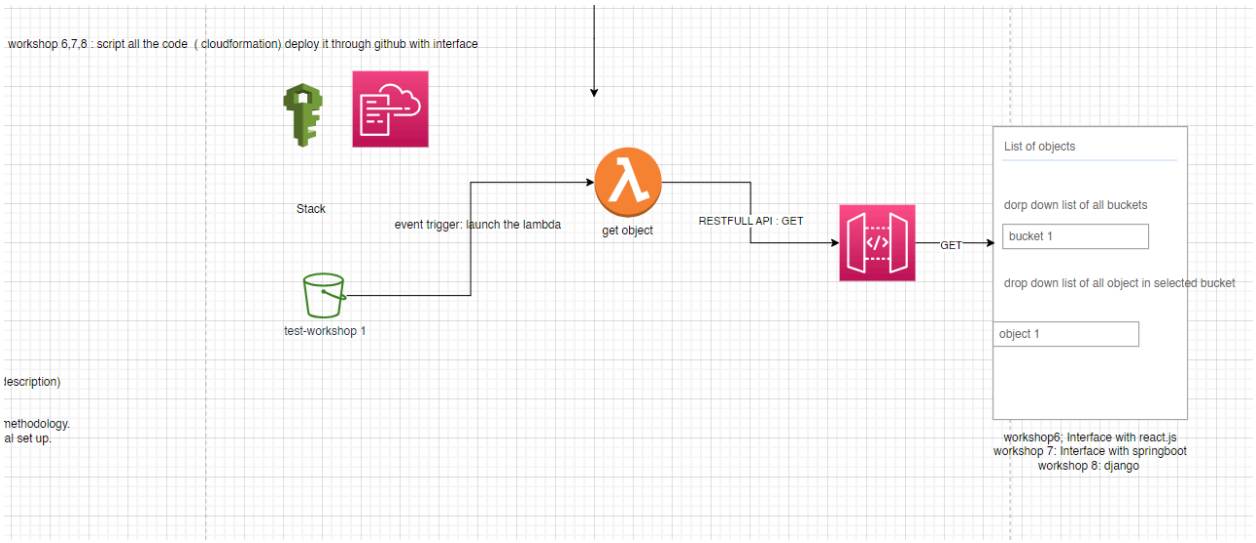
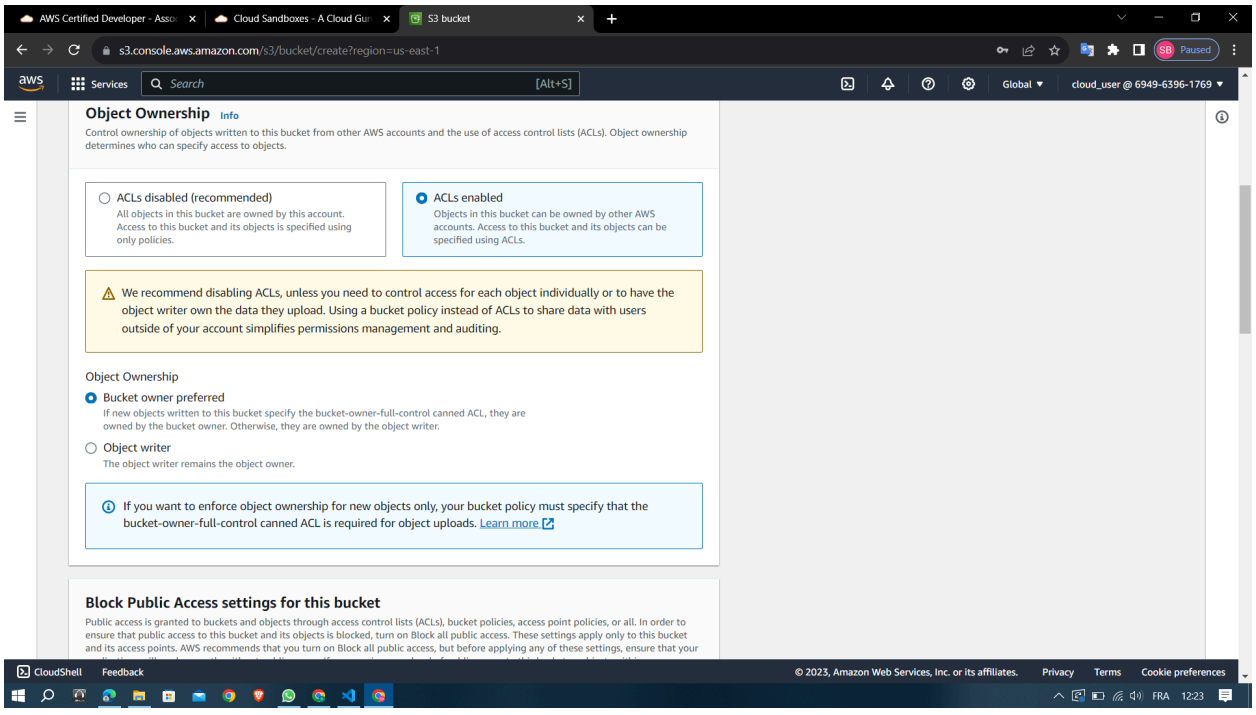
