerUser
AWS managed
Provides full access to AWS CodeCommit...
AWSDataPipeline_PowerUser
AWS managed
Provides full access to Data Pipeline, list ...
AWSKeyManagementServicePowerUser
AWS managed
Provides access to AWS Key Managemen...
PowerUserAccess
AWS managed - job function
Provides full access to AWS services and ...
Set permissions boundary - optional
3
Cancel
Previous
Next

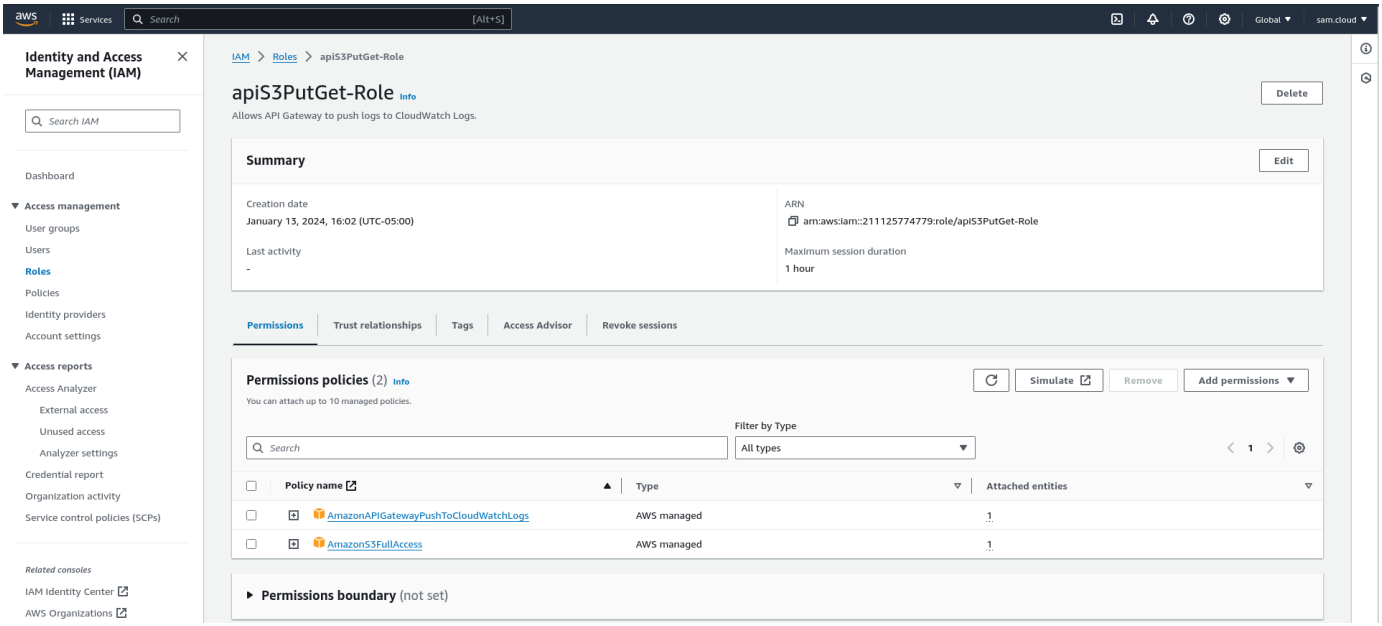

aws
Services
Search
[Alt+S]
Global
cloudtenzin
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
lambda_photogallery_role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Step 1: Select trusted entities
Edit
Trust policy
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"sts:AssumeRole"
],
"Principal": {
"Service": [
"lambda.amazonaws.com"
]
}
}
]
}
Step 2: Add permissions
Edit
Permissions policy summary
Policy name
Type
Attached as
AmazonCognitoPowerUser
AWS managed
Permissions policy
Step 3: Add tags
Add tags - optional
Info
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.
Cancel
Previous
Create role


aws
Services
Search
[Alt+S]
Global
cloudtenzin
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies (SCPs)
Related consoles
IAM Identity Center
AWS Organizations
IAM > Roles > lambda_photogallery_role
lambda_photogallery_role Info
Allows Lambda functions to call AWS services on your behalf.
Delete
Summary
Edit
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 (1) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Attach policies
Create inline policy
Filter by Type
Search
All types
Policy name
Type
Attached entities
AmazonCognitoPow...
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.
NOTEBOOKS
All notes
2022 TSP
AD
Backup Solution
Bash Cheat sheet
Bash Training
Binary Exploitation
BMC
Bug Bounty Note
CEH Practical
cheatsheet
CyberSec
DNS network setu
Docker
Documentation fo
Fixing VMWare Iss
Free Labs
GATECH NOTES
AMA Assignme
CCN Corner
CS-6260-A Cryp
Synchronize


aws
Services
Search
[Alt+S]
Global
cloudtenzin
IAM > Roles > lambda_photogallery_role > Add permissions
Attach policy to lambda_photogallery_role
Current permissions policies (1)
Other permissions policies (1/906)
Filter by Type
AmazonDynamoDBFullAccess
All types
1 match
Policy name
Type
Description
AmazonDynamoDBFullAccess
AWS managed
Provides full access to Amazon DynamoD...
Cancel
Add permissions

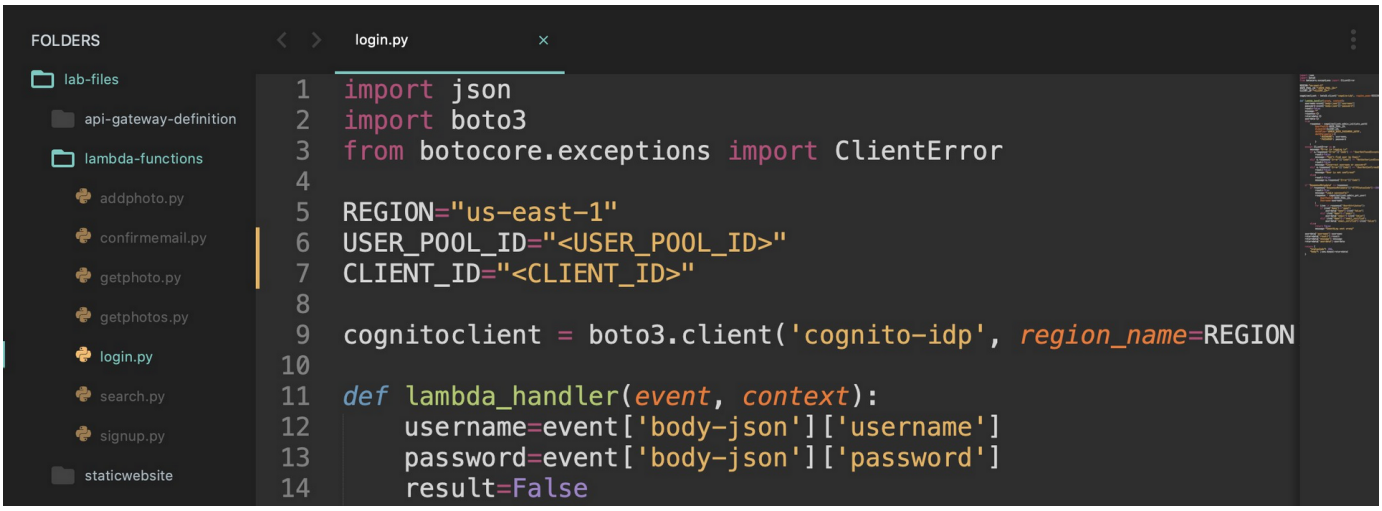


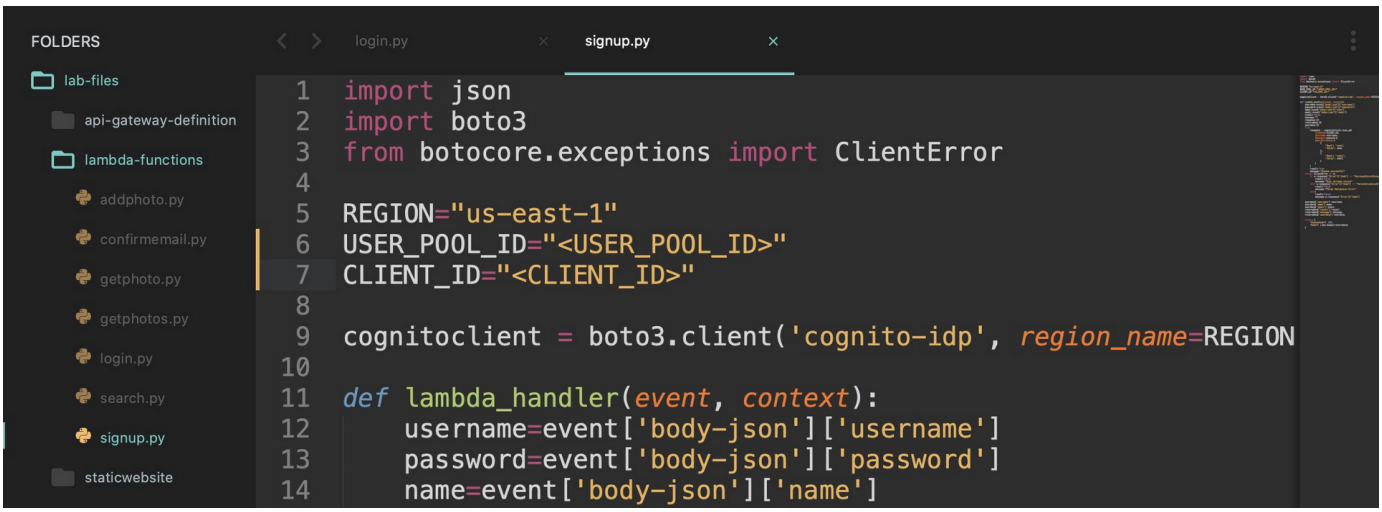
- Create another IAM role called '**apiS3PutGet-Role**' to upload the photos to the S3 Bucket service, as shown below. Attach policy called **AmazonAPIGatewayPushToCloudWatchLogs** to this role. Then attach policy called **AmazonS3FullAccess** to this role.



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.



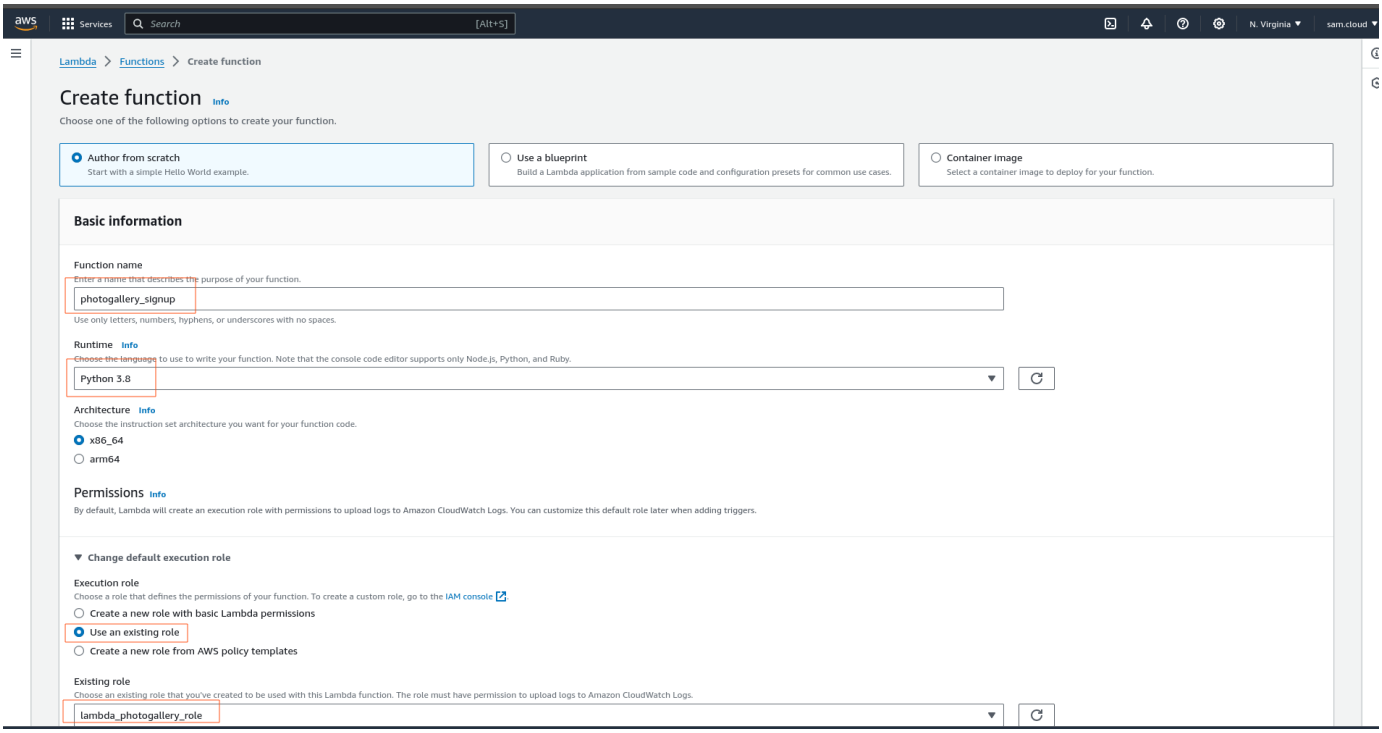


7. Create Lambda functions

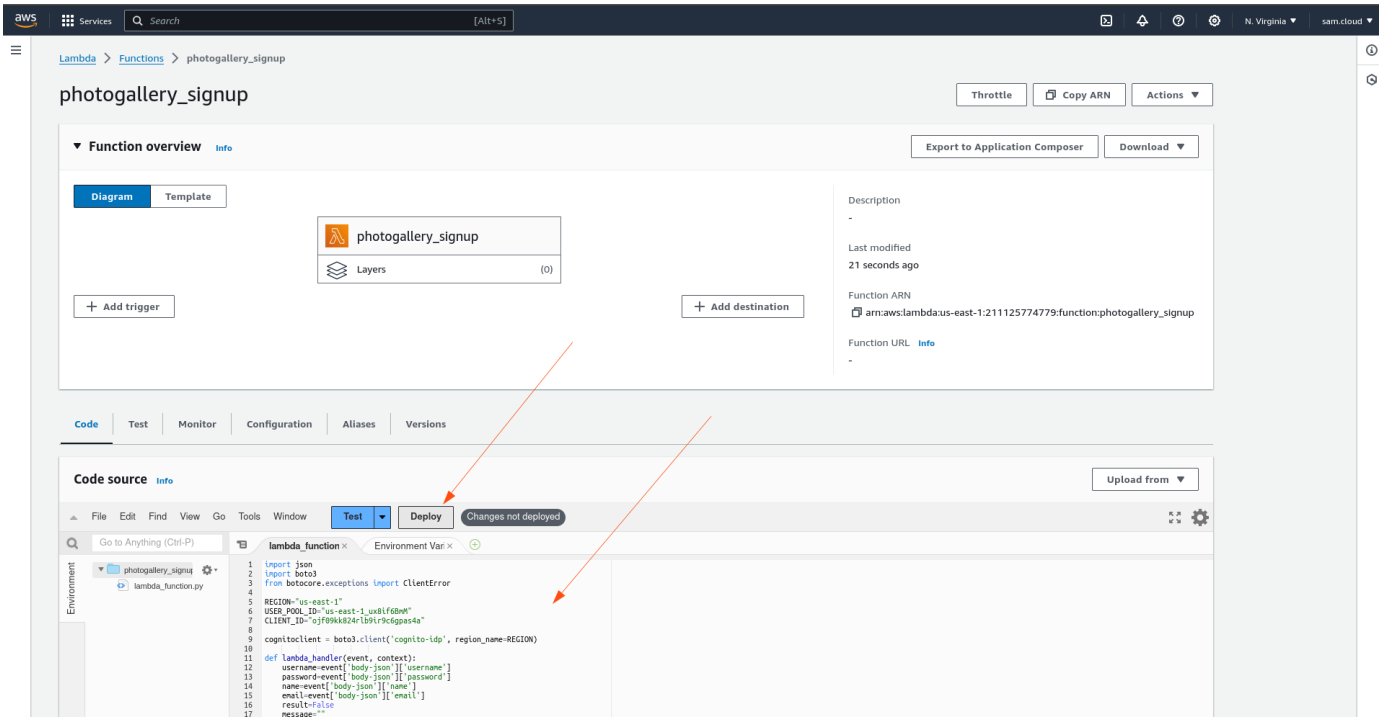
- We are going to create seven serverless functions under the AWS Lambda console. Each function follow the same process to deploy them. Please, follow each of the screenshots below.

