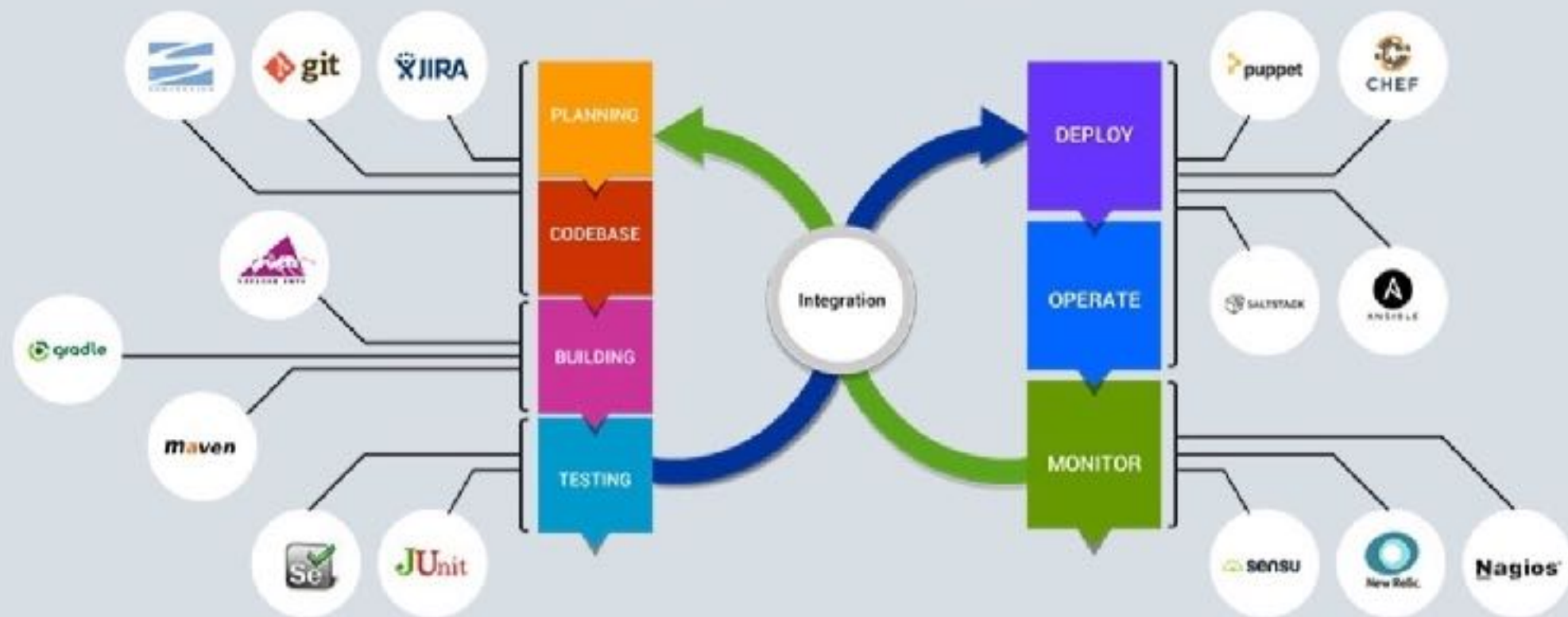


AWS

KESHAV KUMMARI

Agile | Linux | AWS | DevOps | Python

Keshav Kummari



WHAT IS AWS LAMBDA?

- AWS Lambda is a compute service that lets you run code without provisioning or managing servers.
- AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second.
- You pay only for the compute time you consume - there is no charge when your code is not running.
- With AWS Lambda, you can run code for virtually any type of application or backend service - all with zero administration.
- AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging.
- All you need to do is supply your code in one of the languages that AWS Lambda supports (currently Node.js, Java, C#, Go and Python).

- You can use AWS Lambda to run your code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table; to run your code in response to HTTP requests using Amazon API Gateway; or invoke your code using API calls made using AWS SDKs.
- With these capabilities, you can use Lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB process streaming data stored in Kinesis, or create your own back end that operates at AWS scale, performance, and security.
- You can also build serverless applications composed of functions that are triggered by events and automatically deploy them using AWS CodePipeline and AWS CodeBuild. For more information, see [Deploying Lambda-based Applications](#).

WHEN SHOULD I USE AWS LAMBDA?

- AWS Lambda is an ideal compute platform for many application scenarios, provided that you can write your application code in languages supported by AWS Lambda (that is, Node.js, Java, Go and C# and Python), and run within the AWS Lambda standard runtime environment and resources provided by Lambda.
- When using AWS Lambda, you are responsible only for your code. AWS Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources.
- This is in exchange for flexibility, which means you cannot log in to compute instances, or customize the operating system or language runtime.
- These constraints enable AWS Lambda to perform operational and administrative activities on your behalf, including provisioning capacity, monitoring fleet health, applying security patches, deploying your code, and monitoring and logging your Lambda functions.

- If you need to manage your own compute resources, Amazon Web Services also offers other compute services to meet your needs.
- Amazon Elastic Compute Cloud (Amazon EC2) service offers flexibility and a wide range of EC2 instance types to choose from.
- It gives you the option to customize operating systems, network and security settings, and the entire software stack, but you are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.
- Elastic Beanstalk offers an easy-to-use service for deploying and scaling applications onto Amazon EC2 in which you retain ownership and full control over the underlying EC2 instances.

- Lambda Benefits

Benefits

NO SERVERS TO MANAGE

AWS Lambda automatically runs your code without requiring you to provision or manage servers. Just write the code and upload it to Lambda.

CONTINUOUS SCALING

AWS Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

SUBSECOND METERING

With AWS Lambda, you are charged for every 100ms your code executes and the number of times your code is triggered. You don't pay anything when your code isn't running.

AWS Lambda

HOW IT WORKS

- AWS Lambda



Flow

USE CASES

- What can you build with AWS Lambda?

Data processing

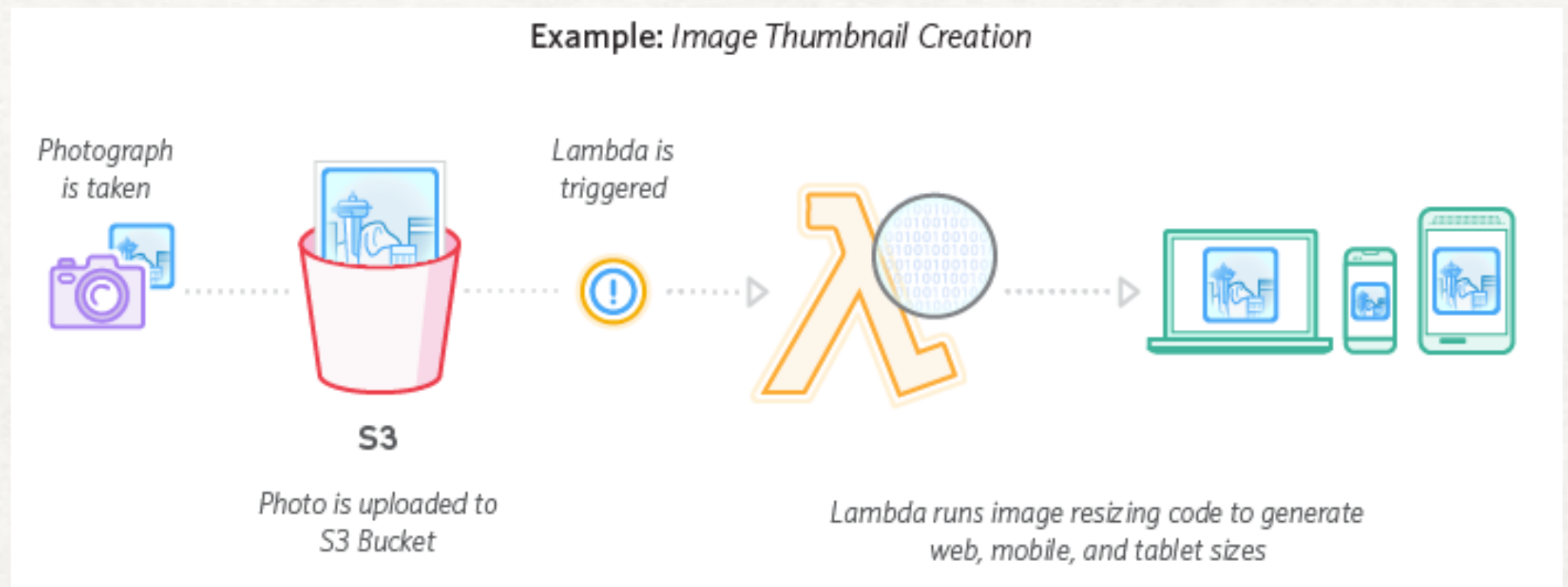
- You can use AWS Lambda to execute code in response to triggers such as changes in data, shifts in system state, or actions by users. Lambda can be directly triggered by AWS services such as S3, DynamoDB, Kinesis, SNS, and CloudWatch, or it can be orchestrated into workflows by AWS Step Functions.
- This allows you to build a variety of real-time serverless data processing systems.