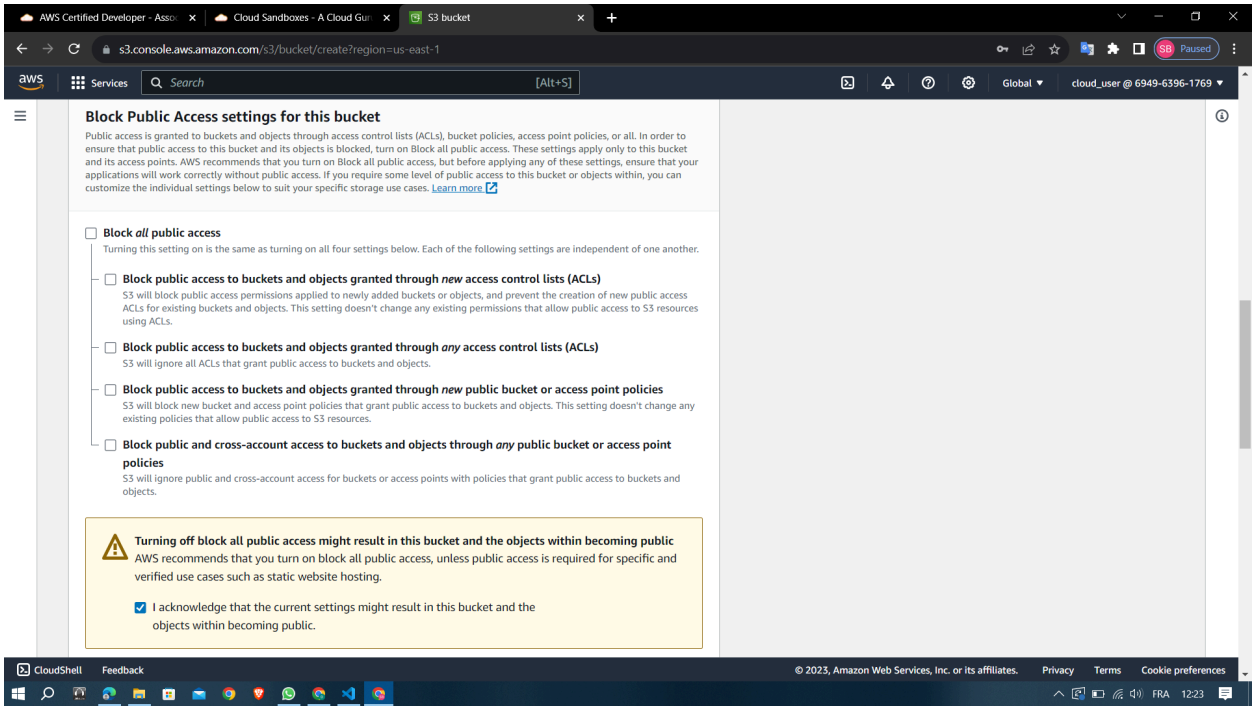