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**"



- 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):

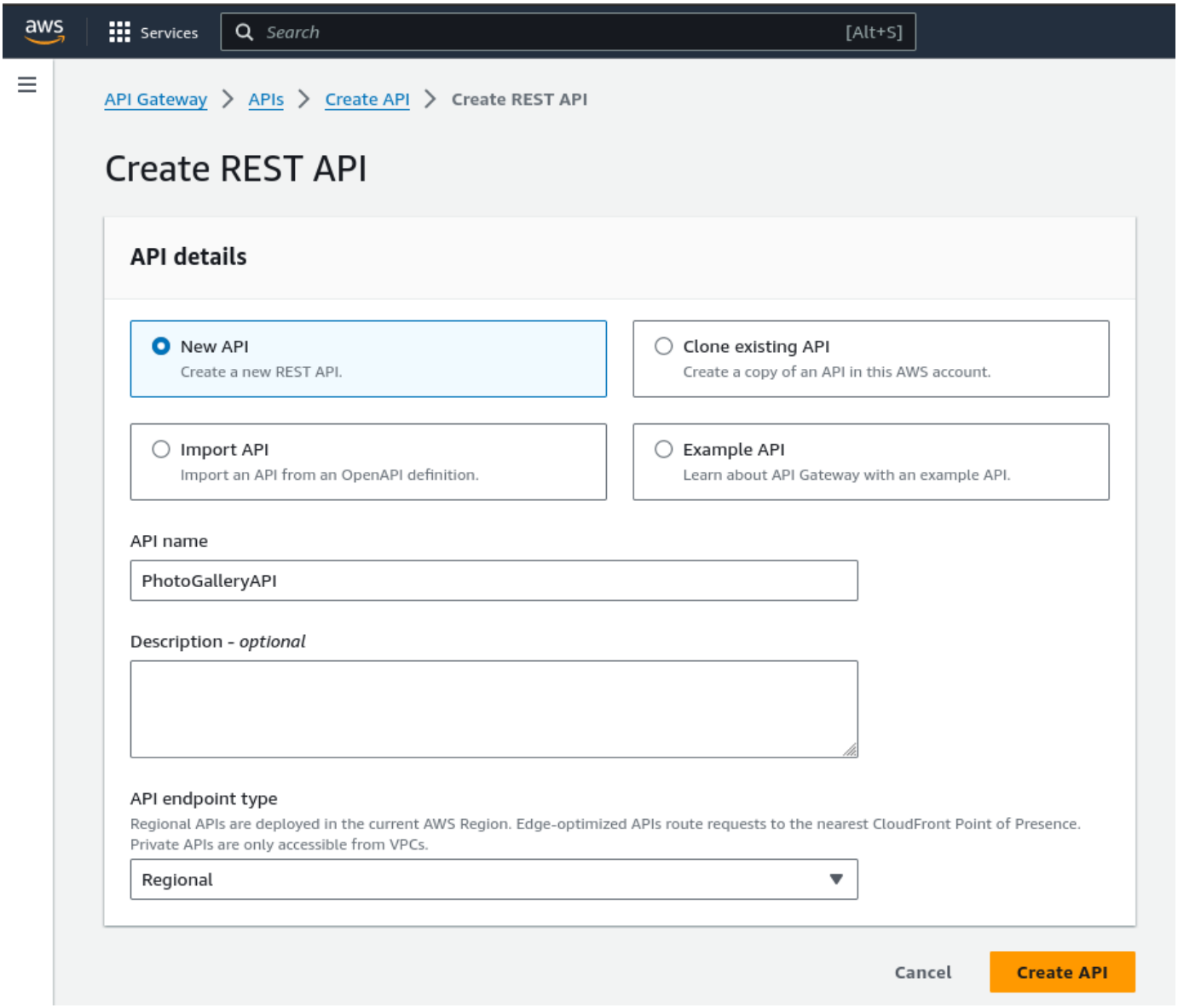
- Under AWS Lambda console, create a new function
- Give this function a name (**photogallery_addphoto**), attached the role previously created (**lambda_photogallery_role**) to this function, and then click "**Create function**"
- After creating the function, under "**Function Code**" copy and paste 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.

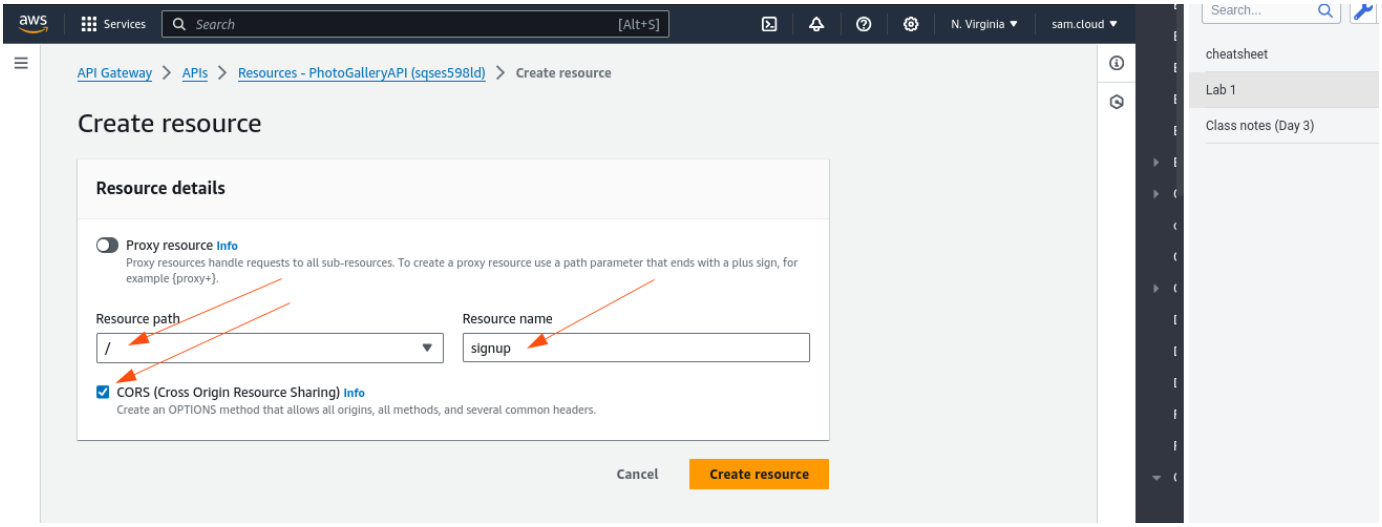


You have to prepare resources and methods for these items.

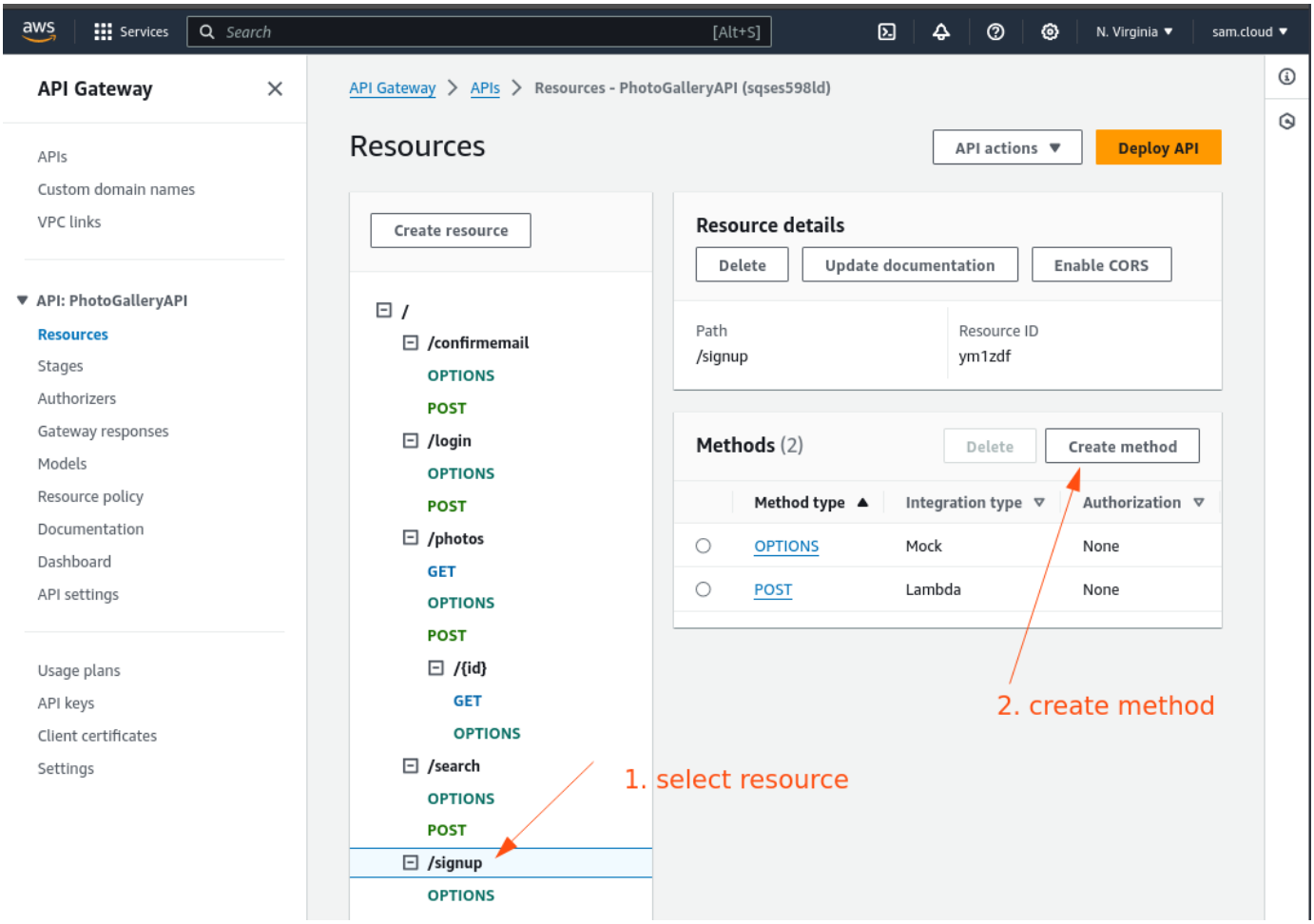
Resource	Method	Integration Request
/signup	POST	Lambda Function: photogallery_signup
/confirmemail	POST	Lambda Function: photogallery_confirmemail
/login	POST	Lambda Function: photogallery_login
/photos	POST	Lambda Function: photogallery_addphoto
/photos	GET	Lambda Function: photogallery_getphotos
/photos/{id}	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:

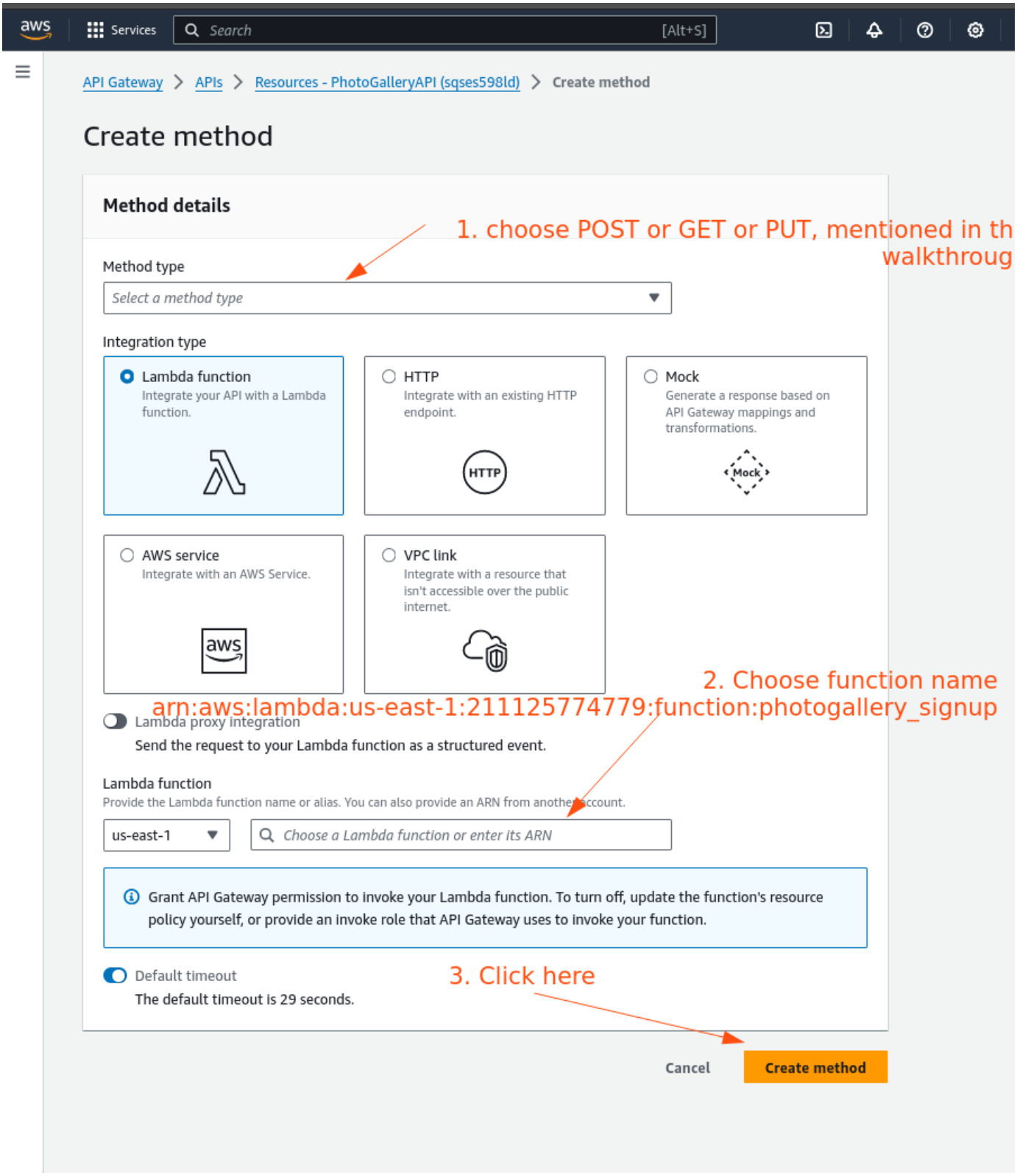
- Create a new **Resource**



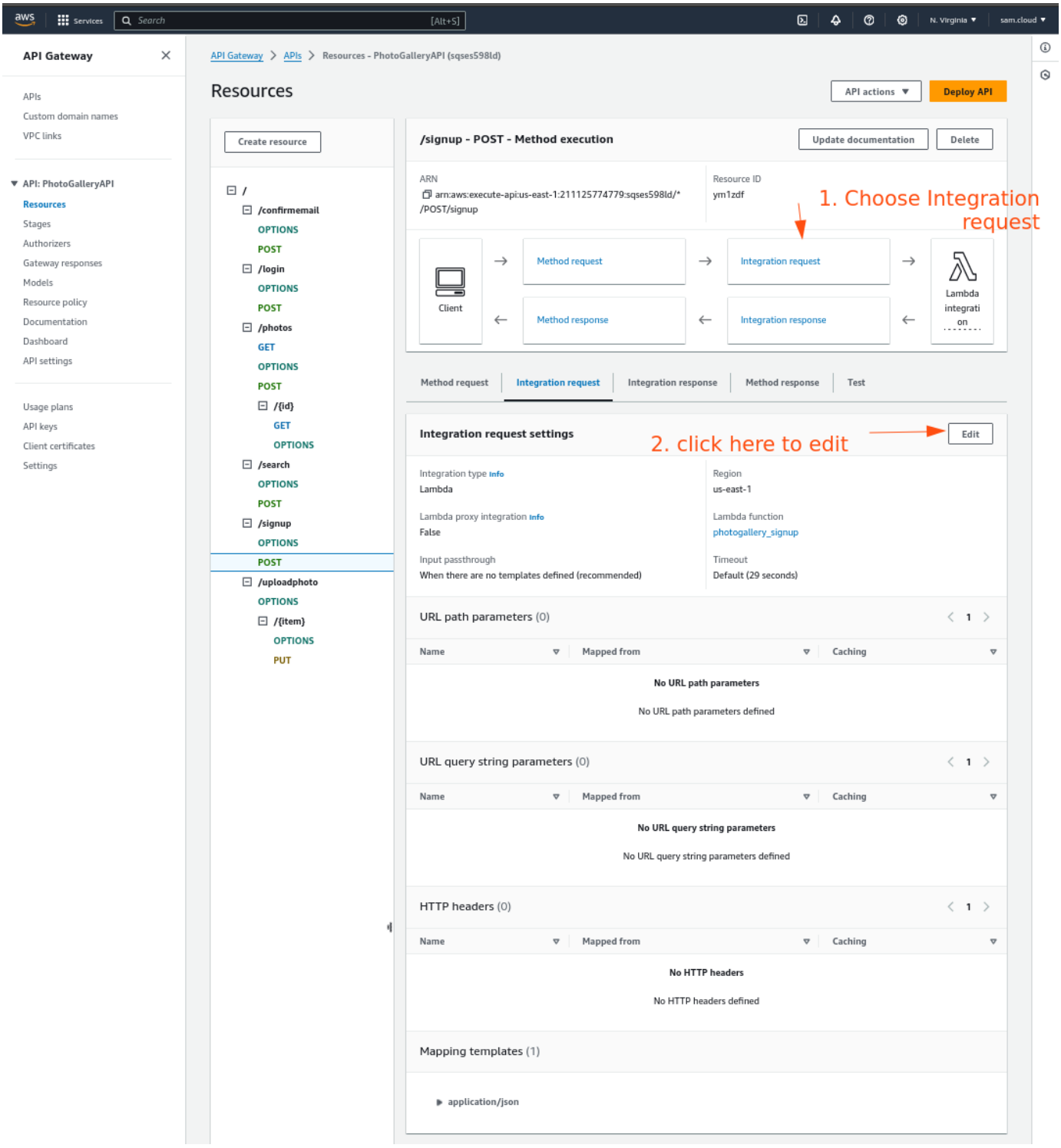
- Give a name to the resource and make sure that "**Enable API Gateway CORS**" is checked
- Once the resource is created, select the resource and create a new Method under that resource.