REAL-TIME FILE PROCESSING

YOU CAN USE AMAZON S3 TO TRIGGER AWS LAMBDA TO PROCESS DATA IMMEDIATELY AFTER AN UPLOAD.

FOR EXAMPLE, YOU CAN USE LAMBDA TO THUMBNAIL IMAGES, TRANSCODE VIDEOS, INDEX FILES, PROCESS LOGS, VALIDATE CONTENT, AND AGGREGATE AND FILTER DATA IN REAL-TIME.

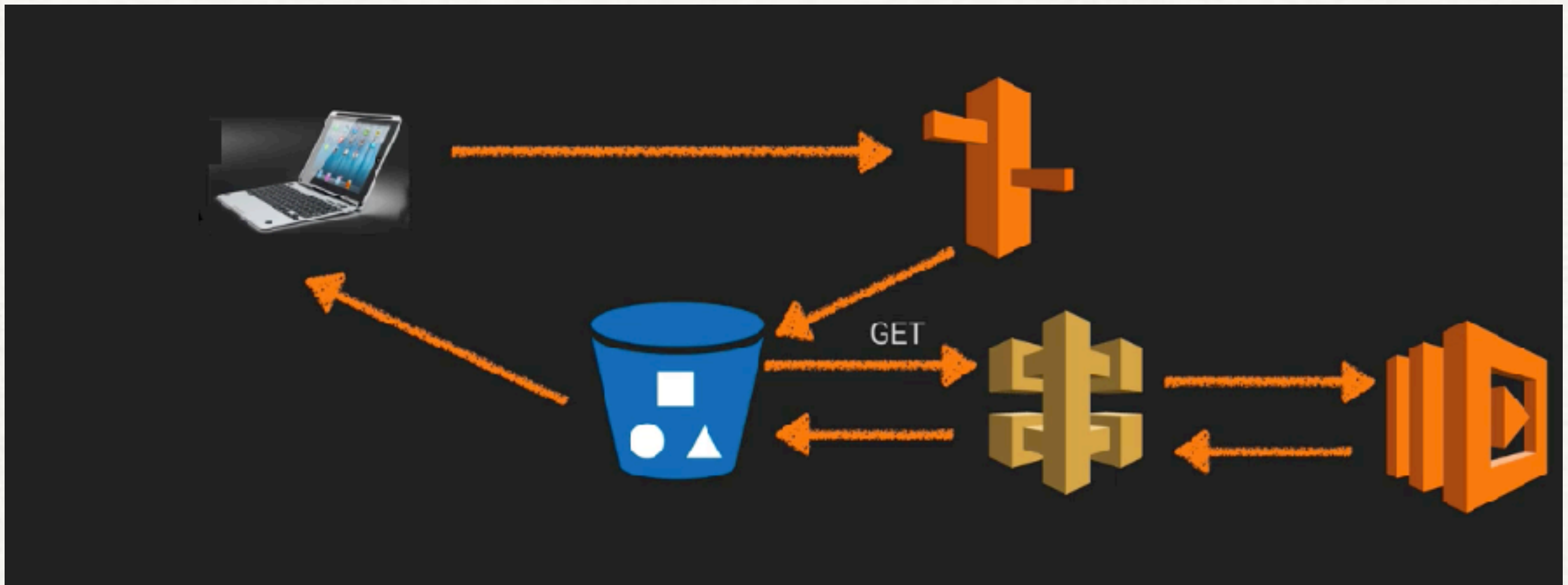
- Example



Example of AWS Lambda

LAMBDA FUNCTION EXAMPLE

- Sample Lambda Flow Diagram



User, Route 53, S3 Bucket, API Gateway & Lambda Function

STEP - 1 : GO TO AWS MGMT CONSOLE

- Click on Lambda

The screenshot shows the AWS Management Console interface. The top navigation bar includes 'Services' and 'Resource Groups'. A search bar is present with the placeholder text 'Find a service by name or feature (for example, EC2, S3 or VM, sto)'. Below the search bar, the 'Compute' category is expanded, showing a list of services: EC2, Lightsail, Elastic Container Service, Lambda (highlighted in orange), Batch, and Elastic Beanstalk. To the right, the 'Developer Tools' category is also visible, listing CodeStar, CodeCommit, CodeBuild, CodeDeploy, CodePipeline, Cloud9, and X-Ray. A secondary window or tab is overlaid on the right side, titled 'AWS Lambda'. It shows the 'Dashboard' and 'Functions' sections. The 'Resources for Asia Pacific (Mumbai)' section displays the following metrics:

Resources for Asia Pacific (Mumbai)	
Lambda function(s)	0
Code storage	0 bytes (0% of 75.0 GB)
Full account concurrency	1000
Unreserved account concurrency	1000

At the bottom of this section is an orange button labeled 'Create function'.

Under Services - Compute - Lambda

Now, click on "Create Function"

STEP-2 : SELECT AUTHOR FROM SCRATCH

Author from scratch [Info](#)

Name

MyTestFunction

Runtime

Python 3.6

Role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create new role from template(s)

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name

Enter a name for your new role.

MyLambdaRole

Policy templates

Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

Simple Microservice permissions X

Cancel

Create function

Click on Create Function

STEP-3 : CROSS CHECK REQUIRED DETAILS

- Scroll Down & Check required details

The screenshot displays the AWS Lambda console interface for a function named "MyTestFunction". The top navigation bar includes the AWS logo, "Services", "Resource Groups", and user information "AWS_DevOps_KK" with a location of "Mumbai". The function name "MyTestFunction" is prominently displayed at the top left of the configuration area. To its right are buttons for "Throttle", "Qualifiers", "Actions", a dropdown for "Select a test event..", and "Test" and "Save" buttons. Below the function name are two tabs: "Configuration" (which is selected) and "Monitoring". The main content area is titled "Designer" and features a left-hand menu under "Add triggers" with options: "API Gateway", "CloudWatch Events", "CloudWatch Logs", "CodeCommit", and "Cognito Sync Trigger". The central workspace shows a key icon and a dashed box with the text "Add triggers from the list on the left". To the right of the workspace, a box labeled "MyTestFunction" is connected to a vertical stack of resources: "Amazon CloudWatch Logs" and "Amazon DynamoDB". A dashed box at the bottom right indicates where resources the function's role has access to will be shown.

Scroll Down & Add Description under Basic Settings

STEP-4 : SCROLL DOWN & ADD DESCRIPTION

- Add Description

The screenshot shows the AWS Lambda console for a function named 'MyTestFunction'. The URL in the browser is <https://ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/MyTestFunction?tab=graph>. The page has a top navigation bar with the AWS logo, 'Services', 'Resource Groups', and user information 'AWS_DevOps_KK' in 'Mumbai'. Below the navigation bar, the function name 'MyTestFunction' is displayed, followed by buttons for 'Throttle', 'Qualifiers', 'Actions', 'Select a test event..', 'Test', and 'Save'. A section for adding tags is present, with fields for 'Key' and 'Value' and a 'Remove' button. The main content area is divided into two panels: 'Execution role' on the left and 'Basic settings' on the right. The 'Execution role' panel shows a dropdown for 'Choose an existing role' and a section for 'Existing role' with a dropdown showing 'service-role/MyLambdaRole'. The 'Basic settings' panel contains the 'Description' field, which is highlighted with a blue border and contains the text 'My Basic Lambda Website'. Below the description are 'Memory (MB)' and 'Timeout' settings. The memory is set to 128 MB, and the timeout is set to 0 minutes and 3 seconds.

Secure | <https://ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/MyTestFunction?tab=graph>

aws Services Resource Groups

MyTestFunction Throttle Qualifiers Actions Select a test event.. Test Save

You can use tags to group and filter your functions. A tag consists of a case-sensitive key-value pair. [Learn more.](#)

Key Value Remove

Execution role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Choose an existing role

Existing role

You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

service-role/MyLambdaRole

Basic settings

Description

My Basic Lambda Website

Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout [Info](#)

0 min 3 sec

Scroll Above & go to IDE & Add Python Function as per next slide

STEP- 5 : GO TO IDE & ADD PYTHON FUNCTION

NOW, ADD PYTHON FUNCTION

The screenshot shows the AWS Lambda console for a function named 'MyTestFunction'. The 'Function code' tab is selected, and the code editor displays a Python lambda handler function. The function code is as follows:

```
1 def lambda_handler(event, context):
2     print('In Lambda handler')
3
4     resp = {
5         'statusCode': 200,
6         'headers': {
7             'Access-Control-Allow-Origin': '*',
8         },
9         'body': 'Keshav Kumari'
10    }
11
12    return resp
```

The console also shows the 'Code entry type' set to 'Edit code inline', the 'Runtime' set to 'Python 3.6', and the 'Handler' set to 'lambda_function.lambda_handler'. The 'Save' button is highlighted in orange.