1. Create an S3 Bucket:

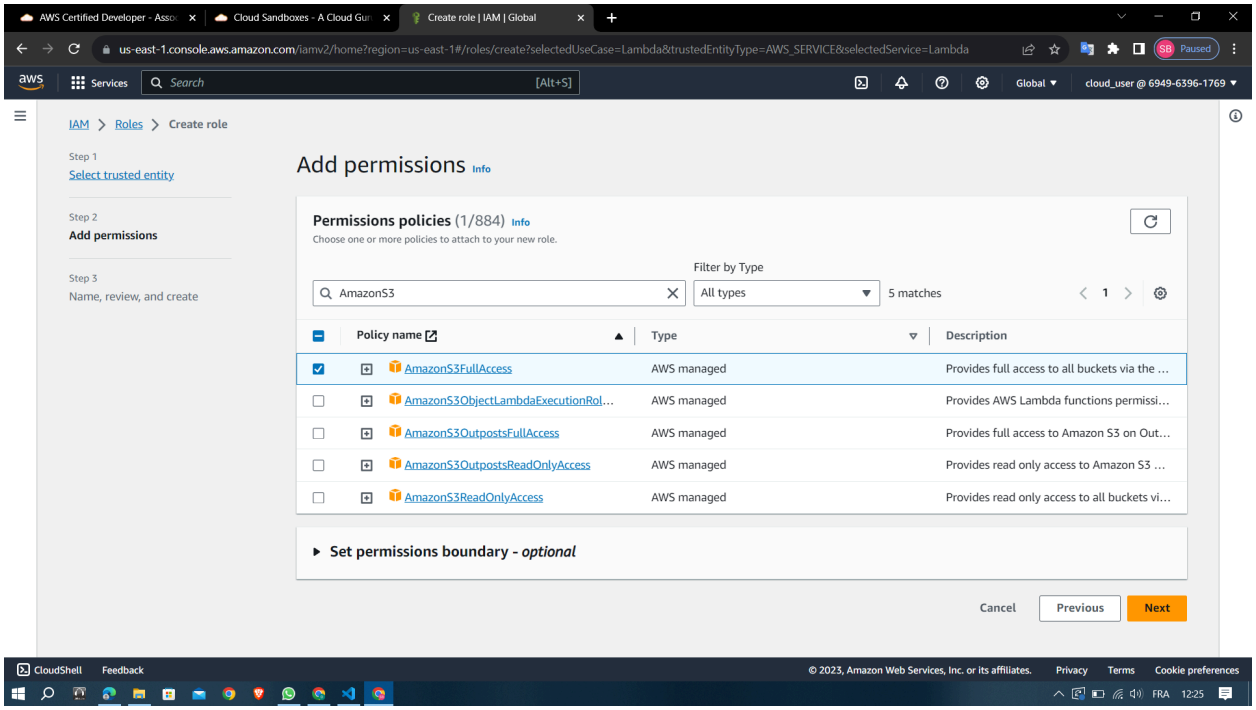
-Sign in to the AWS Management Console.
-Navigate to S3.
-Click “Create bucket”.
-Enter a unique bucket name and choose a region.
-Configure options and set permissions as required.
-Review and then click "Create bucket".





2. Create a Role and Attach Policy to the S3 Bucket:

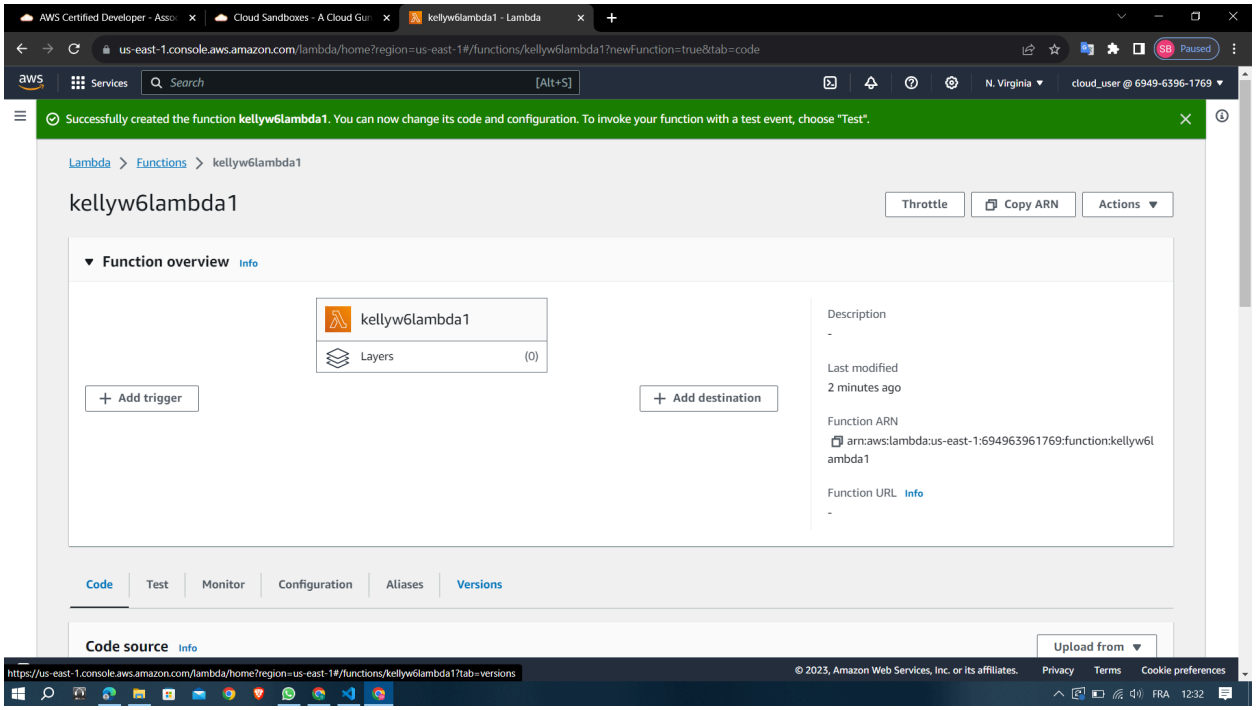
-Navigate to IAM (Identity and Access Management).
-Click on “Roles” and then “Create role”.
-Choose "Lambda" as the trusted entity (since the role will be assumed by Lambda).
-Click “Next” to attach permissions.
-Create a policy that gives access to the S3 bucket. Use the policy generator or write a policy manually. An example policy for read access:



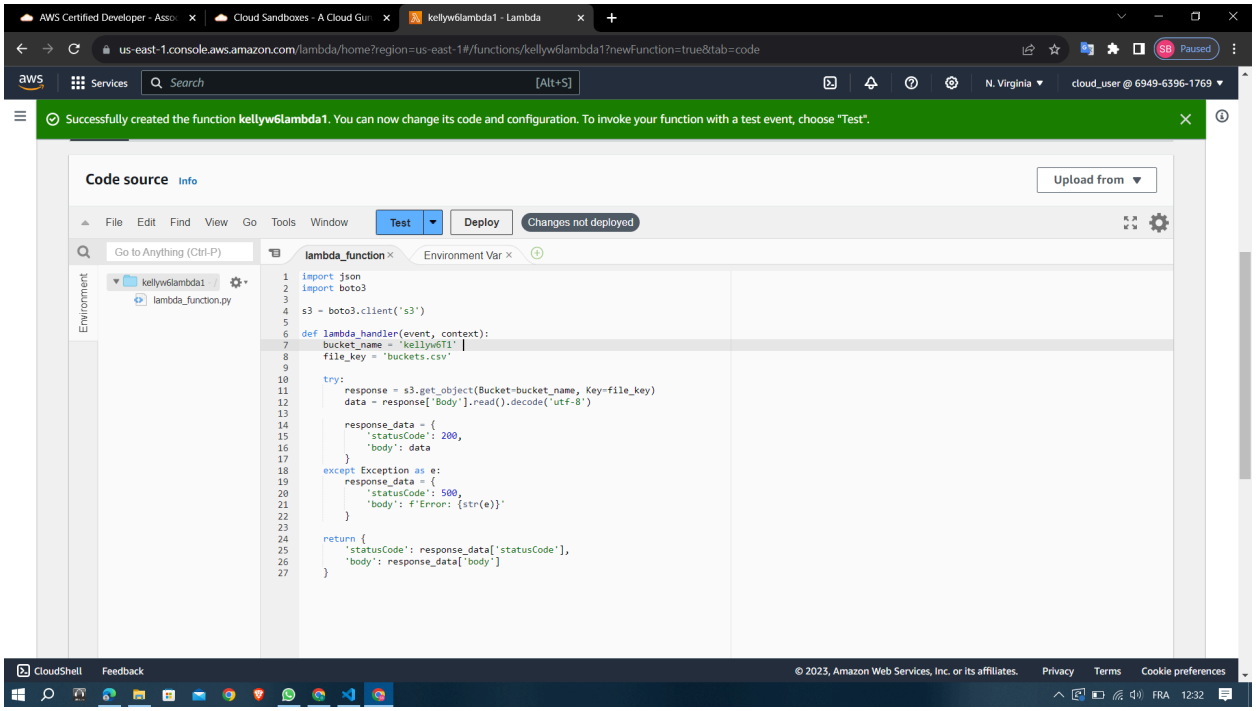

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Effect": "Allow",
                        "Action": [
                                "s3:GetObject",
                                "s3:PutObject",
                                "s3:ListBucket"
                        ],
                        "Resource": [
                                "arn:aws:s3:::your-bucket-name",
                                "arn:aws:s3:::your-bucket-name/*"