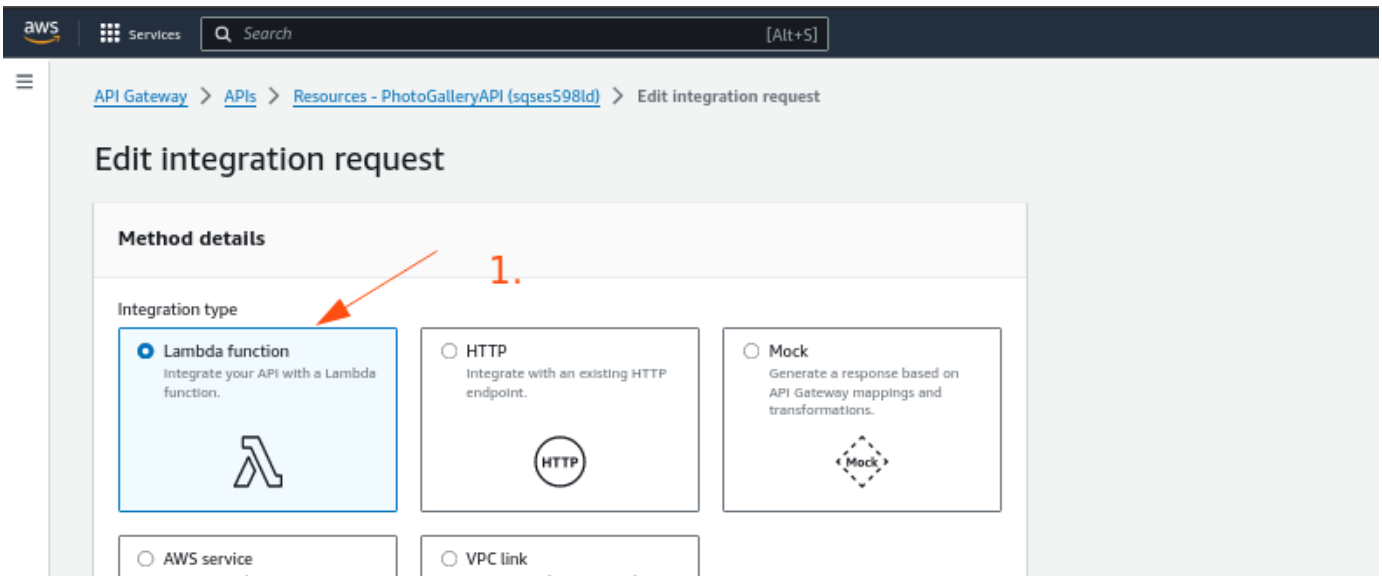


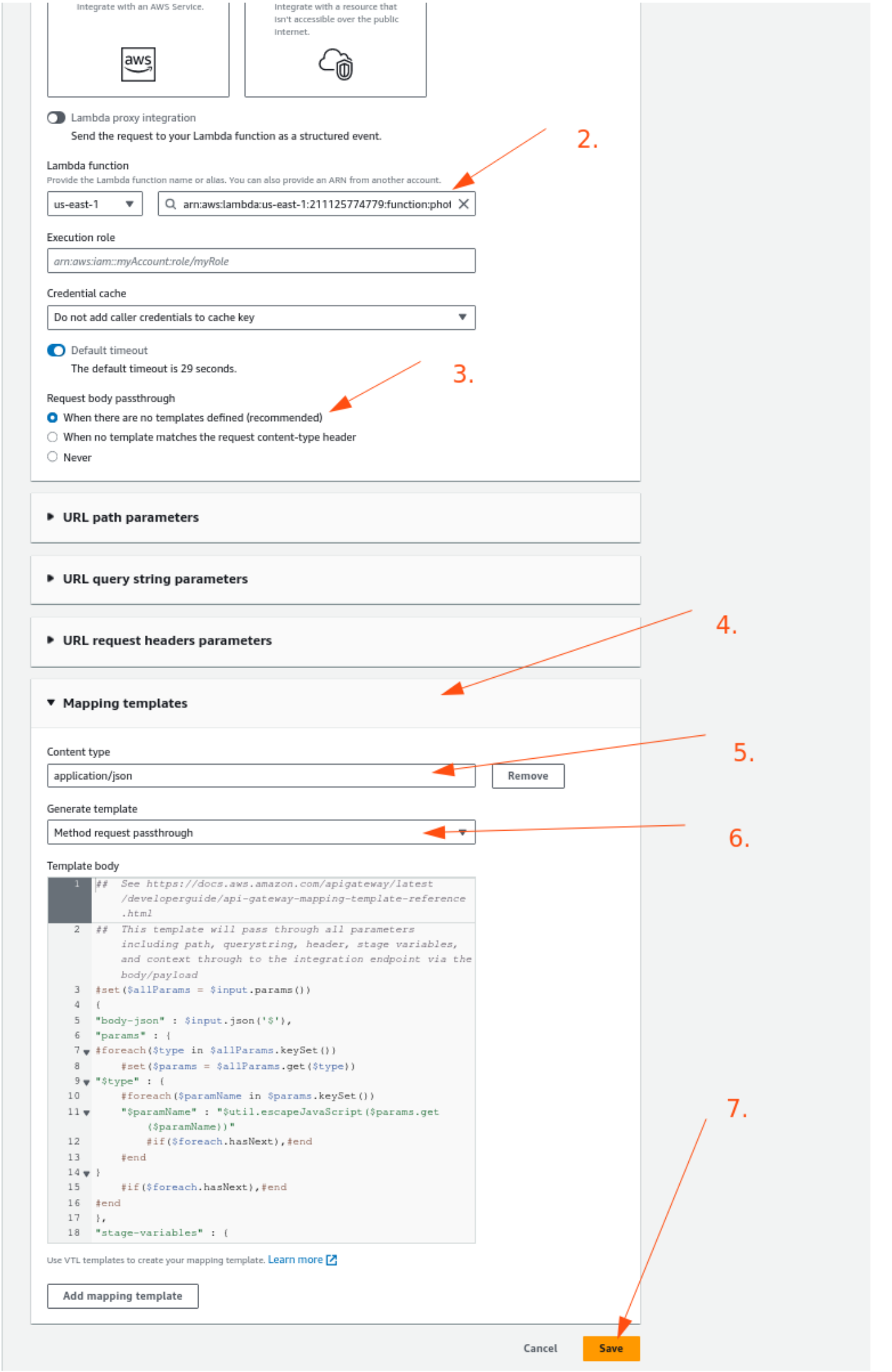
- Select POST as the method for this resource

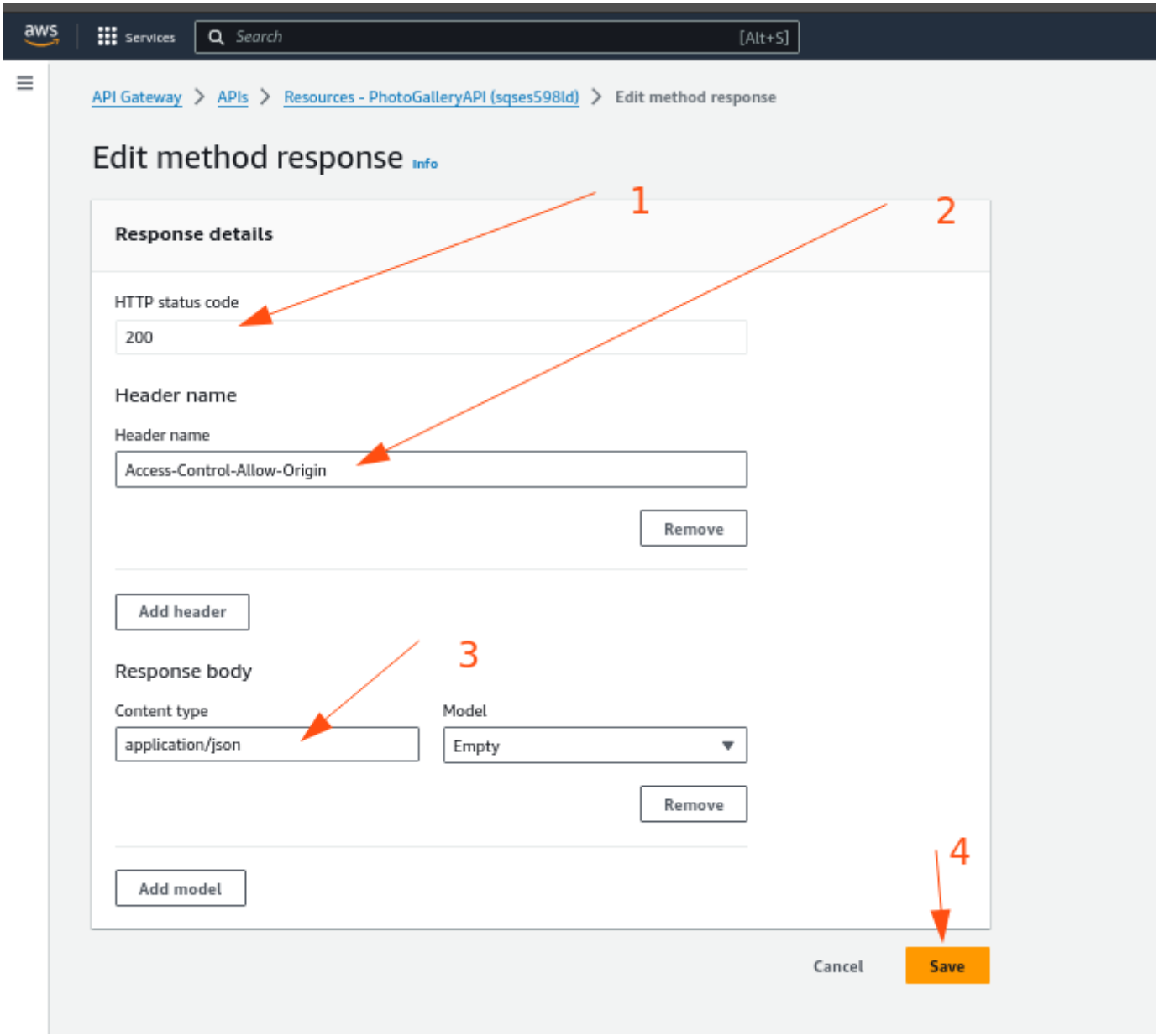


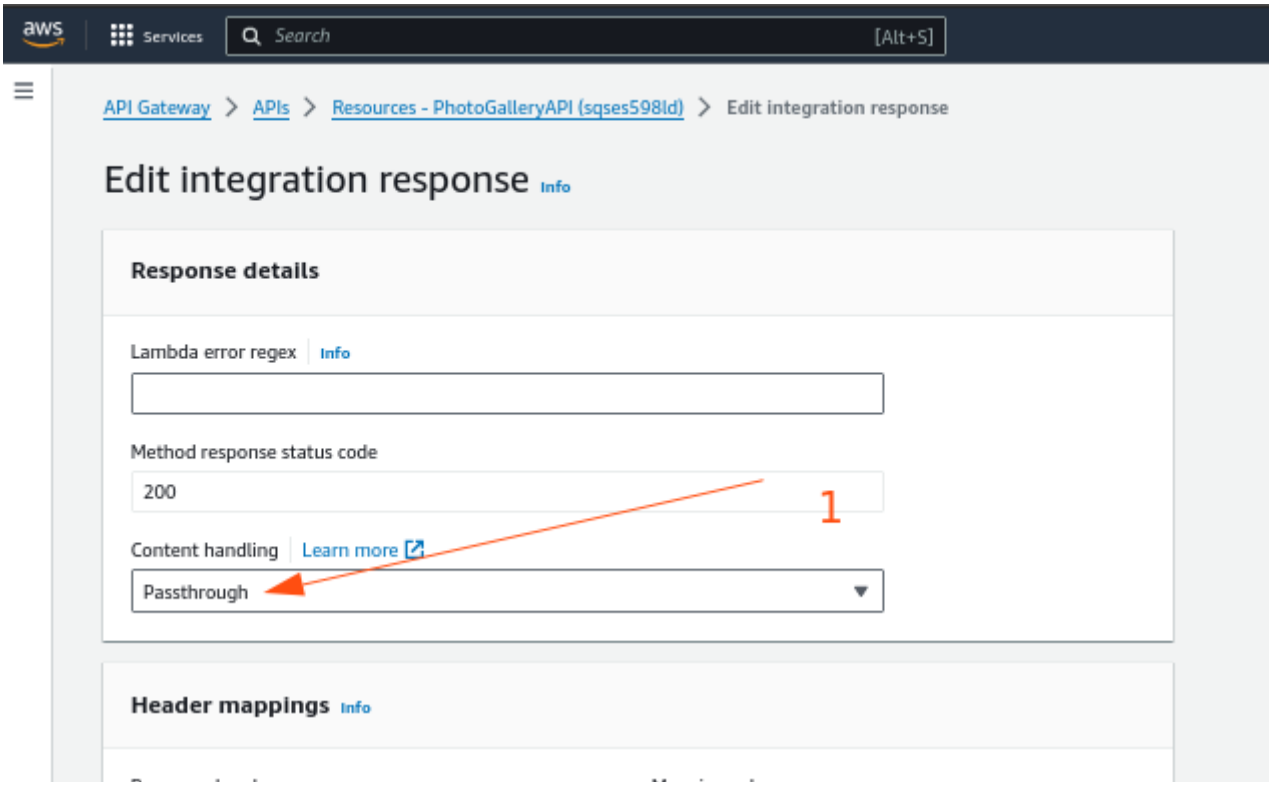
- Select the method created and attach the lambda function as the integration type.
- After saving the integration type, select “**Integration Request**” under the method. Under the “**Mapping Template**”, click “**Edit**” and “**Add Mapping Template**”. Add “**application/json**” to the mapping. Select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