Save the function

STEP- 6: ADD TRIGGERS

NOW, WE NEED TO ADD TRIGGERS, AS PART OF TRIGGERS THERE ARE MANY TRIGGERS. DOUBLE CLICK ON "API GATEWAY".

The screenshot shows the AWS Lambda console interface. At the top, the navigation bar includes the AWS logo, 'Services', 'Resource Groups', a search icon, a notification bell, 'AWS_DevOps_KK', 'Mumbai', and 'Support'. Below the navigation bar, the breadcrumb trail reads 'Lambda > Functions > MyTestFunction'. The function's ARN is displayed as 'arn:aws:lambda:ap-south-1:631258978605:function:MyTestFunction'.

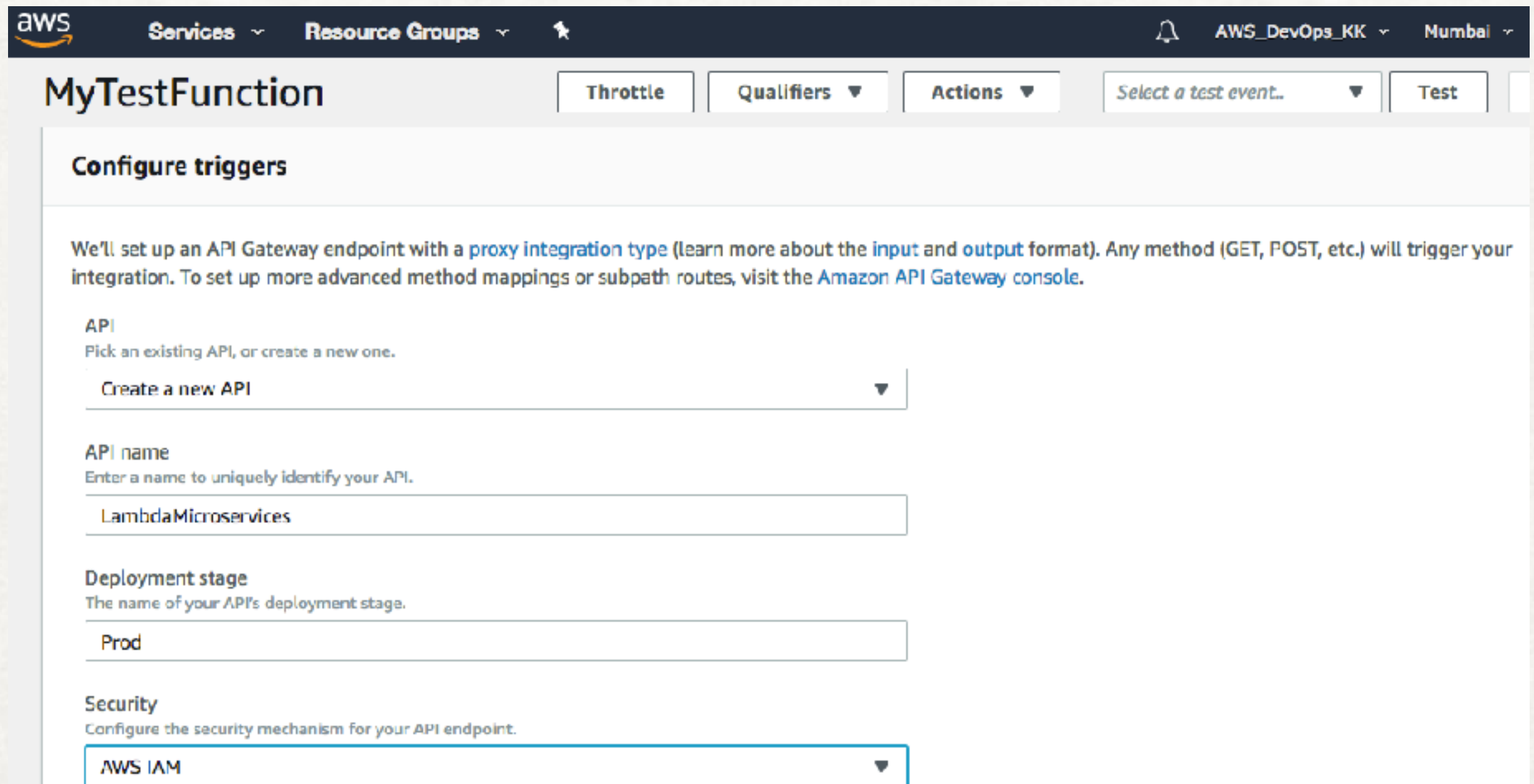
The main section is titled 'MyTestFunction'. It contains several buttons: 'Throttle', 'Qualifiers' (with a dropdown arrow), 'Actions' (with a dropdown arrow), 'Select a test event..' (with a dropdown arrow), 'Test', and 'Save'. Below these buttons is a green success message: 'Congratulations! Your Lambda function "MyTestFunction" has been successfully created. You can now change its code and configuration. Click on the "Test" button to input a test event when you are ready to test your function.' with a close button (X).

Below the message are two tabs: 'Configuration' (selected) and 'Monitoring'. Under the 'Configuration' tab, there is a 'Designer' section. On the left, under 'Add triggers', it says 'Click on a trigger from the list below to add it to your function.' The list includes 'API Gateway' (highlighted), 'CloudWatch Events', and 'CloudWatch Logs'. In the center, a diagram shows the function 'MyTestFunction' (labeled 'Saved') connected to three triggers: 'API Gateway' (labeled 'Configuration required'), 'Amazon CloudWatch Logs', and 'Amazon DynamoDB'.

Scroll Down & Fill the required fields

STEP- 7 : CONFIGURE THE TRIGGERS

ADD THE REQUIRED FIELDS ACCORDINGLY



The screenshot shows the AWS API Gateway console interface. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', and a user profile icon. On the right, it shows a notification bell, the account name 'AWS_DevOps_KK', and the region 'Mumbai'. Below the navigation bar, the main header area displays 'MyTestFunction' followed by tabs for 'Throttle', 'Qualifiers', 'Actions', a dropdown for 'Select a test event..', and a 'Test' button. The 'Configure triggers' section is active, showing instructions: 'We'll set up an API Gateway endpoint with a proxy integration type (learn more about the input and output format). Any method (GET, POST, etc.) will trigger your integration. To set up more advanced method mappings or subpath routes, visit the Amazon API Gateway console.' Below this, there are four configuration fields: 1. 'API' with a dropdown menu currently showing 'Create a new API'. 2. 'API name' with a text input field containing 'LambdaMicroservices'. 3. 'Deployment stage' with a text input field containing 'Prod'. 4. 'Security' with a dropdown menu currently showing 'AWS IAM'.

aws Services Resource Groups

MyTestFunction Throttle Qualifiers Actions Select a test event.. Test

Configure triggers

We'll set up an API Gateway endpoint with a [proxy integration type](#) (learn more about the [input](#) and [output](#) format). Any method (GET, POST, etc.) will trigger your integration. To set up more advanced method mappings or subpath routes, visit the [Amazon API Gateway console](#).

API
Pick an existing API, or create a new one.

Create a new API

API name
Enter a name to uniquely identify your API.

LambdaMicroservices

Deployment stage
The name of your API's deployment stage.

Prod

Security
Configure the security mechanism for your API endpoint.

AWS IAM

Click on Add & Save the changes

STEP - 8 : CROSS CHECK THE API GATEWAY

API Gateway

LambdaMicroservices

Enabled

Delete

arn:aws:execute-api:ap-south-1:631258978605:erqywjzexg/*/*/MyTestFunction

▼ Details

Security: **AWS_IAM**

Method: **ANY**

Resource path: **/MyTestFunction**

API name: **LambdaMicroservices**

Identifier: **api-gateway/erqywjzexg/*/*/MyTestFunction**

Authorization: **AWS_IAM**

Invoke URL: **https://erqywjzexg.execute-api.ap-south-1.amazonaws.com/Prod/MyTestFunction**