                        ]
                }
        ]
}
```

3. Create Lambda function to list objects of a given bucket:

-Navigate to Lambda in the AWS Console.
-Click “Create function”.
-Name your function, and choose the runtime (e.g., Python, Node.js).
-In the “Role” section, choose the role you created in the previous step.
-Once the function is created, write your code.



Lambda code:

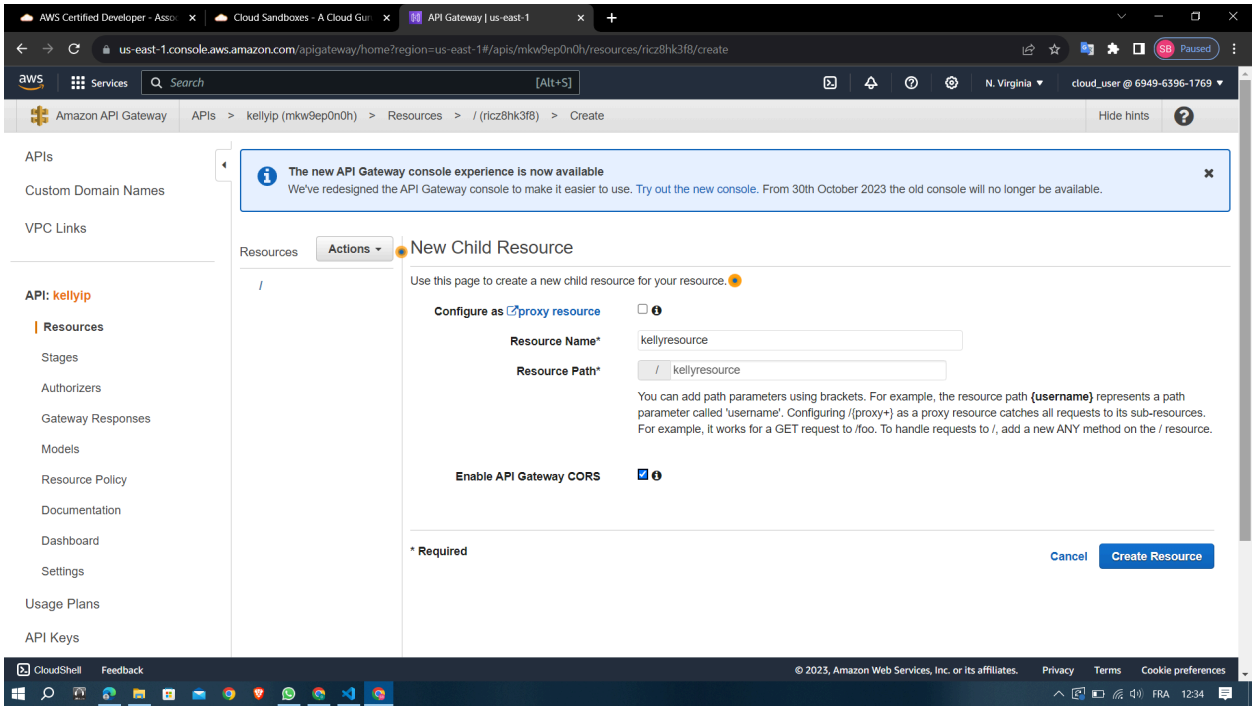


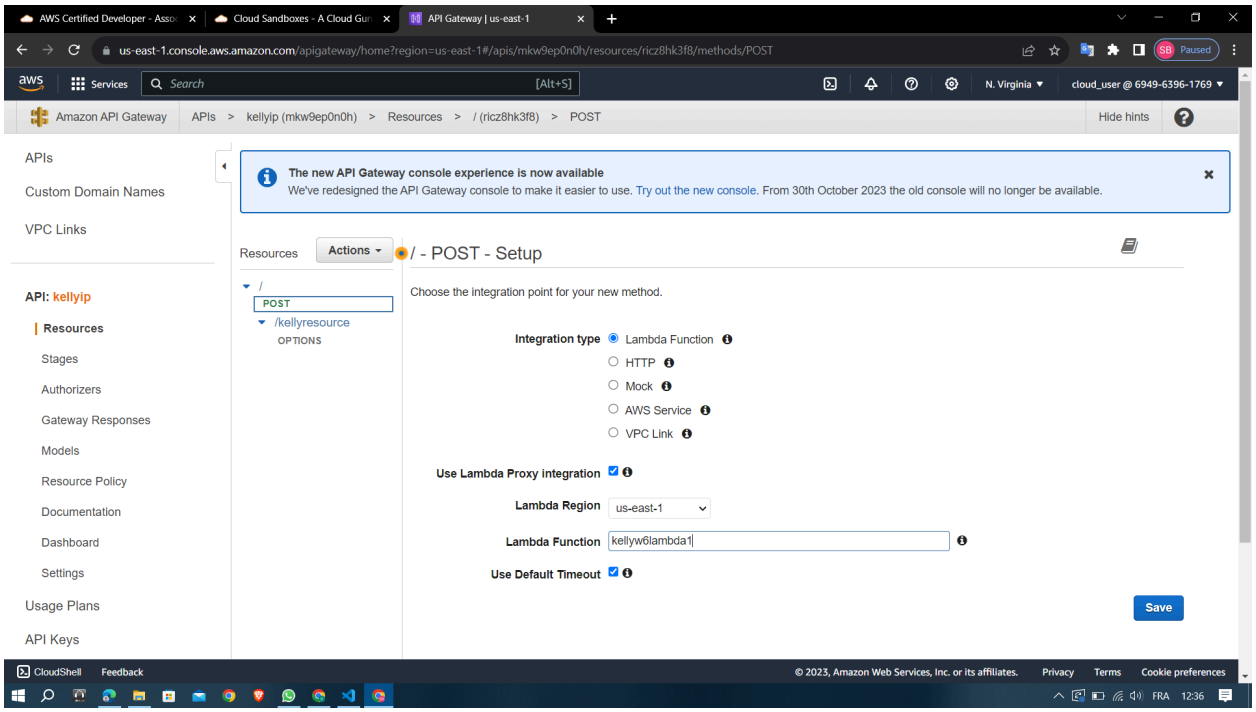