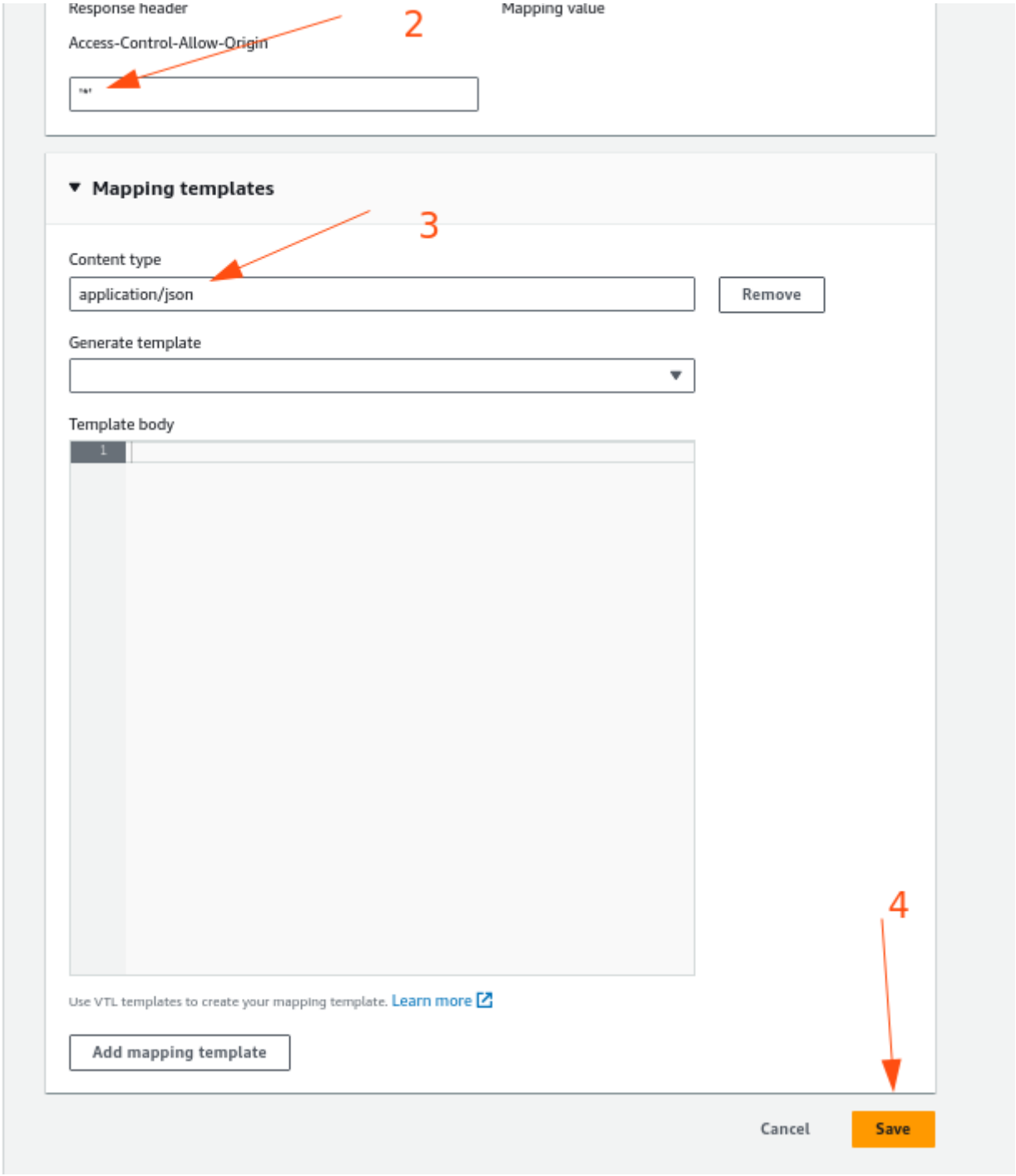

API Gateway
APIs
Custom domain names
VPC links
API: PhotoGalleryAPI
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
API keys
Client certificates
Settings
API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld)
Resources
API actions
Deploy API
Create resource
/
/confirmemail
OPTIONS
POST
/login
OPTIONS
POST
/photos
GET
OPTIONS
POST
/{id}
GET
OPTIONS
/search
OPTIONS
POST
/signup
OPTIONS
POST
/uploadphoto
OPTIONS
/{item}
OPTIONS
PUT
/signup - POST - Method execution
Update documentation
Delete
ARN
arn:aws:execute-api:us-east-1:211125774779:sqses598ld/*/POST/signup
Resource ID
ym1zdf
1. Choose Integration request
Client
Method request
Integration request
Lambda integration
Method response
Integration response
Method request
Integration request
Integration response
Method response
Test
Integration request settings
2. click here to edit
Edit
Integration type Info
Lambda
Region
us-east-1
Lambda proxy integration Info
False
Lambda function
photogallery_signup
Input passthrough
When there are no templates defined (recommended)
Timeout
Default (29 seconds)
URL path parameters (0)
Name
Mapped from
Caching
No URL path parameters
No URL path parameters defined
URL query string parameters (0)
No URL query string parameters
No URL query string parameters defined
HTTP headers (0)
No HTTP headers
No HTTP headers defined
Mapping templates (1)
application/json


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.
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.
2.
Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.
us-east-1
arn:aws:lambda:us-east-1:211125774779:function:phot
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.
3.
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
4.
Mapping templates
Content type
application/json
Remove
5.
Generate template
Method request passthrough
6.
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 $params.keySet())
11 "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
12 #if($foreach.hasNext),#end
13 #end
14 }
15 #if($foreach.hasNext),#end
16 #end
17 },
18 "stage-variables" : {
Use VTL templates to create your mapping template. Learn more
Add mapping template
7.
Cancel
Save


aws
Services
Search
[Alt+S]
API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Edit method response
Edit method response Info
1
2
Response details
HTTP status code
200
Header name
Header name
Access-Control-Allow-Origin
Remove
Add header
3
Response body
Content type
application/json
Model
Empty
Remove
4
Add model
Cancel
Save


aws
Services
Search
[Alt+S]
API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Edit integration response
Edit integration response Info
Response details
Lambda error regex Info
Method response status code
200
1
Content handling Learn more
Passthrough
Header mappings Info

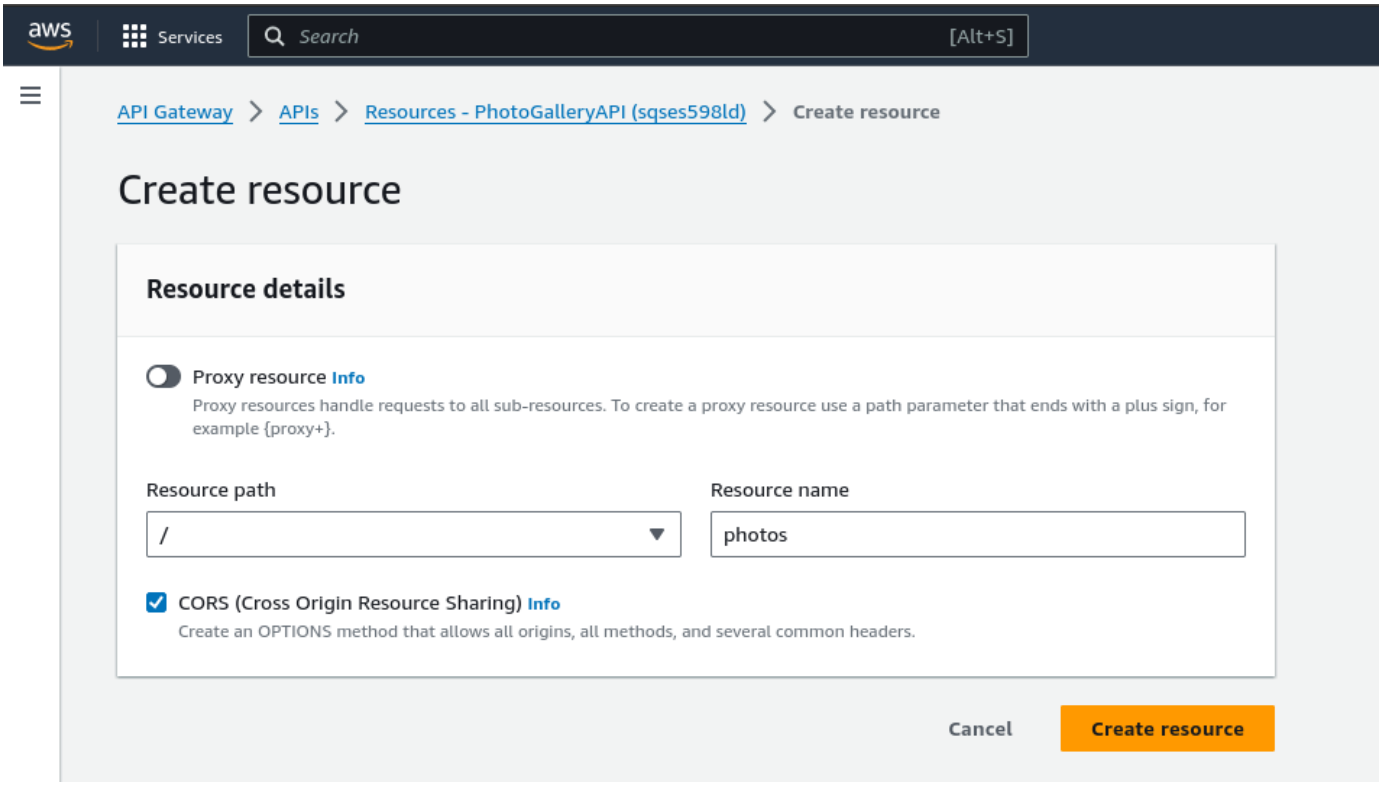


To create another resource, click “/“ under the Resources column

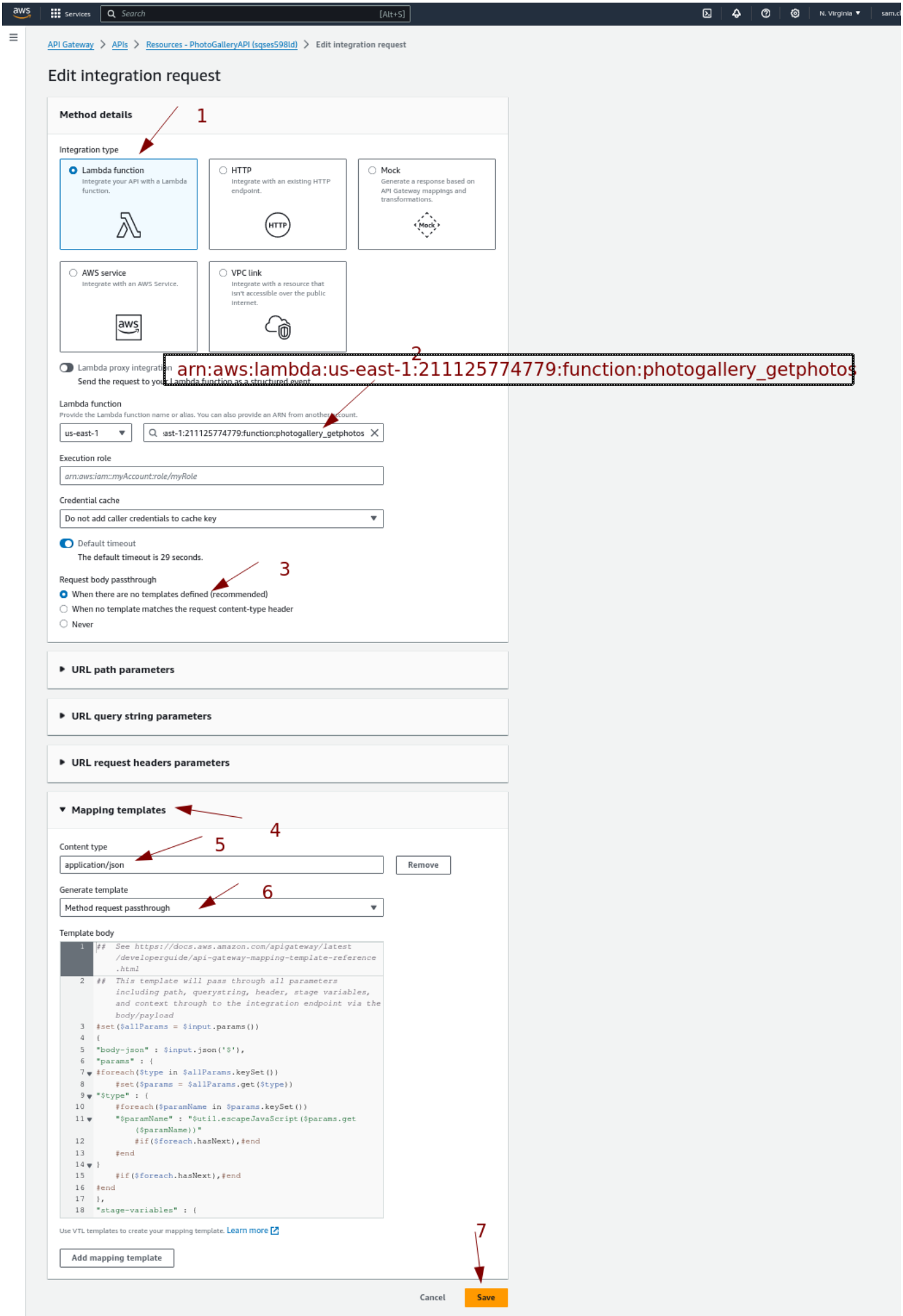
Repeat the same process for /confirmemail, /login, and /search

For */* , a GET request a lambda function:

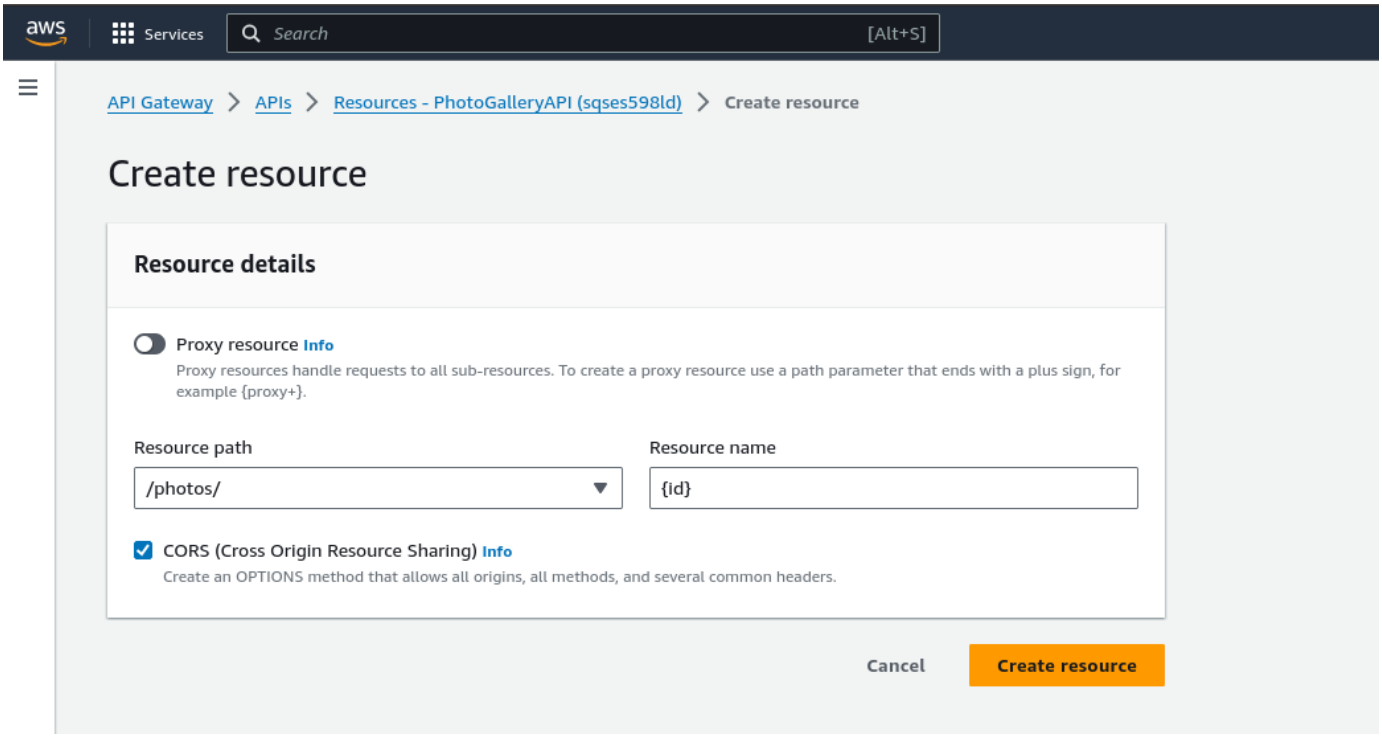
- Create a new **Resource** under the **Actions** dropdown button. Give a name to the resource and make sure that “**Enable API Gateway CORS**” is checked. Set the mapping template for all the method created below.