Stage: **Prod**

Services ^ Resource Groups v

API

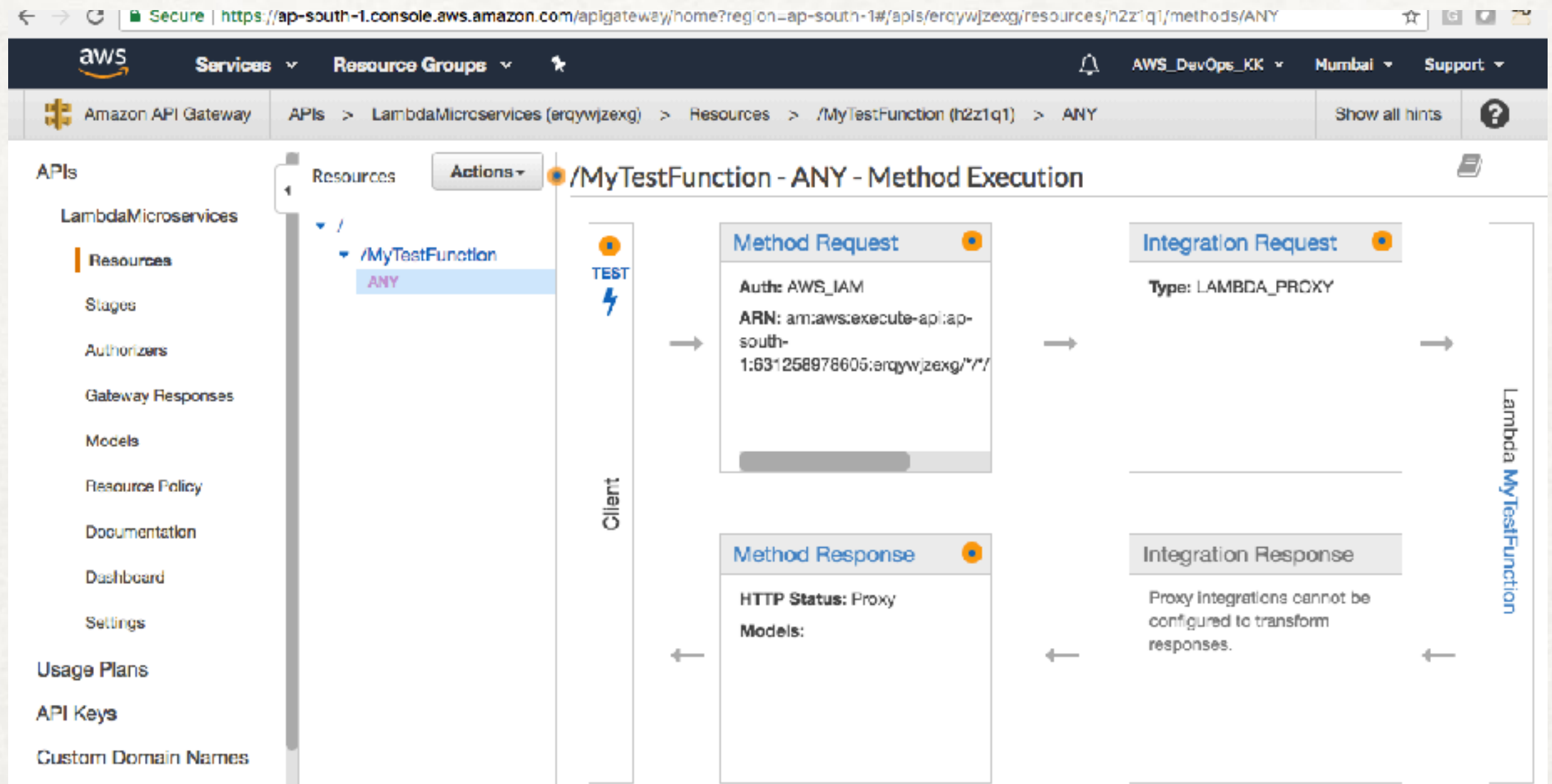
API Gateway
Build, Deploy and Manage APIs

Alexa for Business
Alexa for Business Provides Tools to Manage Alexa in Your Organization

Go to AWS Mgmt Console & Look for API Gateway

STEP - 9 : GO TO API GATEWAY

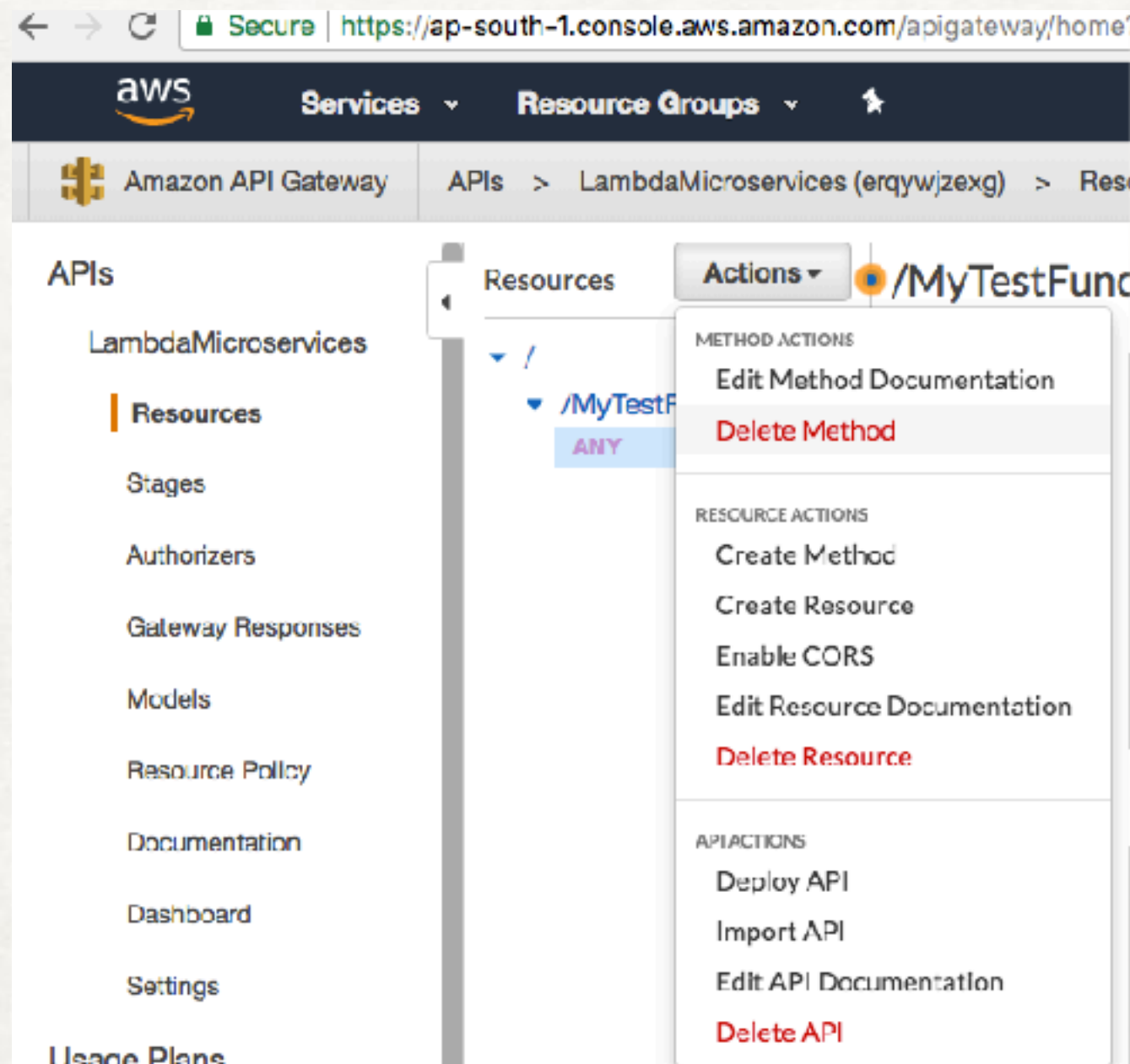
CLICK ON RESOURCES & CLICK ON "ANY" METHOD