```
import json
import boto3

s3 = boto3.client('s3')
```

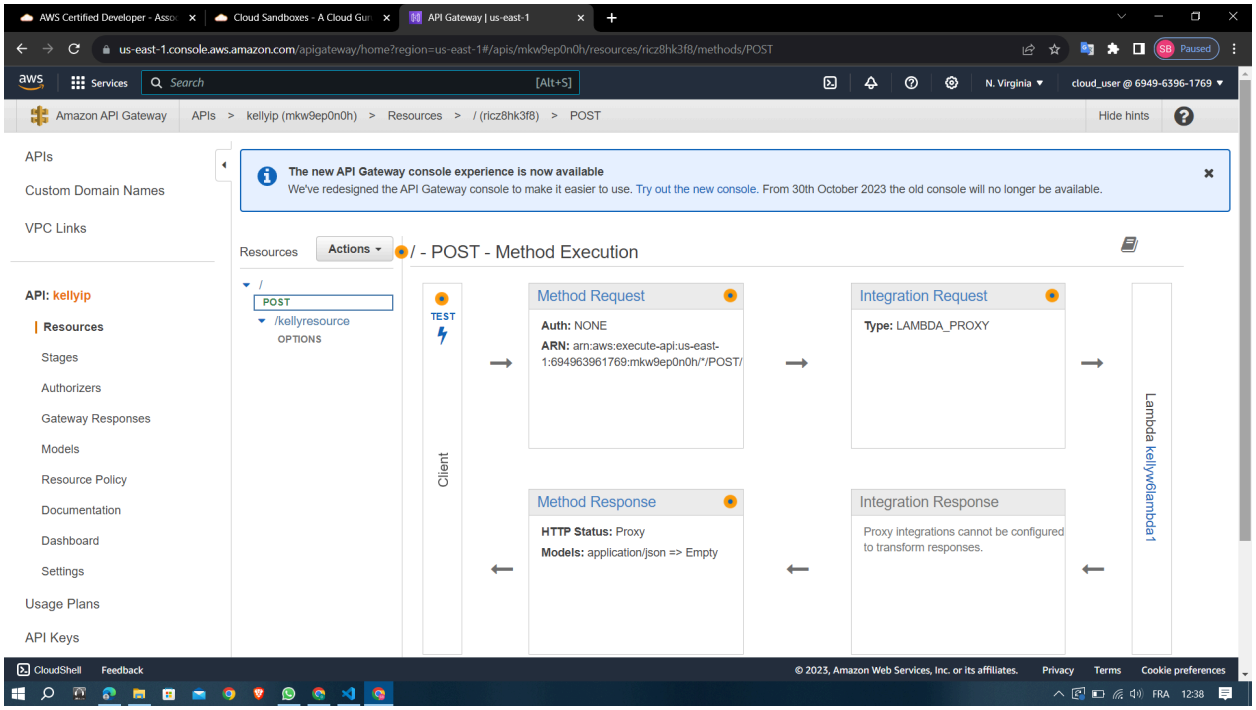
```
def lambda_handler(event, context):
    bucket_name = '"'
  //  file_key = 'buckets.csv'