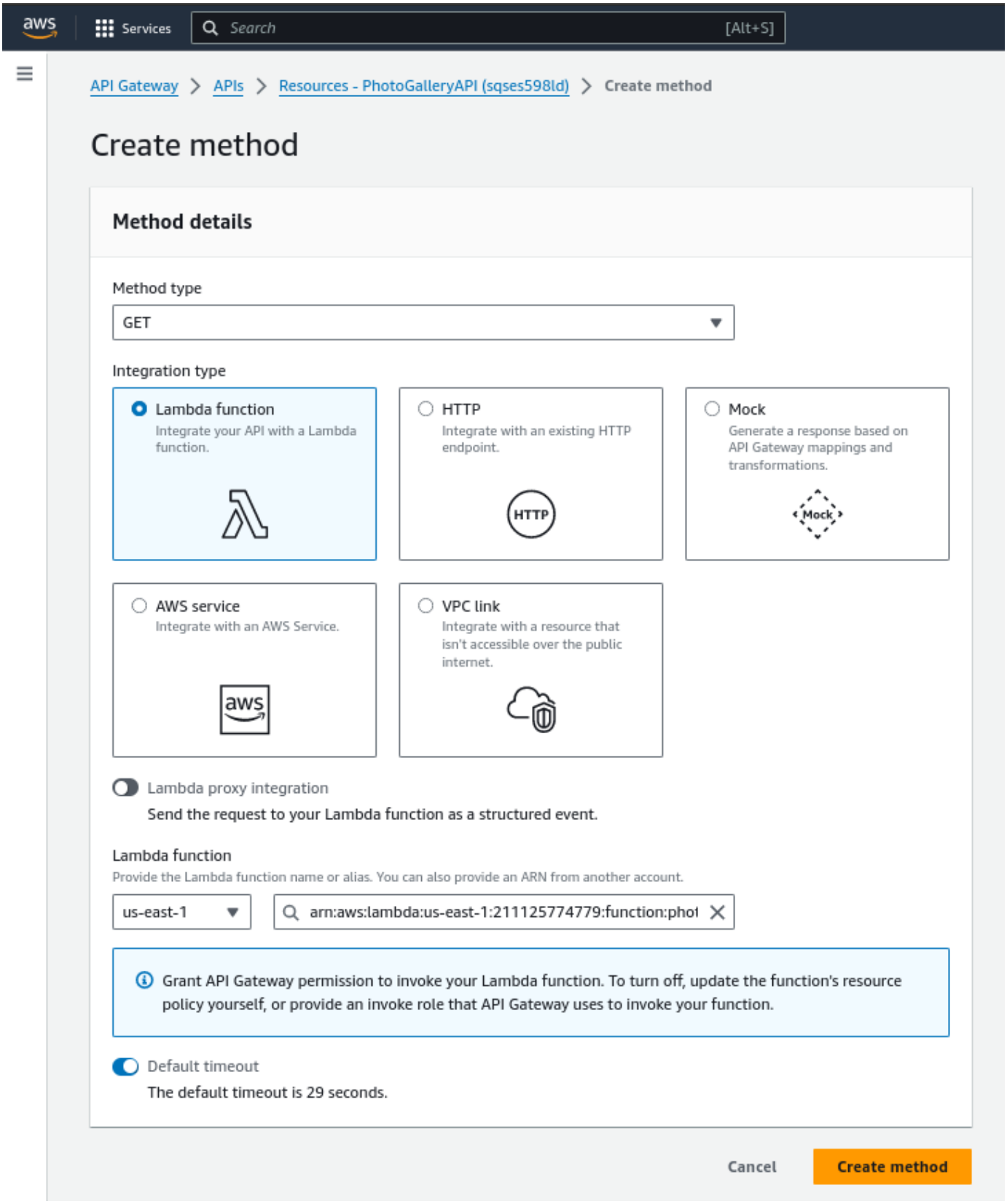


- Once the resource is created, select the resource and create a new **Method** under that resource.
- Select **GET** as the method for this resource. Select the method created and attach the lambda function as the integration type. (**getphotos**)

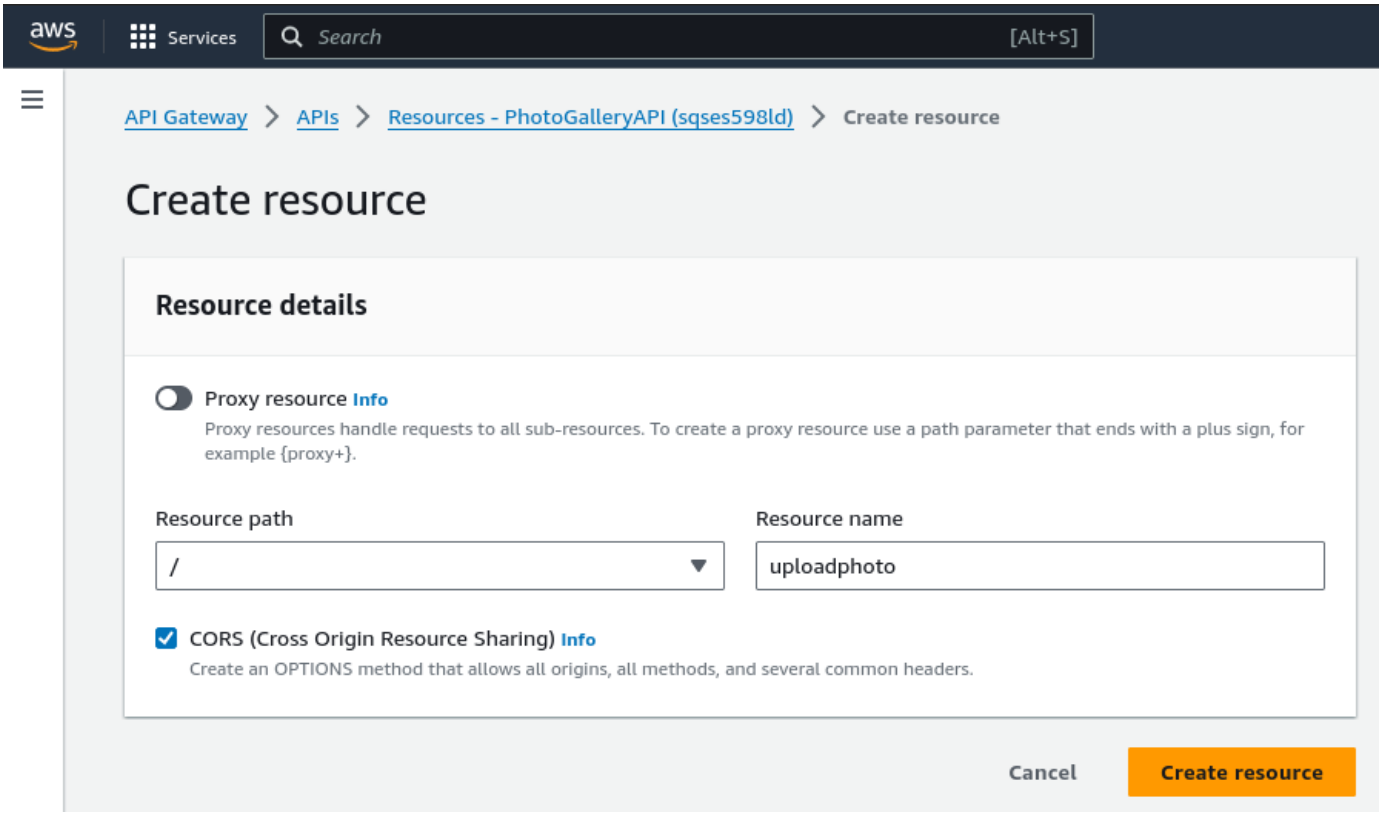

aws
Services
Search
[Alt+S]
N. Virginia
API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598id) > Edit integration request
Edit integration request
Method details
1
Integration type
Lambda function
Integrate your API with a Lambda function.
HTTP
Integrate with an existing HTTP endpoint.
Mock
Generate 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.
2
arn:aws:lambda:us-east-1:211125774779:function:photogallery_getphotos
Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.
us-east-1
ast-1:211125774779:function:photogallery_getphotos
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.
3
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
4
Content type
5
application/json
Remove
Generate template
6
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 $params.keySet())
11 "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
12 #if($foreach.hasNext),#end
13 #end
14 }
15 #if($foreach.hasNext),#end
16 #end
17 },
18 "stage-variables" : {
Use VTL templates to create your mapping template. Learn more
Add mapping template
7
Cancel
Save

- 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.

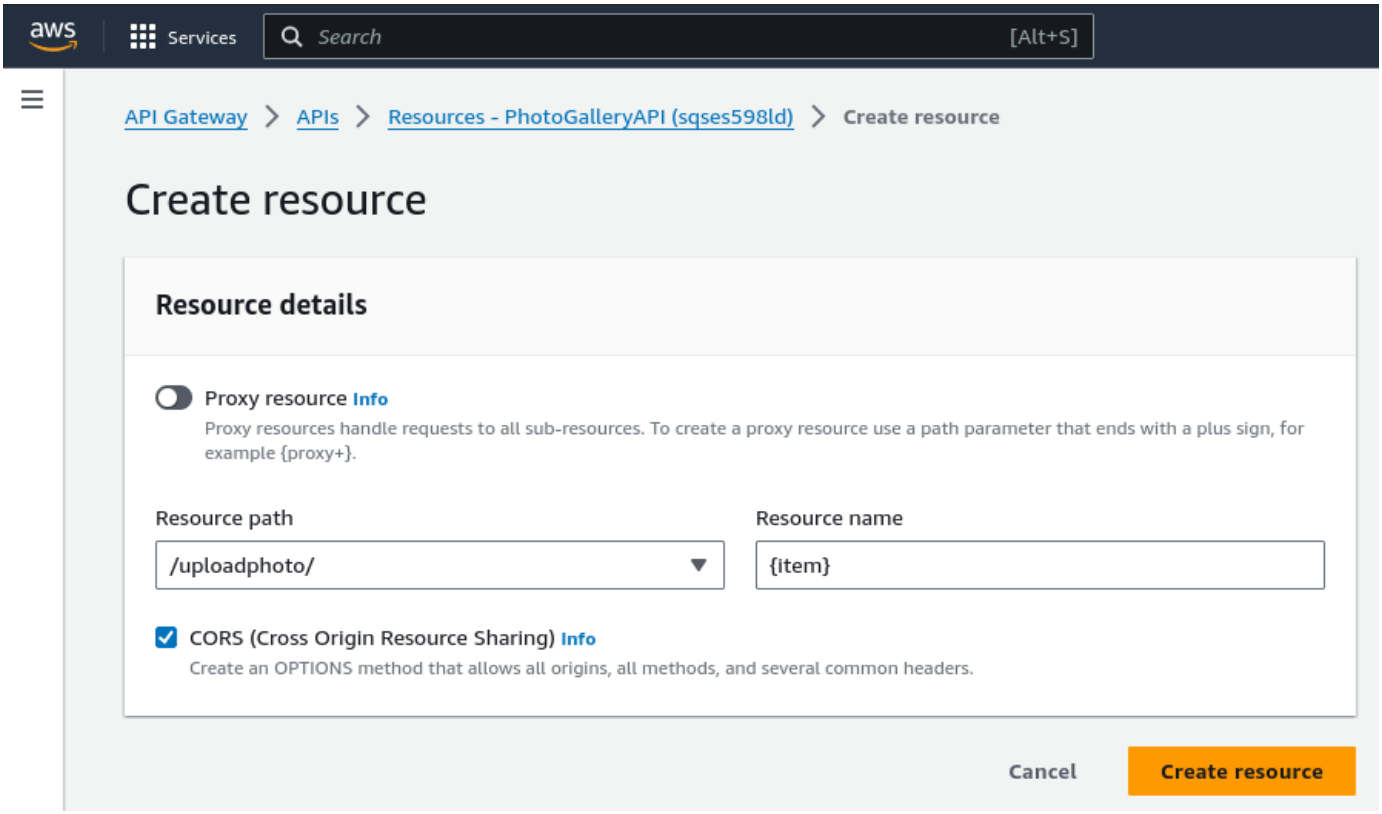




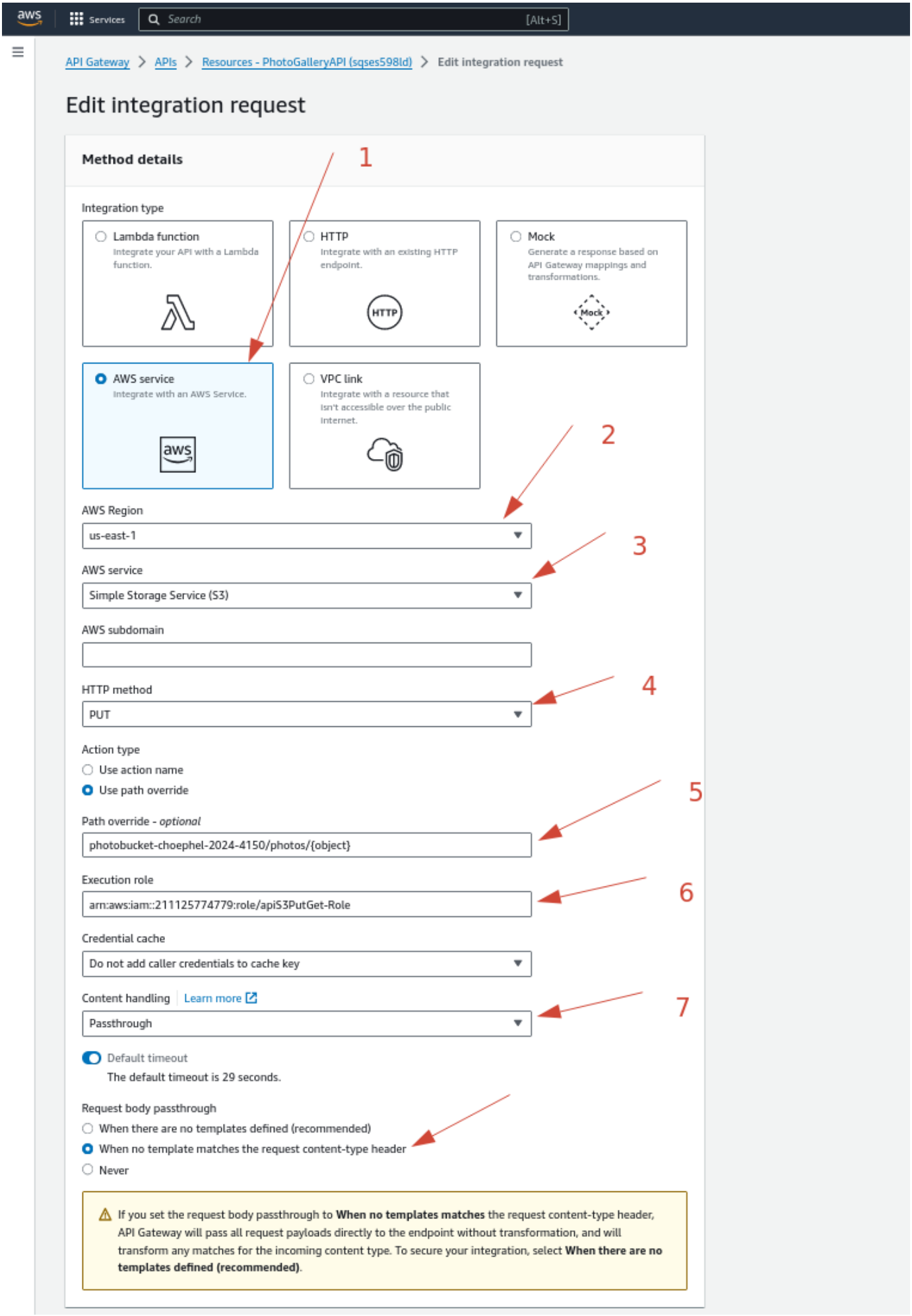
- After saving the integration type, select "**Integration Request**", click **Edit**. Choose "**When there are no templates defined (recommended)**" under the Request body passthrough, under the "**Mapping Template**",. Under the Content-Type, add "**application/json**" to the mapping. Finally, Under 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

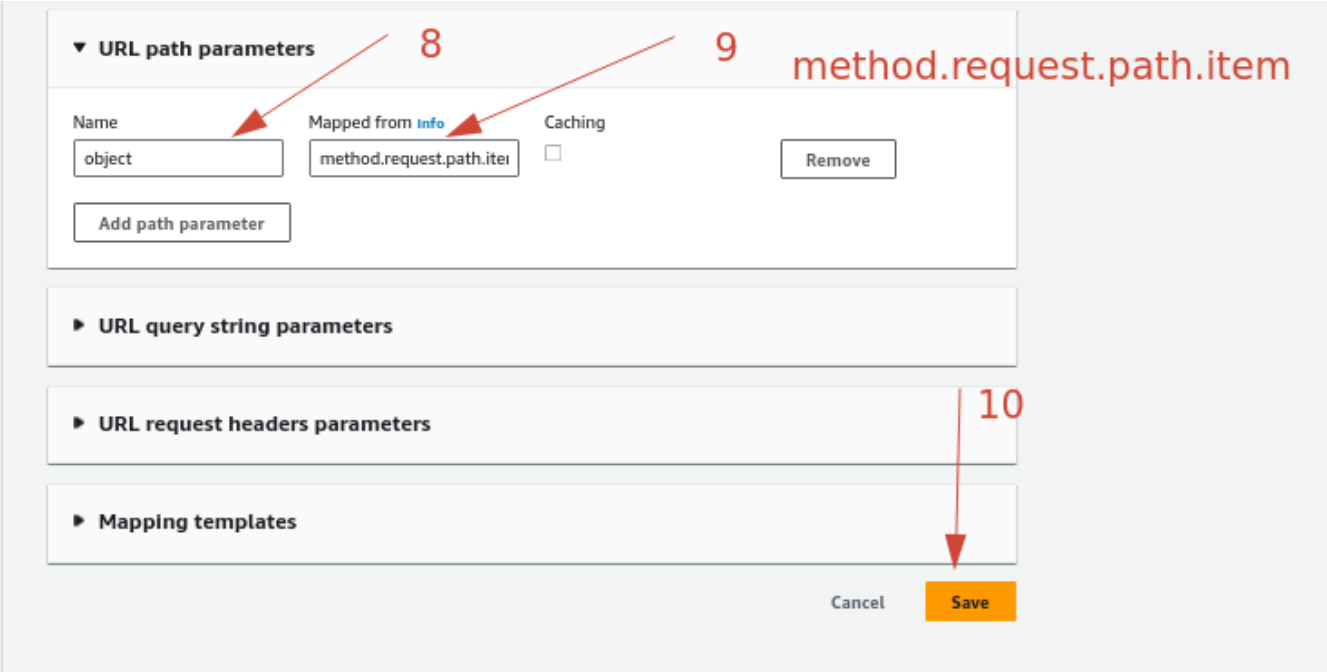


Within the same resource, create another resource as shown below.