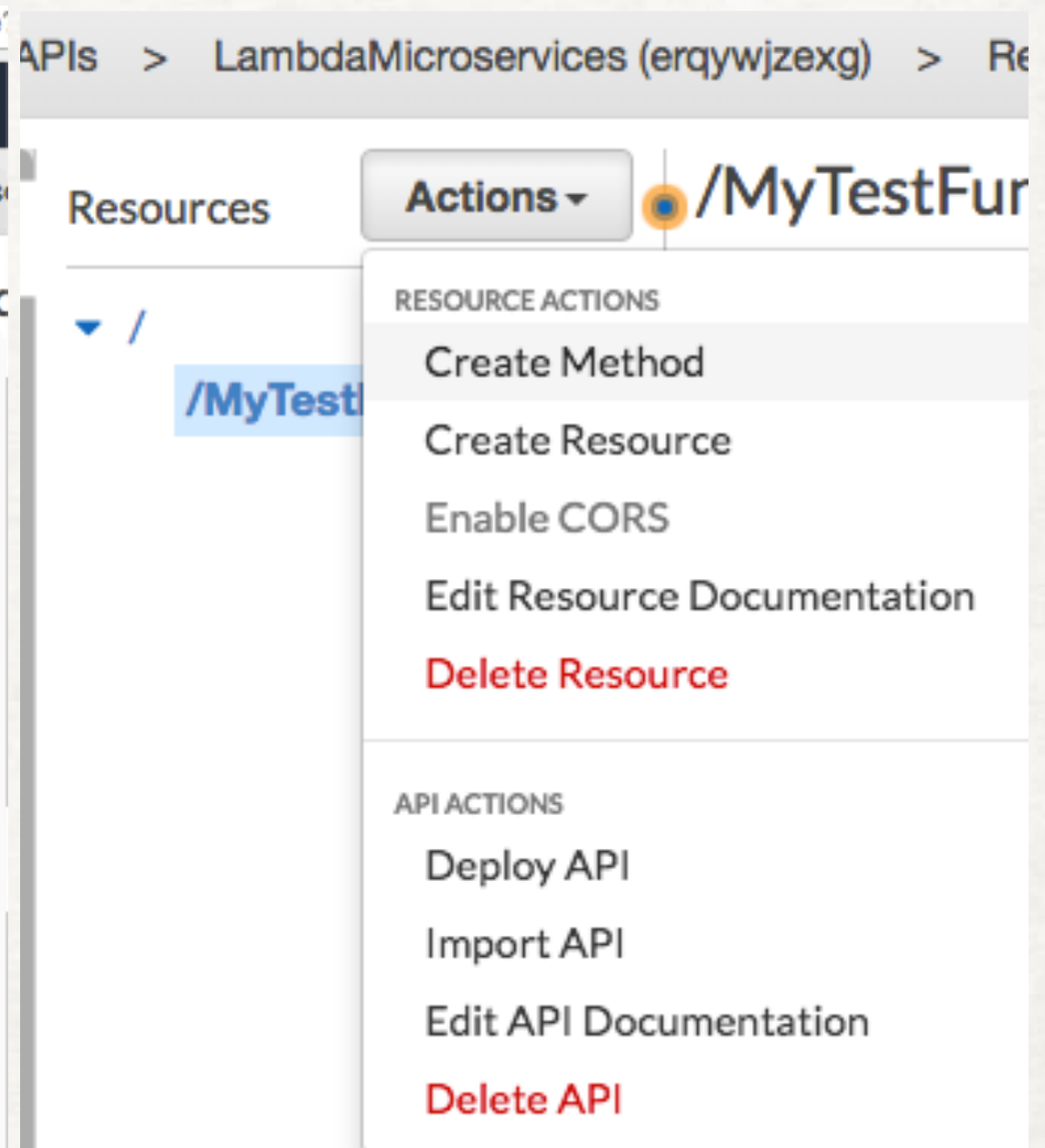
Cross check the details

STEP - 10 : DELETE "ANY" & ADD "GET" METHOD

- Go to Actions & Delete & Add methods



Delete ANY method



Click on Create Method & Select "Get"

STEP - 11 : GET METHOD DETAILS

GET METHOD

APIs > LambdaMicroservices (erqywjzexg) > Resources > /MyTestFunction (h2z1q1) > GET Show all hints ?

Resources **Actions** /MyTestFunction - GET - Setup

Resources

- /
- /MyTestFunction
 - GET

Choose the integration point for your new method.