    try:
        response = s3.get_object(Bucket=bucket_name, Key=file_key)
        data = response['Body'].read().decode('utf-8')

        response_data = {
            'statusCode': 200,
            'body': data
        }
    except Exception as e:
        response_data = {
            'statusCode': 500,
            'body': f'Error: {str(e)}'
        }

    return {
        'statusCode': response_data['statusCode'],
        'body': response_data['body']
    }
```

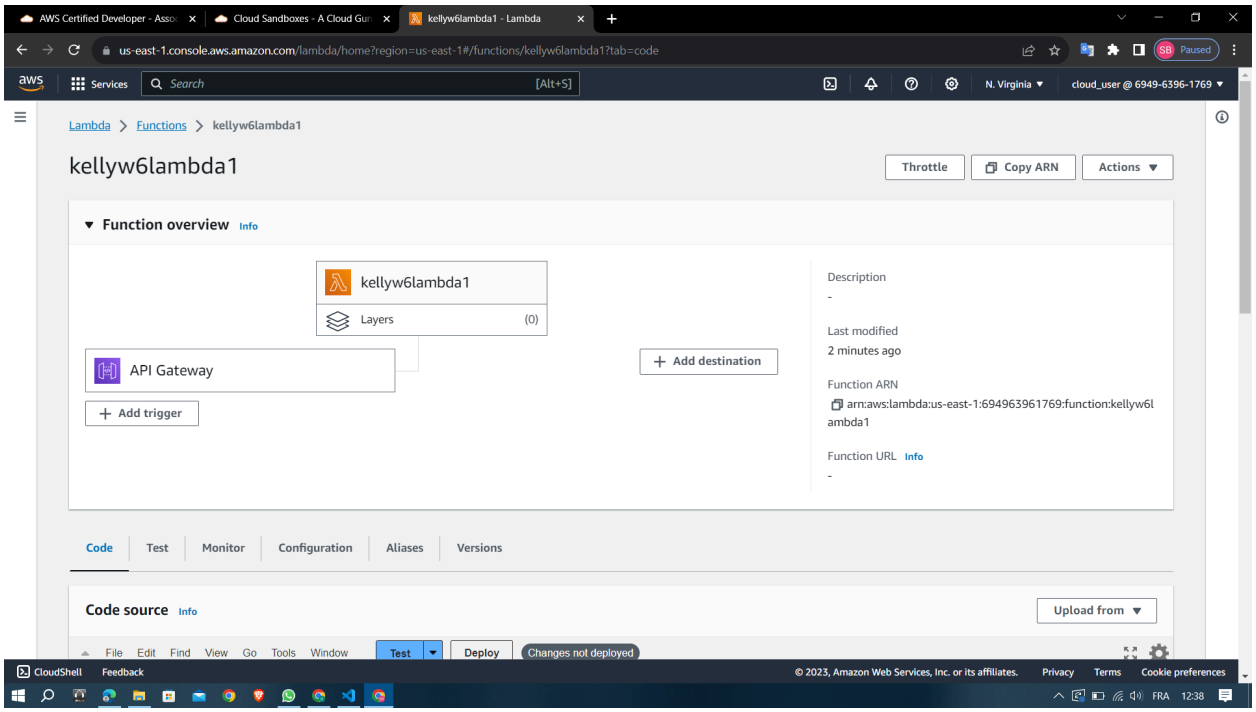
4. Setup API Gateway:

-Navigate to API Gateway in AWS Console and create a new REST API.
-Create a new resource (e.g., /listObjects).
-Create a POST method (to send the bucket name).
-Set the integration type to Lambda and select the function you created.
-Deploy your API.


AWS Certified Developer - Assoc
Cloud Sandboxes - A Cloud Gur
API Gateway | us-east-1
us-east-1.console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/mkw9ep0n0h/resources/ricz8hk3f8/create
Services
Search
[Alt+S]
N. Virginia
cloud_user @ 6949-6396-1769
Amazon API Gateway
APIs > kellyip (mkw9ep0n0h) > Resources > / (ricz8hk3f8) > Create
Hide hints
APIs
Custom Domain Names
VPC Links
API: kellyip
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Dashboard
Settings
Usage Plans
API Keys
The new API Gateway console experience is now available
We've redesigned the API Gateway console to make it easier to use. Try out the new console. From 30th October 2023 the old console will no longer be available.
Resources
Actions
New Child Resource
Use this page to create a new child resource for your resource.
Configure as proxy resource
Resource Name*
kellyresource
Resource Path*
/
kellyresource
You can add path parameters using brackets. For example, the resource path {username} represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.
Enable API Gateway CORS
* Required
Cancel
Create Resource
CloudShell
Feedback
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
FRA
12:34


AWS Certified Developer - Assoc
Cloud Sandboxes - A Cloud Gur
API Gateway | us-east-1
us-east-1.console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/mkw9ep0n0h/resources/ricz8hk3f8/methods/POST
Services
Search
[Alt+S]
N. Virginia
cloud_user @ 6949-6396-1769
Amazon API Gateway
APIs > kellyip (mkw9ep0n0h) > Resources > / (ricz8hk3f8) > POST
Hide hints
APIs
Custom Domain Names
VPC Links
API: kellyip
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Dashboard
Settings
Usage Plans
API Keys
The new API Gateway console experience is now available
We've redesigned the API Gateway console to make it easier to use. Try out the new console. From 30th October 2023 the old console will no longer be available.
Resources
Actions
/ - POST - Setup
/
POST
/kellyresource
OPTIONS
Choose the integration point for your new method.
Integration type
Lambda Function
HTTP
Mock
AWS Service
VPC Link
Use Lambda Proxy integration
Lambda Region
us-east-1
Lambda Function
kellyw6lambda1
Use Default Timeout
Save
CloudShell
Feedback
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
FRA
12:36





NB: Enable CORS:
In API Gateway, ensure that you've enabled CORS for your resource to let your React app call the API.

5. Frontend with React:

-Initialize a new project with Create React App:

```
npx create-react-app s3-object-browser
cd s3-object-browser
```

-Install Axios or another HTTP client:

```
npm install axios
```

-Create a component to interact with your API:

```
import React, { useState, useEffect } from 'react';

function App() {
  const [buckets, setBuckets] = useState([]);
  const [selectedBucket, setSelectedBucket] = useState('');
  const [objects, setObjects] = useState([]);

  useEffect(() => {

    fetch('arn:aws:lambda:us-east-1:694963961769:function:kellyw6lambda1')
      .then(response => response.json())
      .then(data => {
        setBuckets(data.buckets);
      })
      .catch(error => {
        console.error("Error fetching the bucket list:", error);
      });
  }, []);

  useEffect(() => {
    if (selectedBucket) {
      // For simplicity, assuming a similar endpoint for fetching objects from a selected bucket

fetch(`arn:aws:lambda:us-east-1:694963961769:function:kellyw6lambda1?bucket=${selectedBu
cket}`)
        .then(response => response.json())
        .then(data => {
          setObjects(data.objects);
        })
        .catch(error => {
          console.error("Error fetching the objects list:", error);
```

```
      });
    }
  }, [selectedBucket]);

  return (
    <div>
      <select onChange={(e) => setSelectedBucket(e.target.value)}>
        <option value="">Select a bucket</option>
        {buckets.map(bucket => (
          <option key={bucket} value={bucket}>{bucket}</option>
        ))}
      </select>

      {selectedBucket && (
        <select>
          <option value="">Select an object</option>
          {objects.map(obj => (
            <option key={obj} value={obj}>{obj}</option>
          ))}
        </select>
      )}
    </div>
  );
}

export default App;
```

-Include this component in App.js and render it.

```
import React from 'react';
import './App.css';
import S3ObjectBrowser from './S3ObjectBrowser';

function App() {
    return (
        <div className="App">
            <h1>S3 Object Browser</h1>
            <S3ObjectBrowser />
        </div>
    );
}
```

```
export default App;
```

-Start the React app:

```
npm start
```