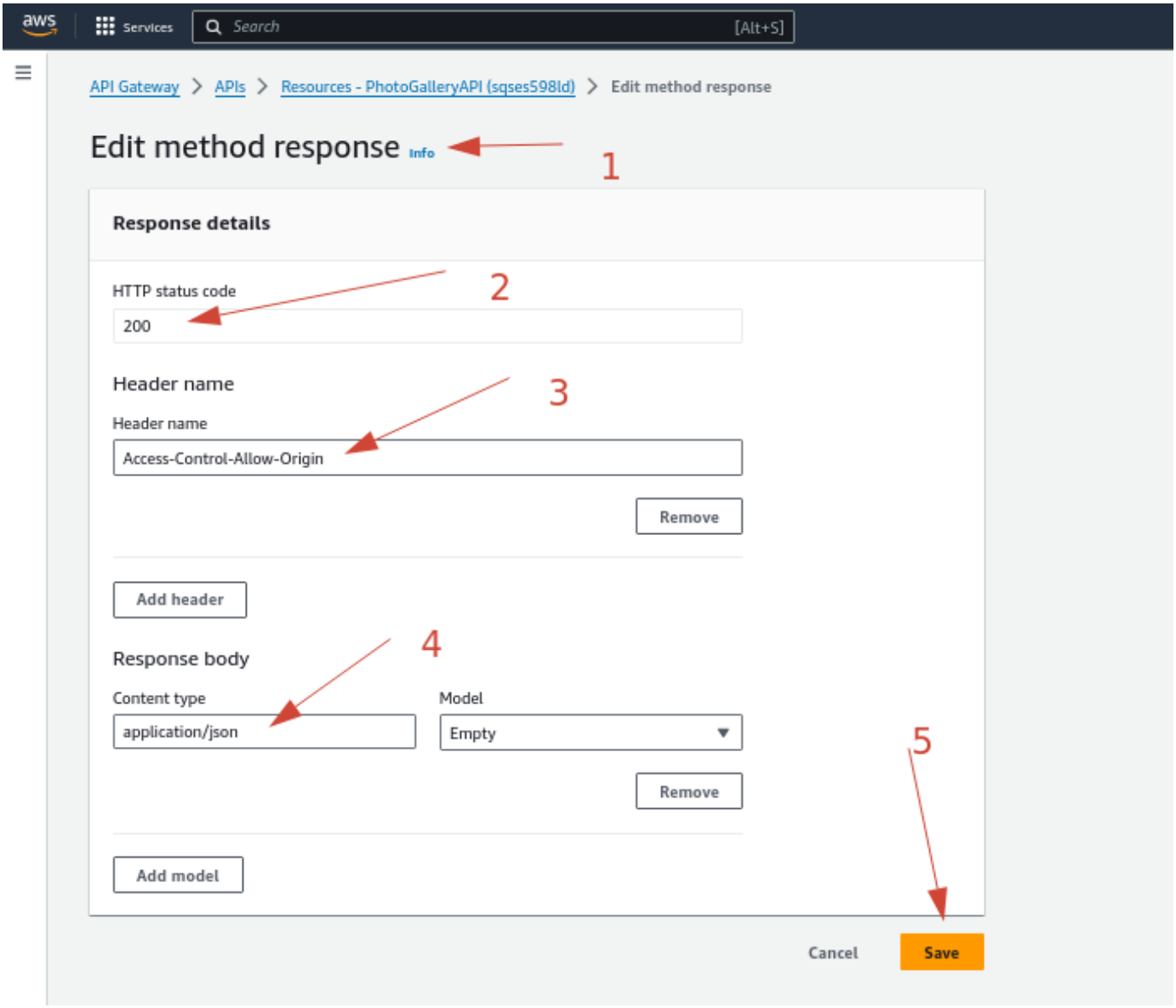
Under the new created resource, create a new **PUT** method.

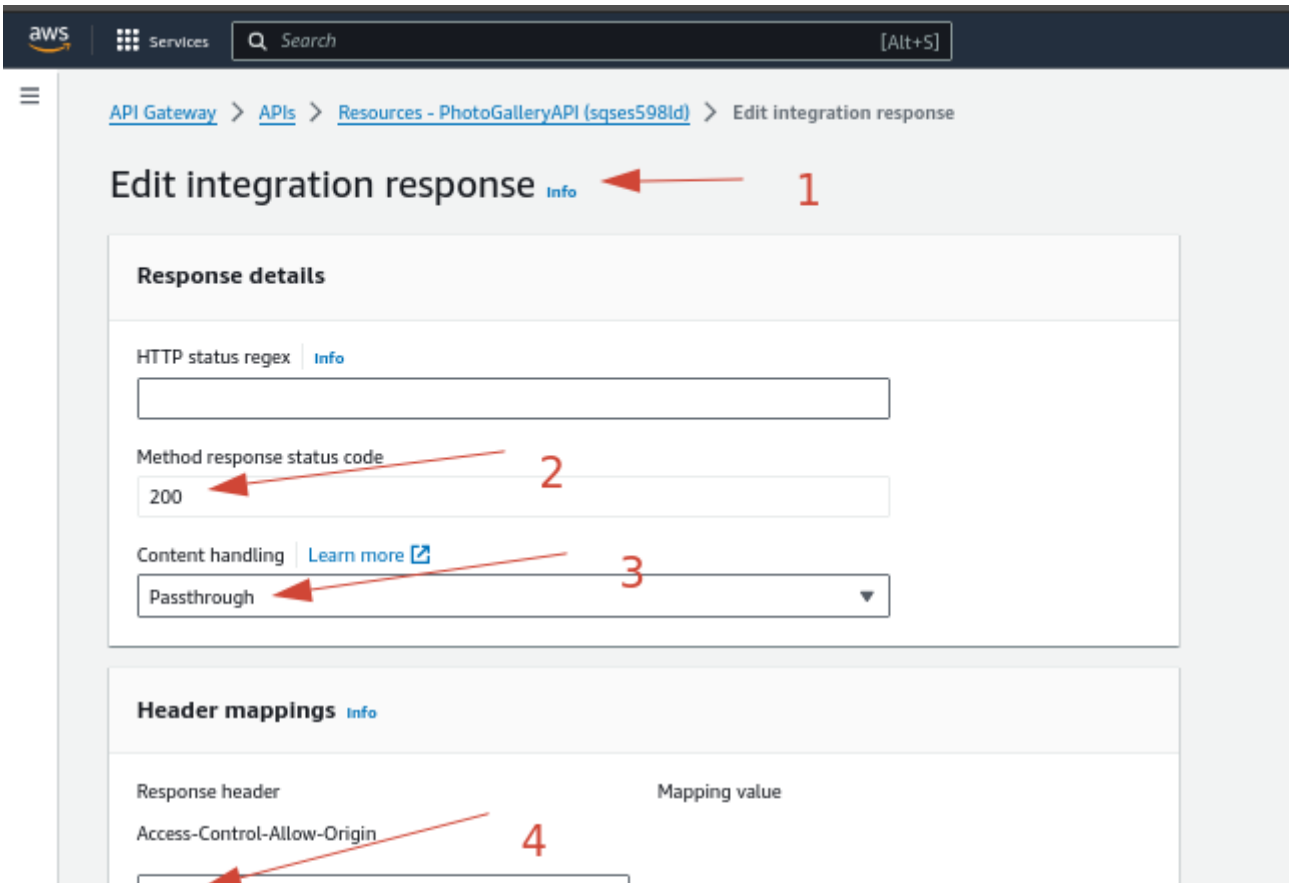


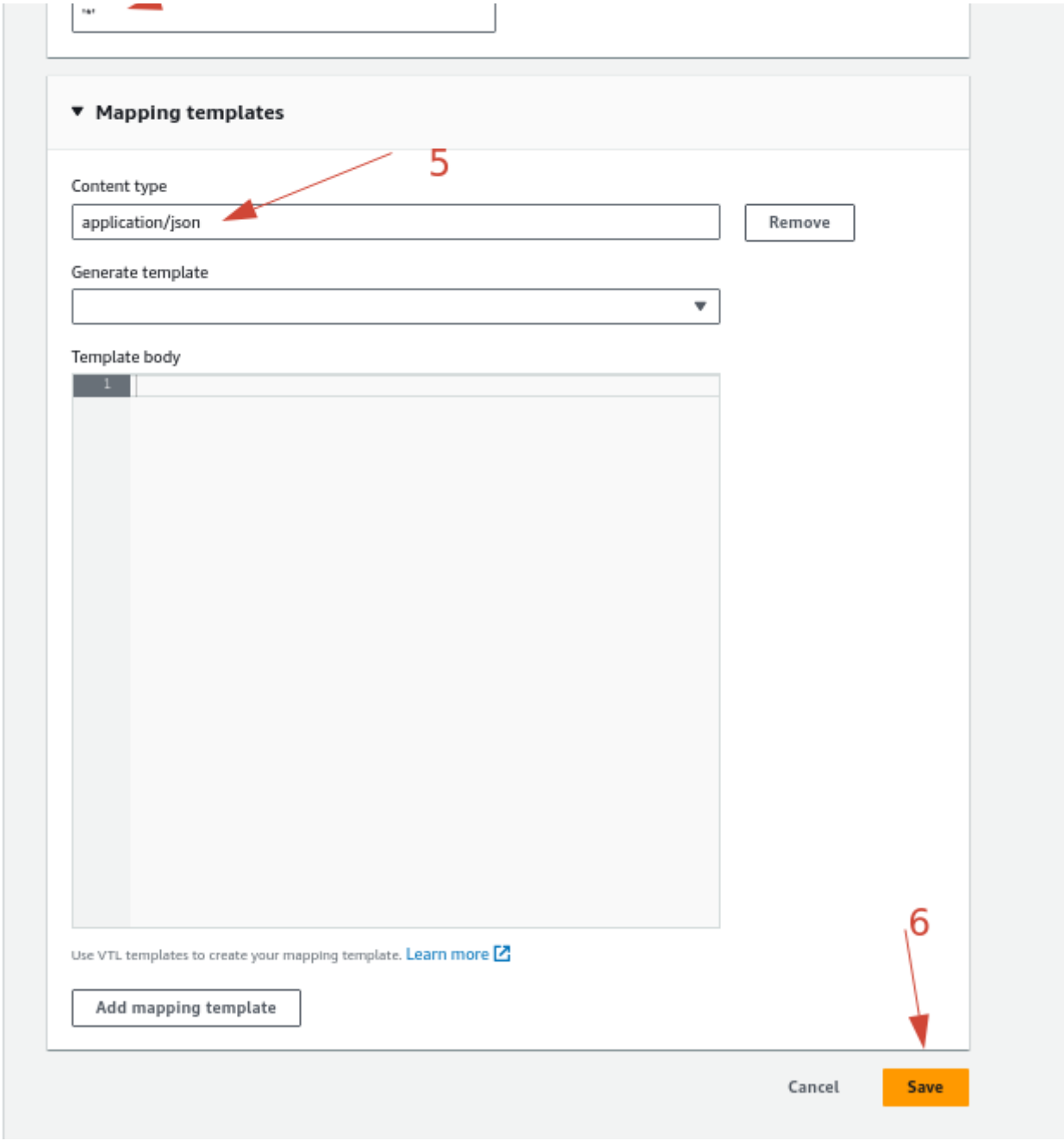


- Select the method created and select the **AWS Region** where the S3 bucket was created (in this case, is **us-east-1** or **N. Virginia**). Under **AWS Service**, choose **Simple Storage Service (S3)**. Under **Action Type**, select "**Use path override**" and add the path override; in this case, 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.

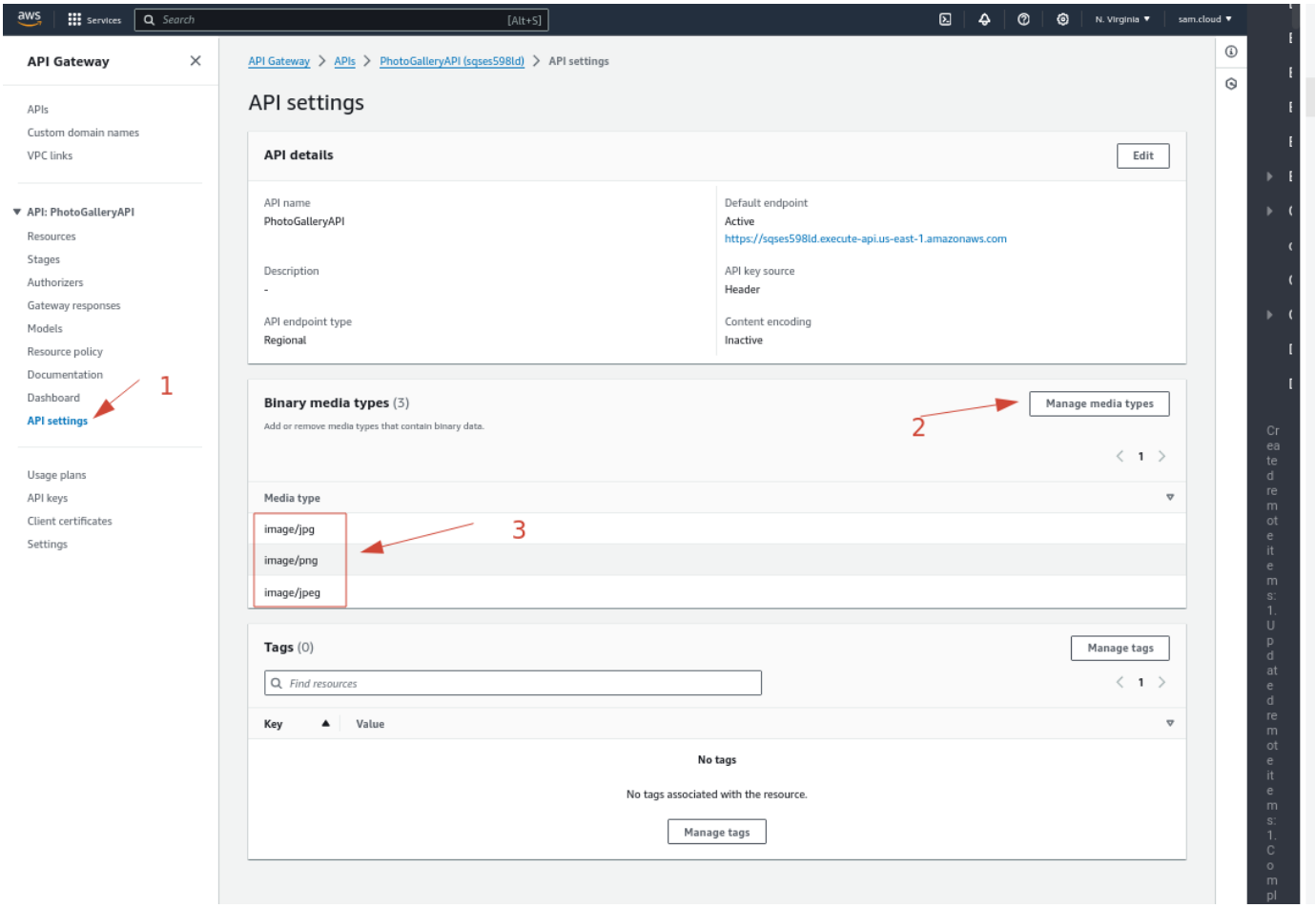
Once the method is created, select the **Integration Request** under the method and create a mapping configuration as shown above.