Integration type ☒ Lambda Function i

☐ HTTP i

☐ Mock i

☐ AWS Service i

☐ VPC Link i

Use Lambda Proxy integration ☒ i

Lambda Region ap-south-1

Lambda Function

MyTestFunction i

Use Default Timeout ☒ i

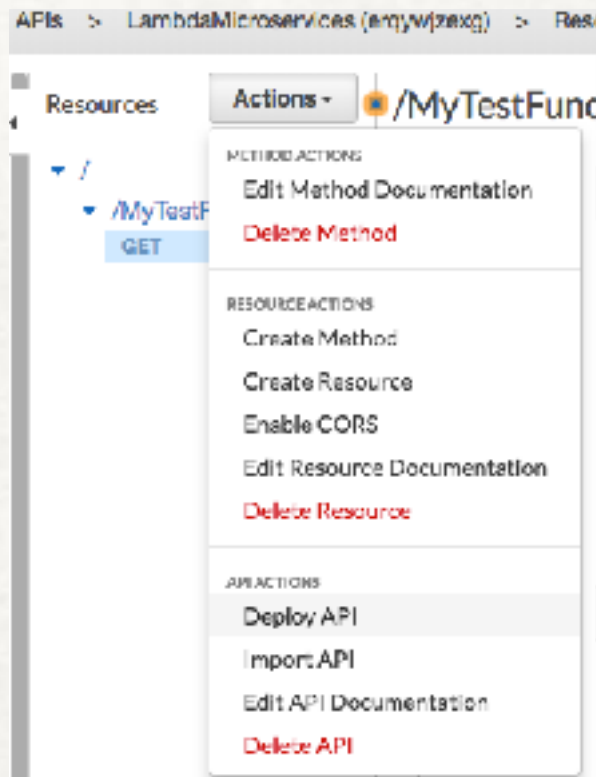
Save

Click on SAVE

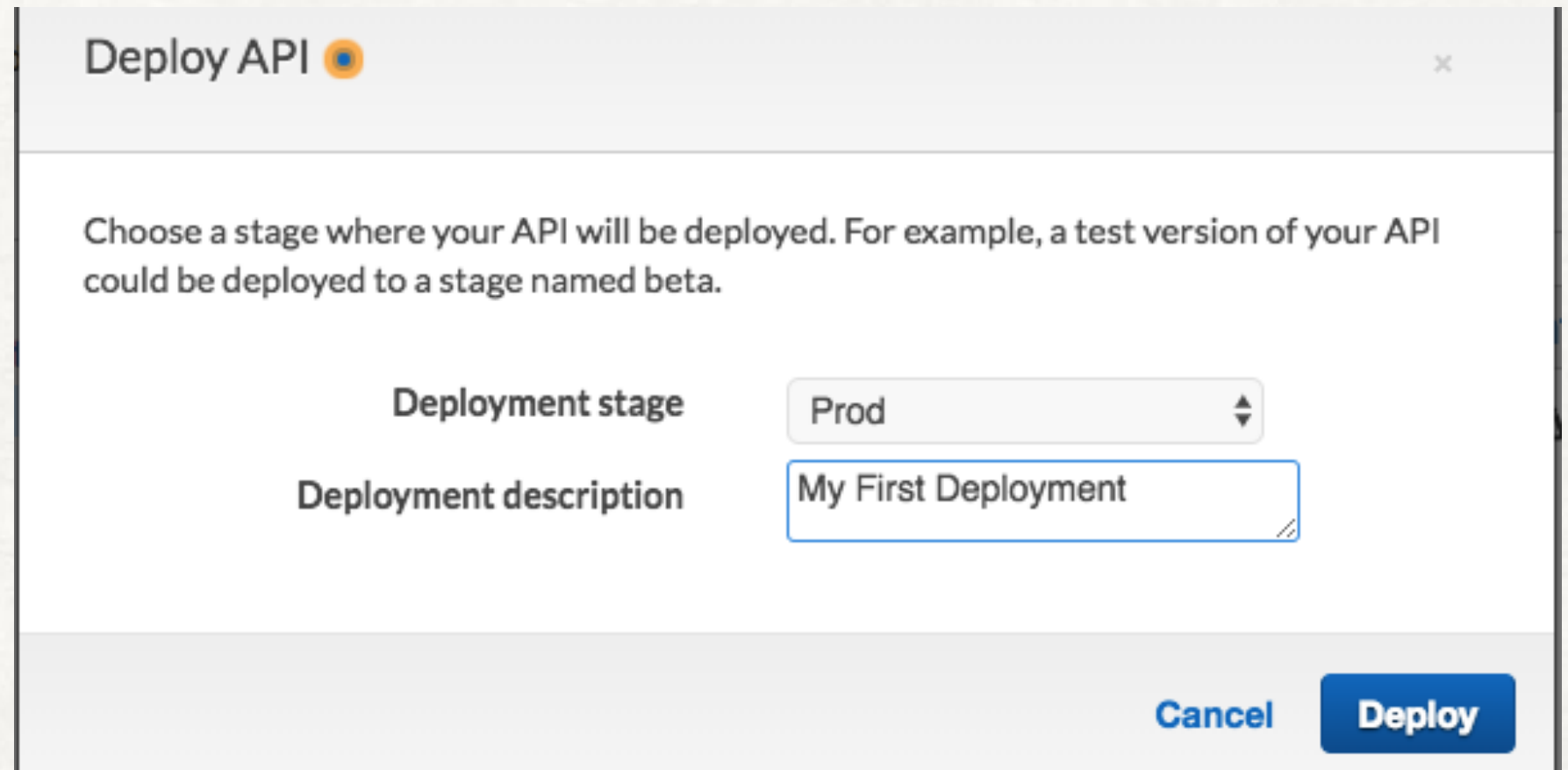
STEP - 12 : DEPLOY

GET METHOD

- Go to Actions & Deploy API



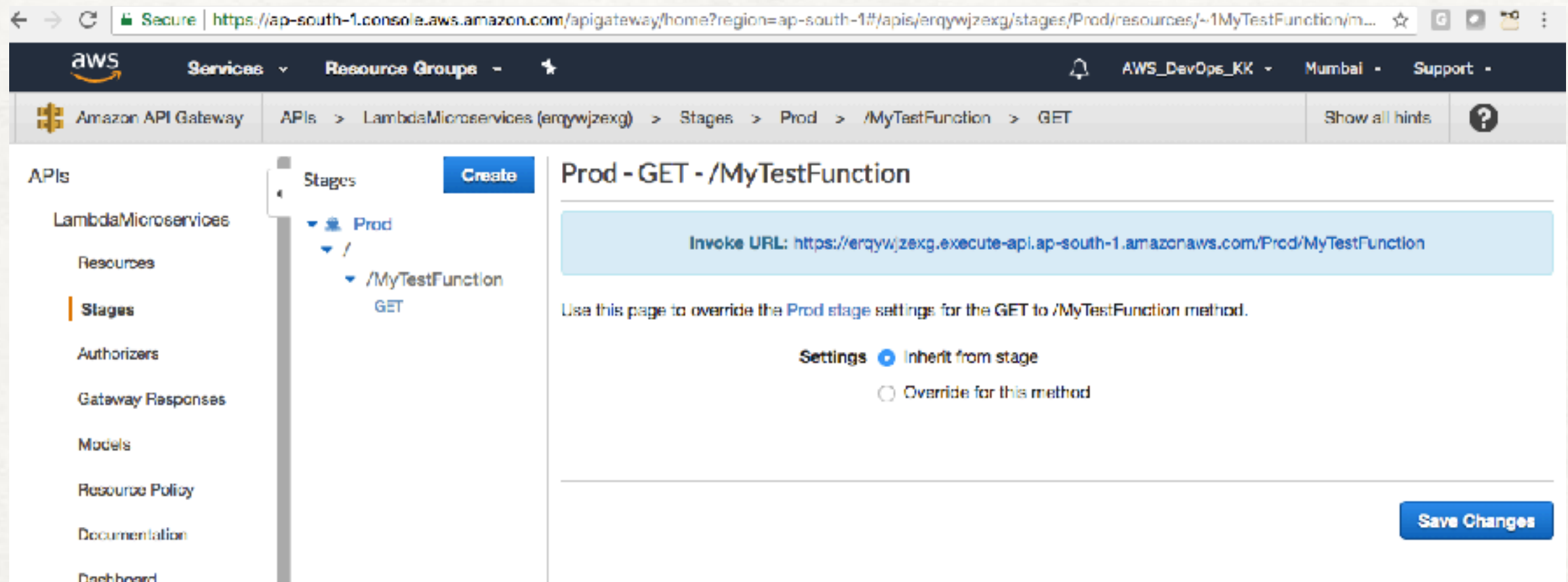
Click on Deploy API



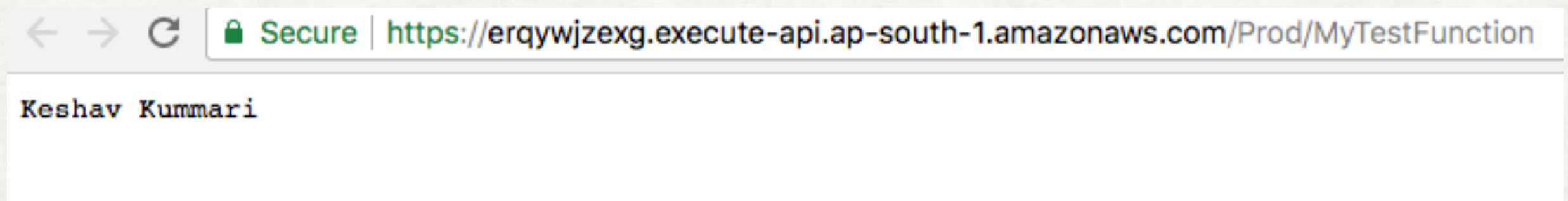
Click on Deploy

STEP-13 : GO TO STAGES & CLICK ON THE URL

- Click on Stages & Click on GET method



Copy the URL & go to the browser & Check

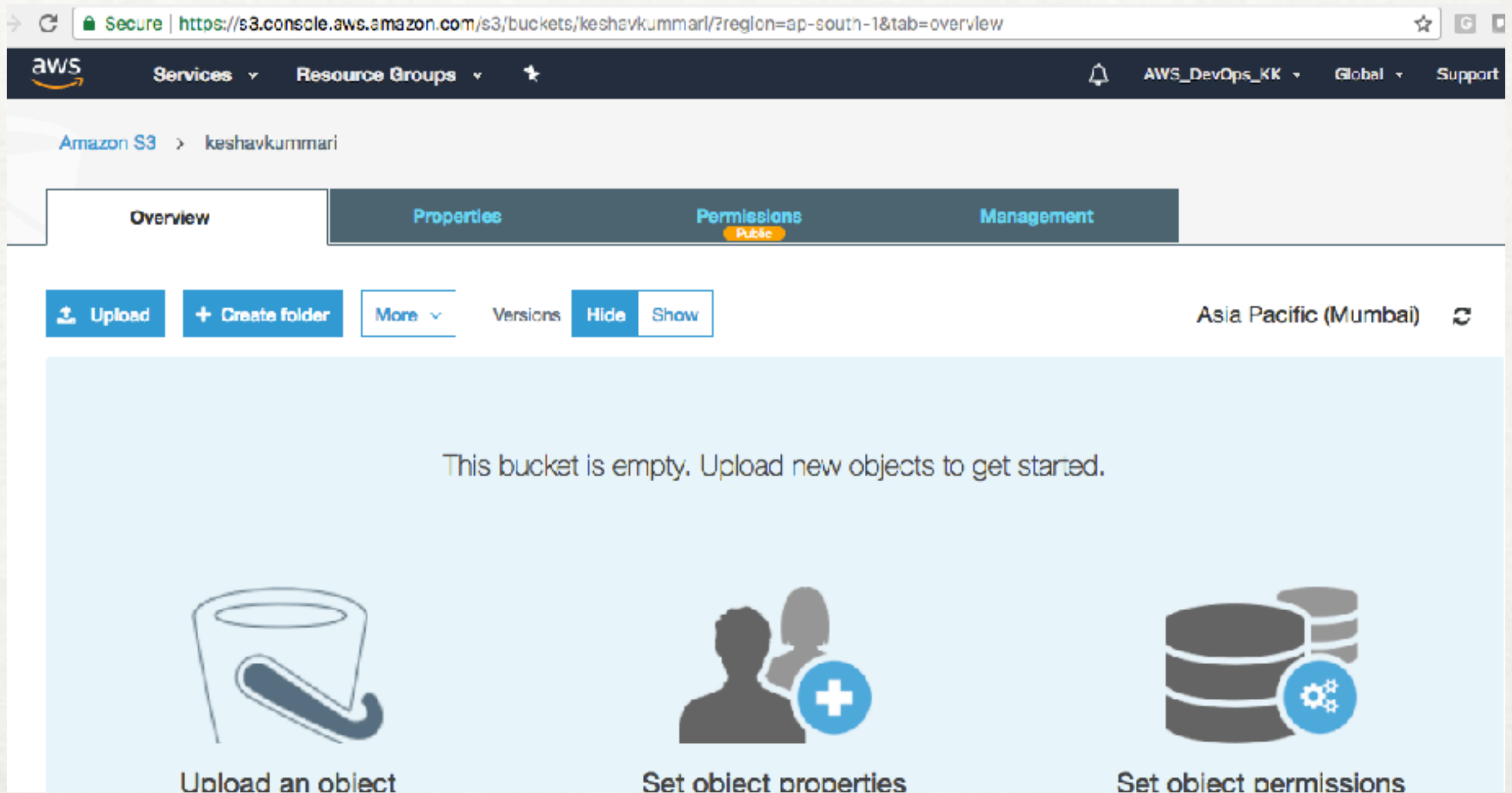


Lambda has been launched a server less website

STEP - 14 : GO TO S3 BUCKET

CREATE A BUCKET I.E. KESHAVKUMMARI

- Upload Files i.e. index.html & error.html



STEP - 15 : UPLOAD FILES

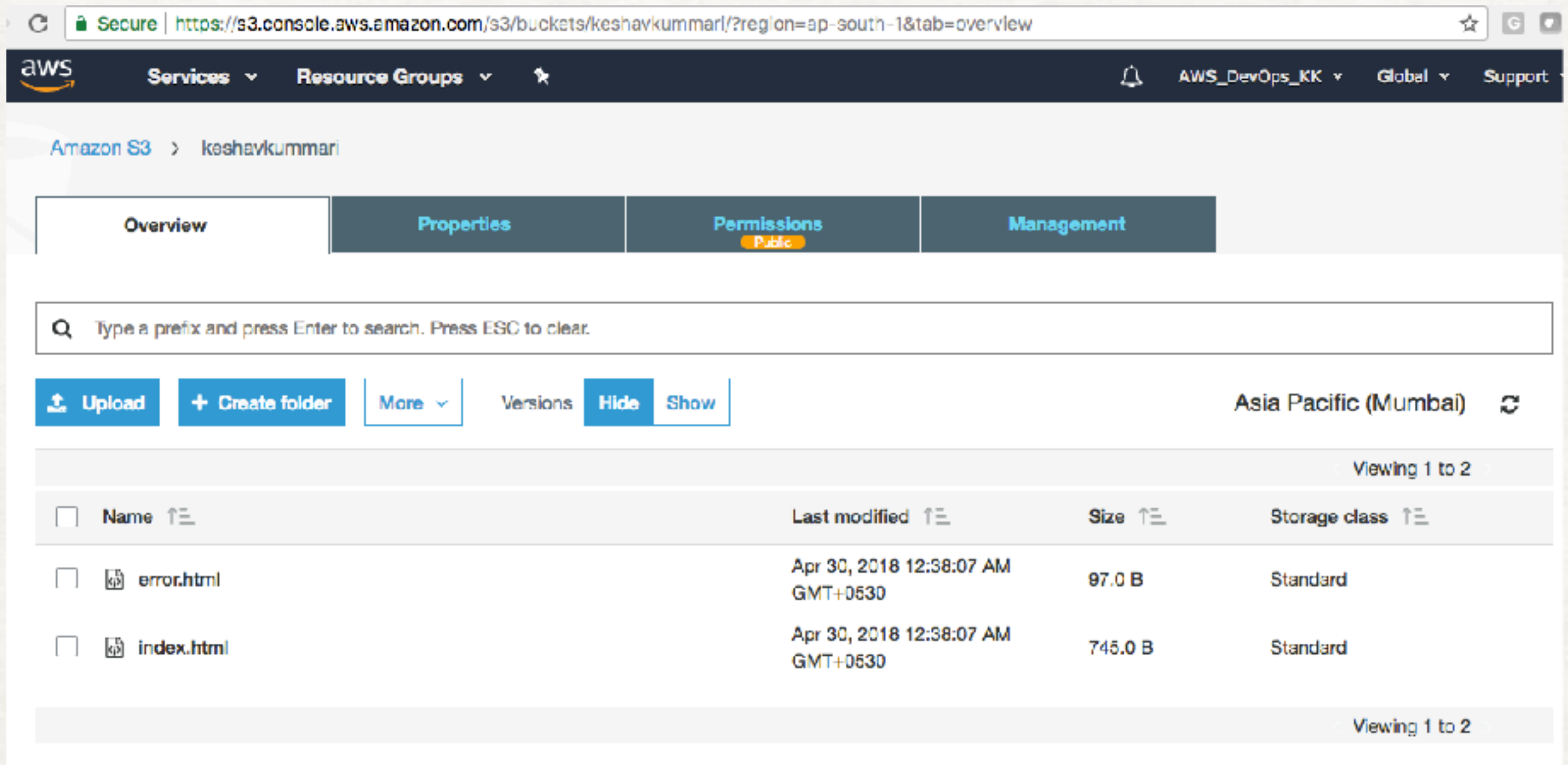
The screenshot shows the AWS S3 console interface. The main header includes the AWS logo, 'Services', 'Resource Groups', and a user profile 'AWS_DevOps_KK'. The breadcrumb trail indicates the current location is 'Amazon S3 > keshavkumhari'. The 'Overview' tab is selected. A modal window titled 'Upload' is open, showing a progress bar with four steps: 1. Select files, 2. Set permissions, 3. Set properties, and 4. Review. The first step is active. Below the progress bar, it shows '2 Files', 'Size: 842.0 B', and 'Target path: keshavkumhari'. A table lists the files: 'error.html' (97.0 B) and 'index.html' (745.0 B). At the bottom of the modal, there are 'Upload' and 'Next' buttons.