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
Remove
Add header
Response body
4
Content type
application/json
Model
Empty
Remove
5
Add model
Cancel
Save

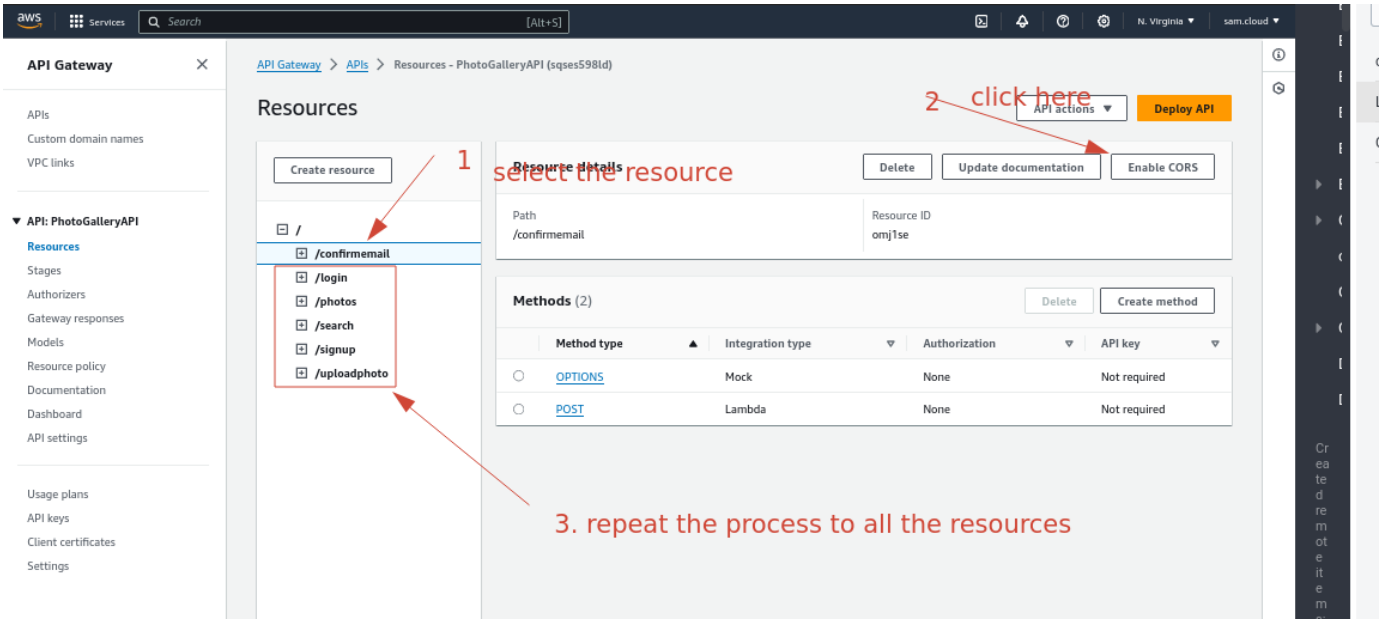

aws
Services
Search
[Alt+S]
API Gateway > APIs > Resources - PhotoGalleryAPI (sqses598ld) > Edit integration response
Edit integration response Info
1
Response details
HTTP status regex Info
Method response status code
2
200
Content handling Learn more
3
Passthrough
Header mappings Info
Response header
Mapping value
Access-Control-Allow-Origin
4

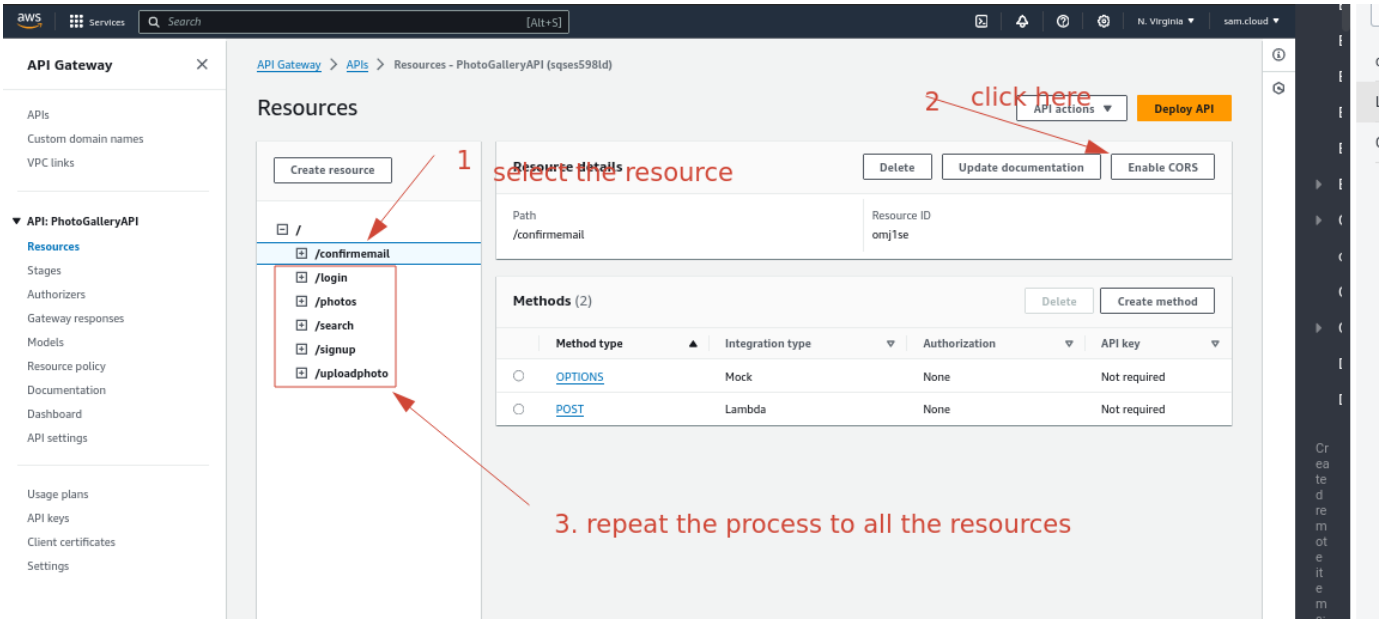


- To allow images to be uploaded to the S3 bucket, we need to add the binary media types. For this case, we are going to allow **JPEG**, **JPG**, and **PNG** to be uploaded to the bucket. To do that, select **API Setting** in the left column and add the media types under **Binary Media Types**, as shown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

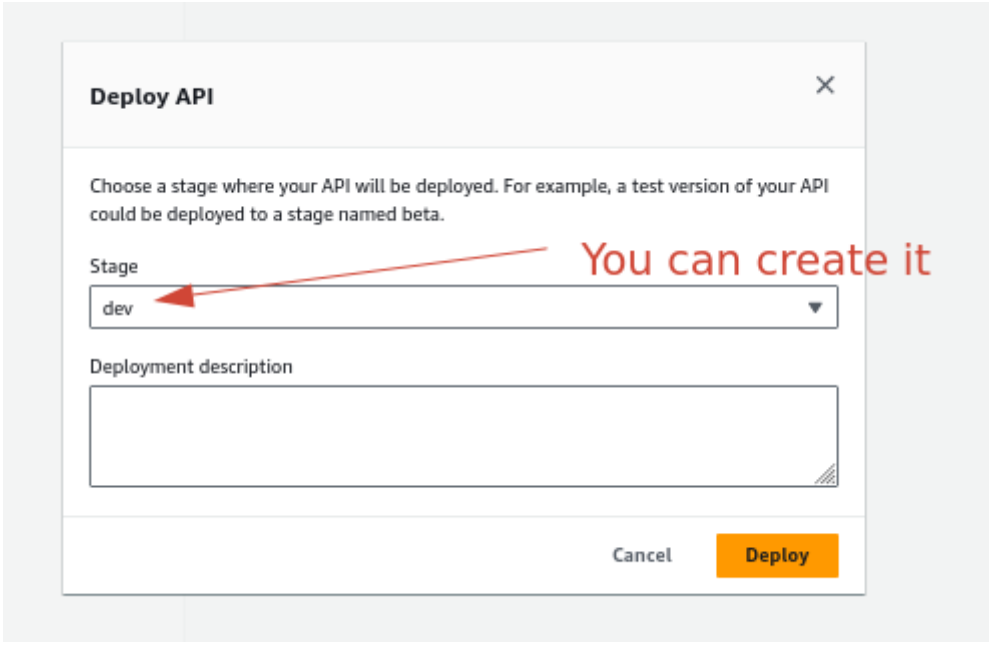


- Make sure to **enable CORS** for the resources. To do that, select the resources and click the **Actions** dropdown and select "**Enable CORS**".





- To deploy the API, click "**Deploy API**", as shown below

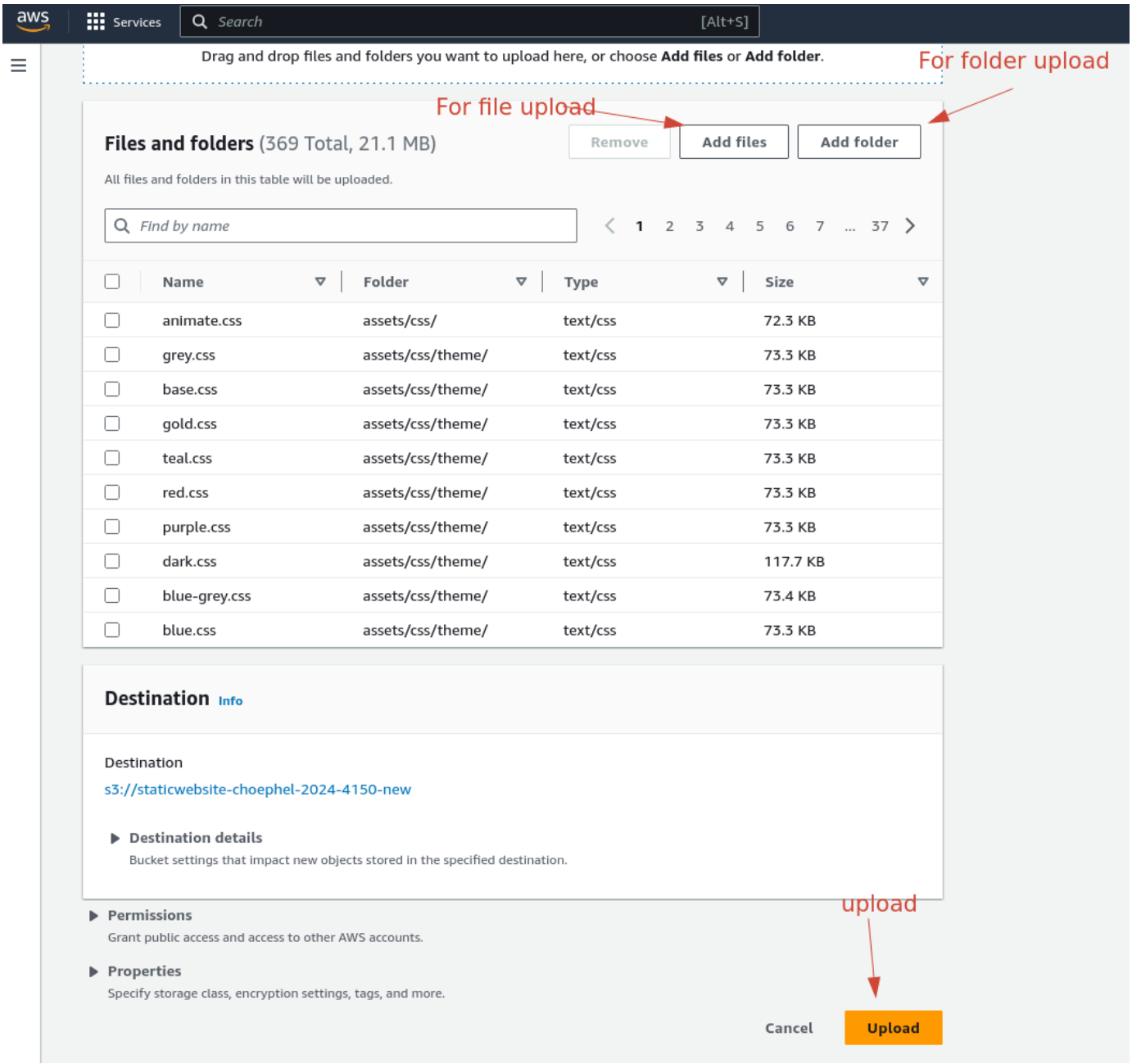


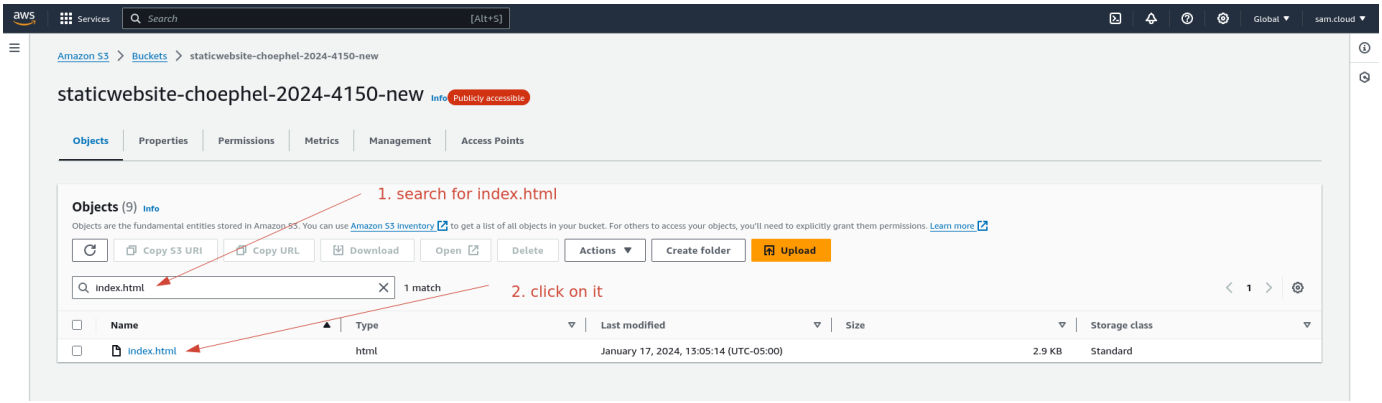
- Create a new stage for deployment and called it "**dev**" as shown below
- Copy and save the API URL; will use this URL to access the resources from the website.

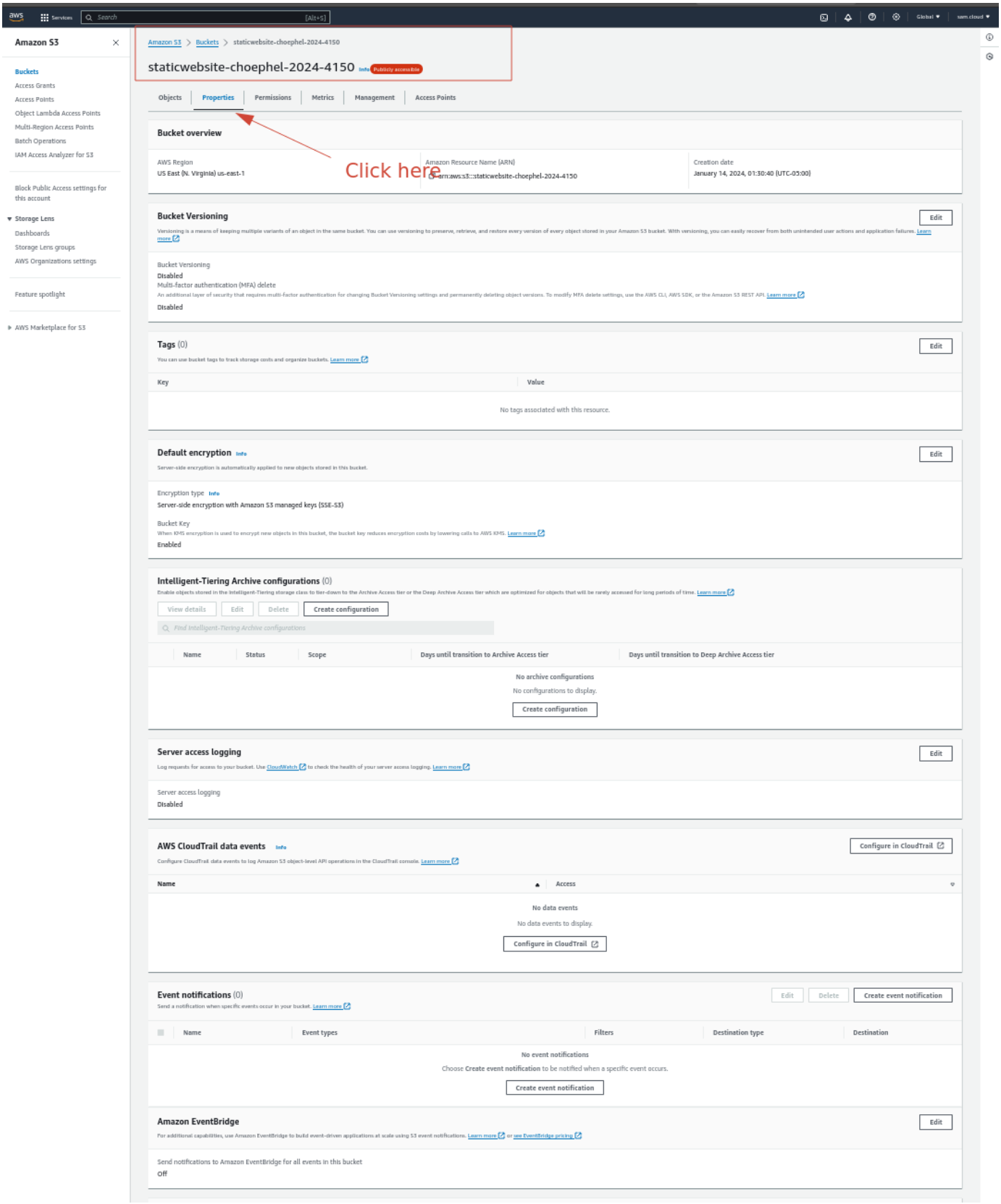
9. Updating the website

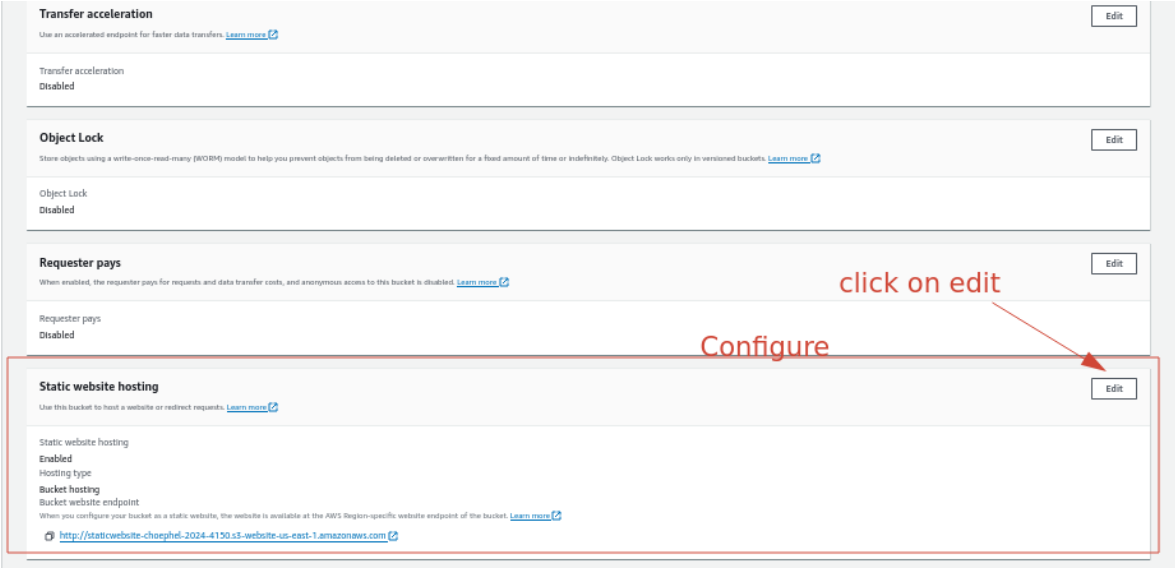
Add some resources to script used for the website. Update the **process.js** file under the staticwebsite/assets directory. Add the **<API_URL>** and **<PHOTOGALLERY_S3_BUCKET_NAME>** in lines 1 and 2, respectively, as shown below