File Name	Size	Action
error.html	97.0 B	X
index.html	745.0 B	X

Click on Next & Grant Public Read Access on those two files

STEP - 16 : FILES ARE UPLOADED

- Cross check those two files access



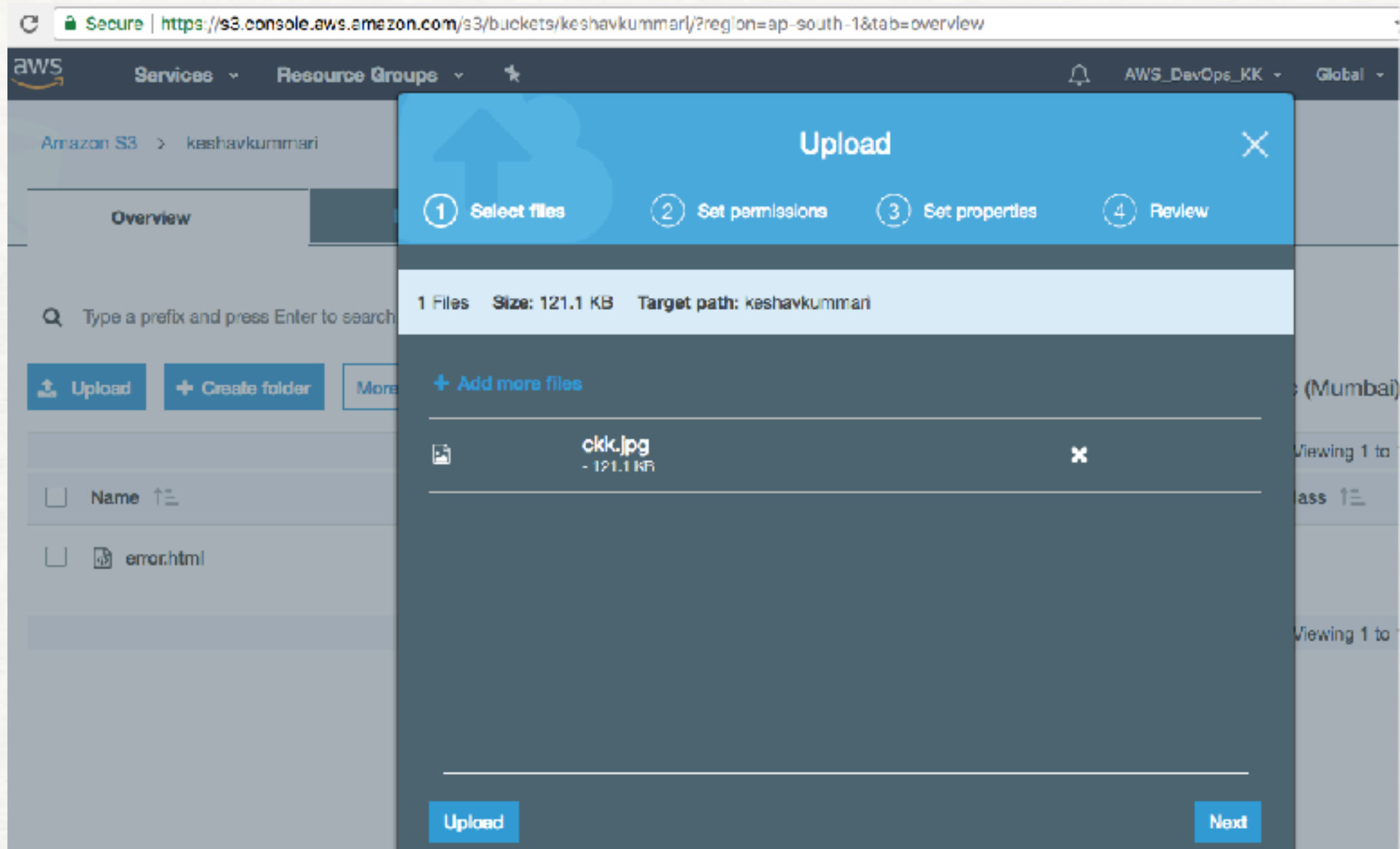
The screenshot shows the AWS S3 console interface for a bucket named 'keshavkummar'. The breadcrumb navigation shows 'Amazon S3 > keshavkummar'. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information 'AWS_DevOps_KK' with a 'Global' dropdown and a 'Support' link. Below the breadcrumb, there are four tabs: 'Overview' (selected), 'Properties', 'Permissions' (with a 'Public' badge), and 'Management'. A search bar is present with the placeholder text 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar, there are buttons for 'Upload', '+ Create folder', and a 'More' dropdown menu. To the right of these buttons are 'Versions', 'Hide', and 'Show' buttons. The region 'Asia Pacific (Mumbai)' is displayed with a refresh icon. The main content area shows a table of objects with columns: 'Name', 'Last modified', 'Size', and 'Storage class'. The table indicates 'Viewing 1 to 2' objects. The objects listed are 'error.html' (97.0 B, Standard storage class) and 'index.html' (745.0 B, Standard storage class), both modified on 'Apr 30, 2018 12:38:07 AM GMT+0530'. A footer bar at the bottom of the table area also indicates 'Viewing 1 to 2'.