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




Amazon S3 > Buckets > staticwebsite-choephel-2024-4150-new
staticwebsite-choephel-2024-4150-new Info Publicly accessible
Objects Properties Permissions Metrics Management Access Points
1. search for index.html
Objects (9) Info
Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload
index.html
1 match
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


Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3 > Buckets > staticwebsite-choephel-2024-4150
staticwebsite-choephel-2024-4150 Info Publicly accessible
Objects Properties Permissions Metrics Management Access Points
Click here
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
Edit
Bucket Versioning
Disabled
Multi-factor authentication (MFA) delete
Disabled
Tags (0)
Edit
Key
Value
No tags associated with this resource.
Default encryption Info
Edit
Encryption type Info
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Bucket Key
Enabled
Intelligent-Tiering Archive configurations (0)
View details Edit Delete Create configuration
Name Status Scope Days until transition to Archive Access tier Days until transition to Deep Archive Access tier
No archive configurations
No configurations to display.
Create configuration
Server access logging
Edit
Server access logging
Disabled
AWS CloudTrail data events Info
Configure in CloudTrail
Name Access
No data events
No data events to display.
Configure in CloudTrail
Event notifications (0)
Edit Delete Create event notification
Name Event types Filters Destination type Destination
No event notifications
Choose Create event notification to be notified when a specific event occurs.
Create event notification
Amazon EventBridge
Edit
Send notifications to Amazon EventBridge for all events in this bucket
Off



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:

1. Competition of the main functionalities of the website as described in the instructions **(80 points)**
2. Allow users to delete a picture in their own gallery **(10 Points)**
3. 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:

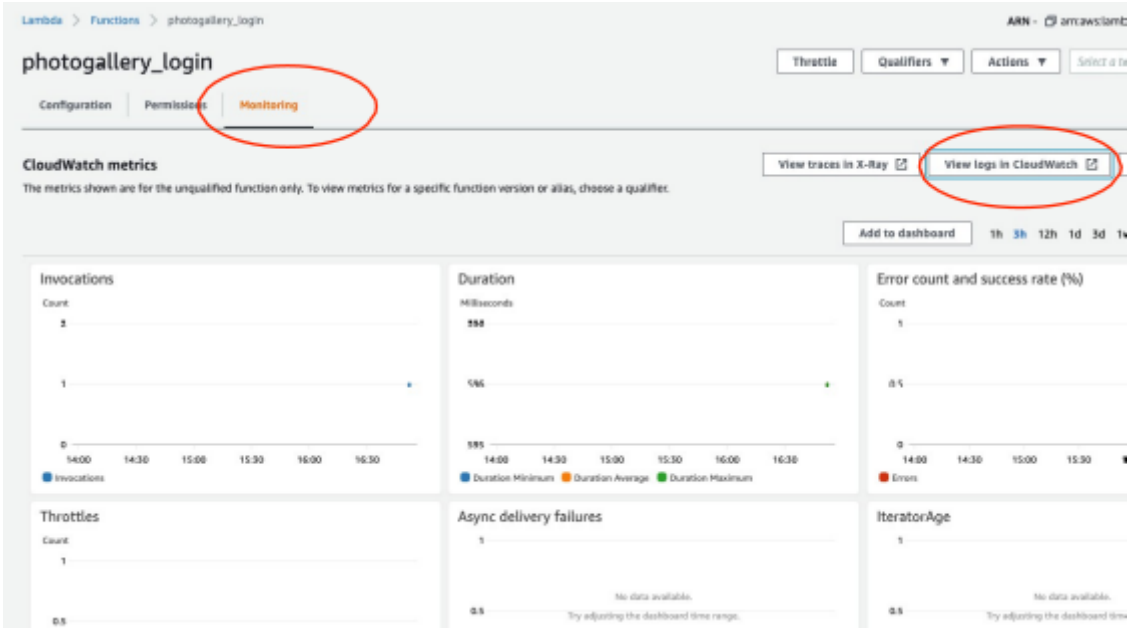
1. Test your code on localhost before uploading to cloud

```
[11:20] (/>ω<)/ ~ $ cd /Users/jolinchen/Desktop/Spring_2021/TA_ECE_4150/Lab1/staticwebsite
[11:20] (/>ω<)/ ~/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 CTRL-C to stop the server
```

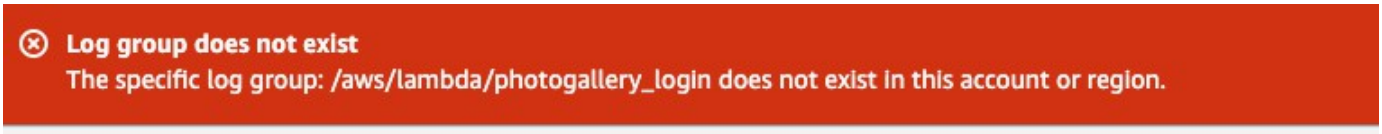
- Go to the root folder of the staticwebsite, type in command line http-server to set up HTTP server for testing. You can find the installation information of http-server at: https://www.npmjs.com/package/http-server. Then open the url as indicated in your terminal.

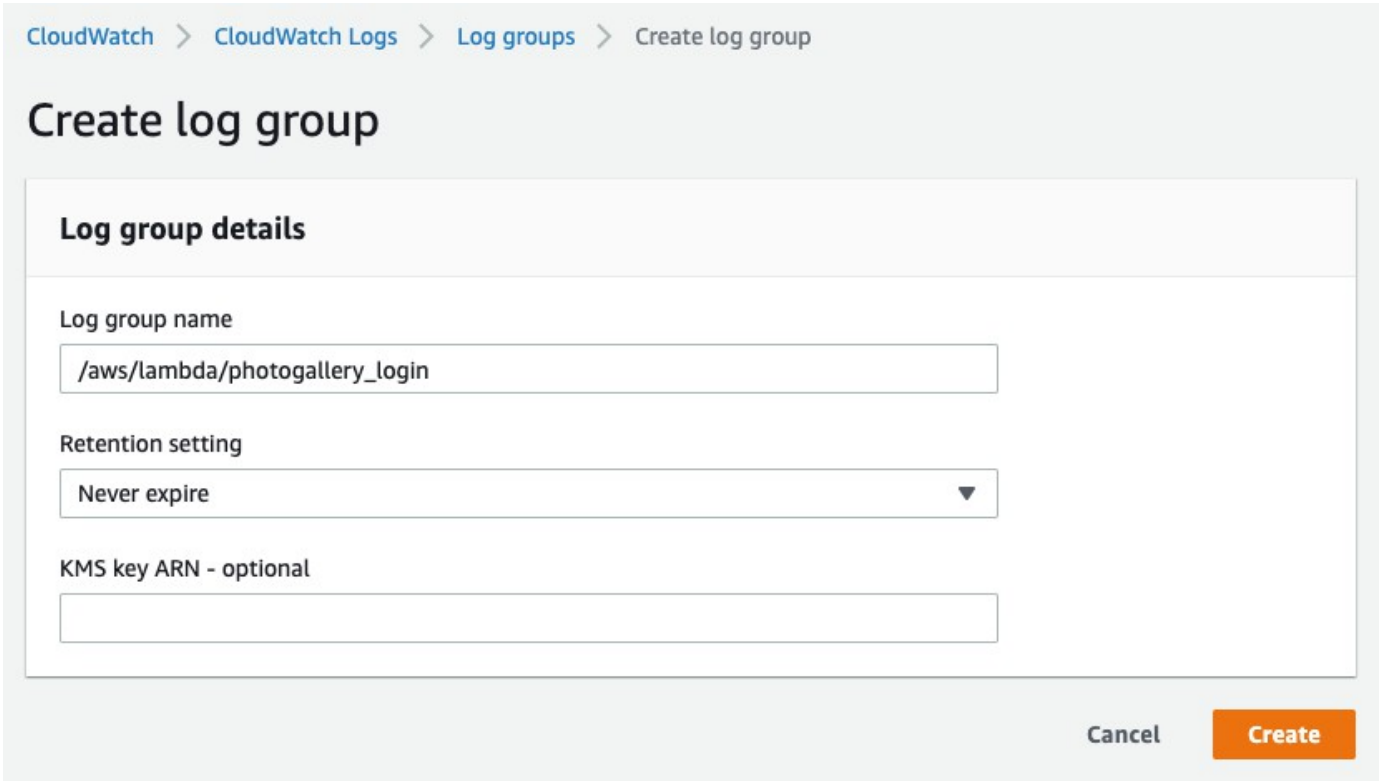
2. Use CloudWatch to monitor data flow

- 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”.

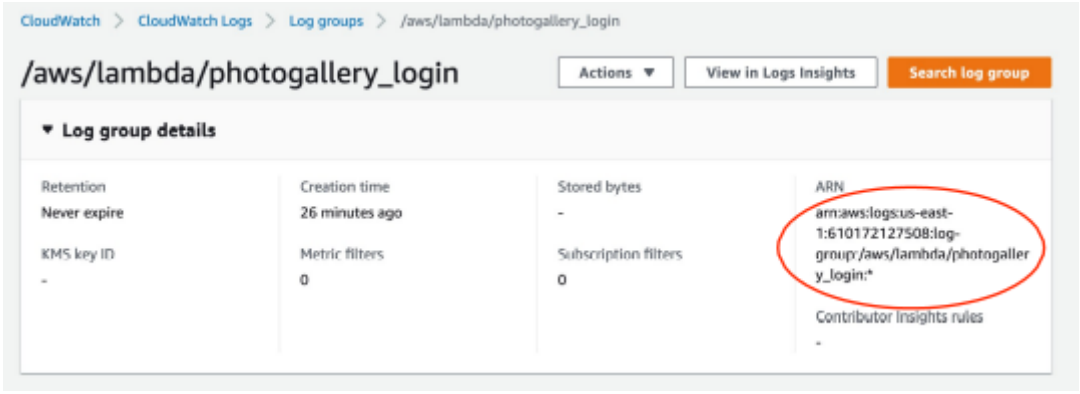


You have to create a log group in order to get the logs. To so do, in your newly opened CloudWatch Management Console page, copy the link above (in this case: group: /aws/lambda/ photogallery_login): Go back to Log groups, choose “Create log group”, in Log group name, paste the link you just copied then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

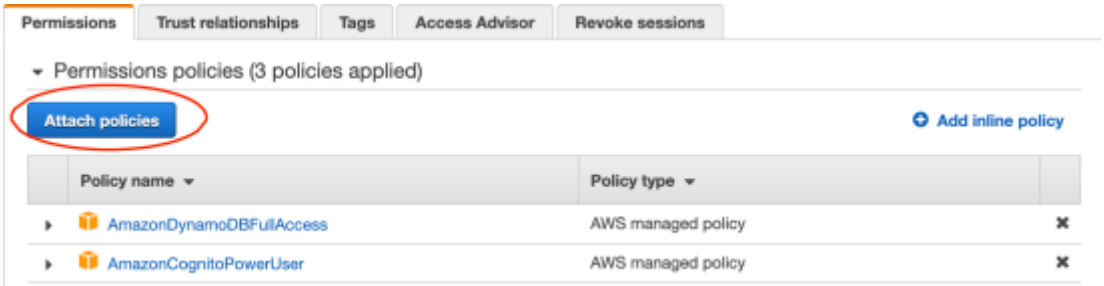




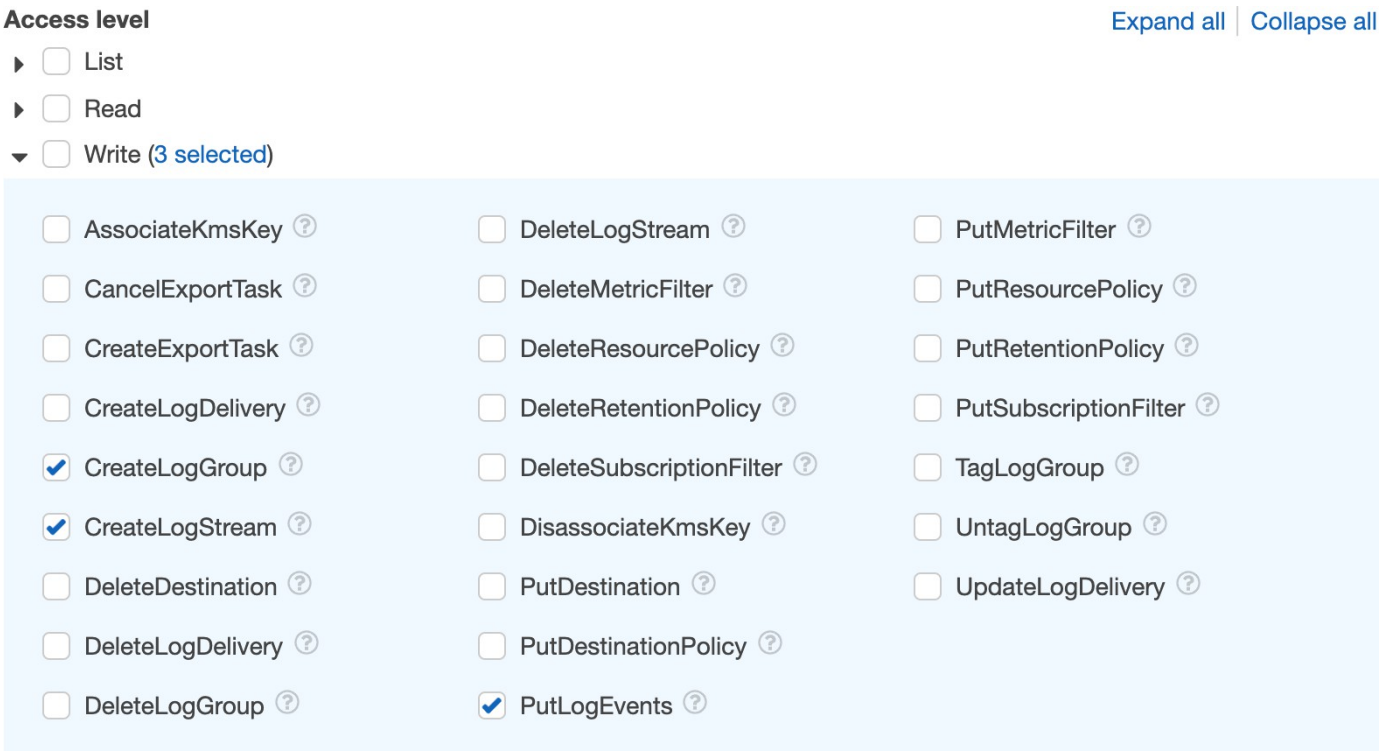
In your newly created log group, click on the dropdown button of “Log group details” and copy the ARN in it:



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”:



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



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.


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-1:610172127508:log-group:/aws/lambda/photogallery_login
                :*"
        }
    ]
}
```

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