Name	Last modified	Size	Storage class
error.html	Apr 30, 2018 12:38:07 AM GMT+0530	97.0 B	Standard
index.html	Apr 30, 2018 12:38:07 AM GMT+0530	745.0 B	Standard

Go to Browser & check index.html & error.html URLs

STEP - 17 : UPLOAD A IMAGE

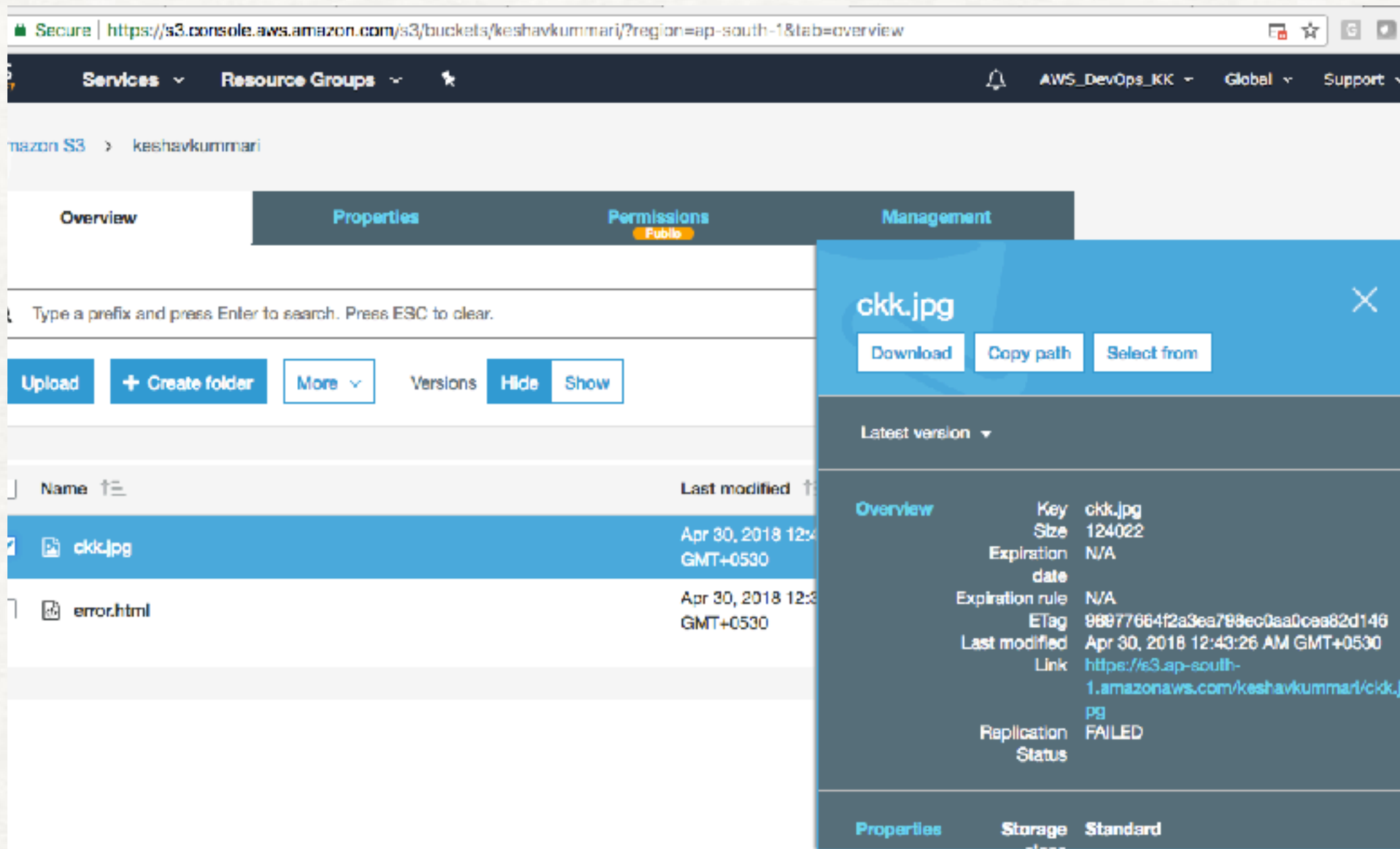
- Grant Public Read Access & Copy the URL & place in index.html file



Cross check the Public Read Access

STEP - 18 : COPY THE URL OF THE IMAGE

- Go to index.html & paste in it



Check the URL status

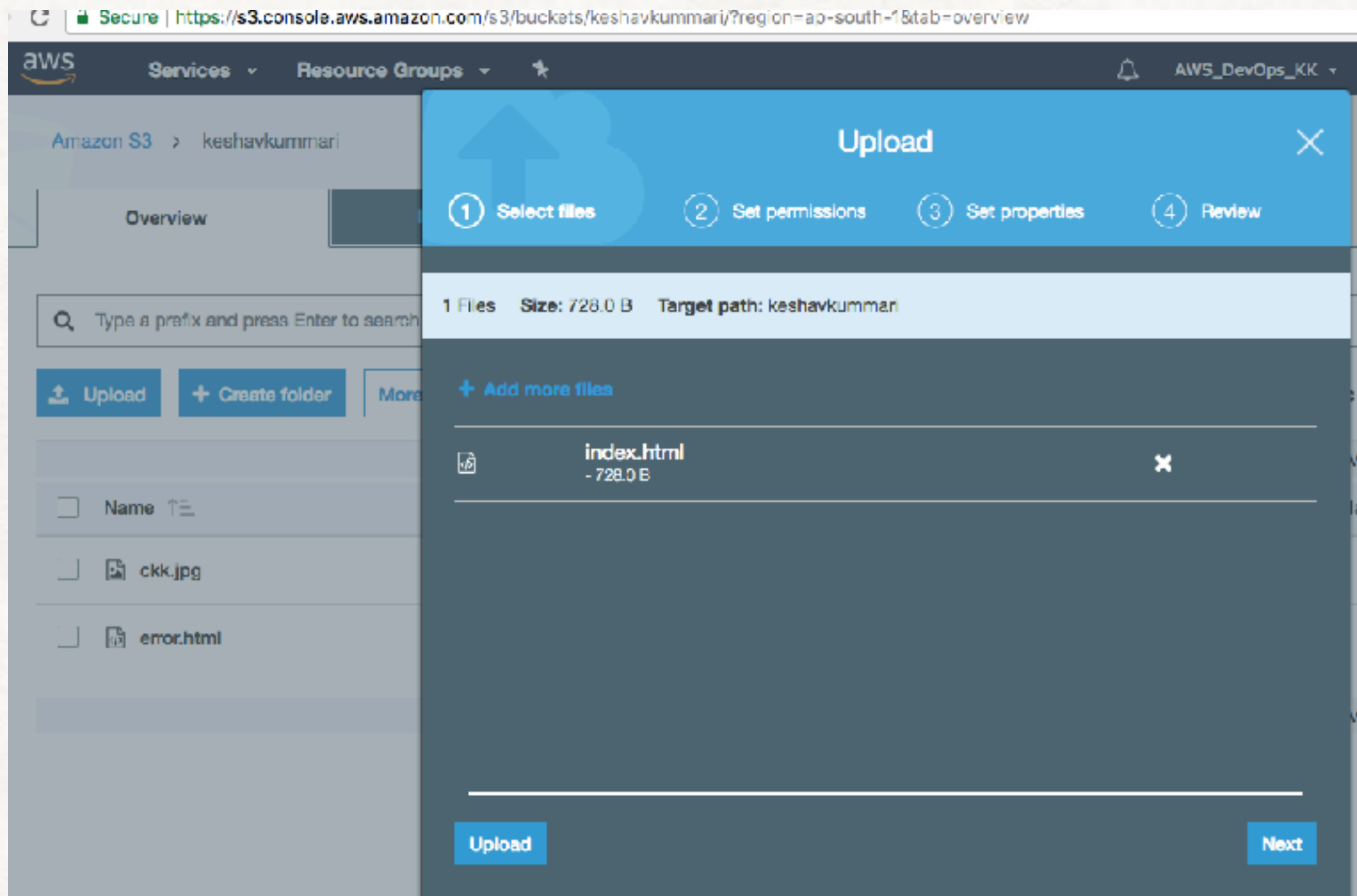
STEP - 19 : OPEN THE INDEX.HTML & ADD THE IMAGE URL

- Go to index.html & paste in it

```
1 <html>
2   <head>
3     <script>
4       function myFunction() {
5         var xhttp = new XMLHttpRequest();
6         xhttp.onreadystatechange = function() {
7           if (this.readyState == 4 && this.status == 200) {
8             document.getElementById("my-demo").innerHTML = this.responseText;
9           }
10        };
11        xhttp.open("GET", "https://erqwjzexg.execute-api.ap-south-1.amazonaws.com/Prod/MyTestFunction", true);
12        xhttp.send();
13      }
14    </script>
15  </head>
16  <body>
17    <div align="center">
18      <br>
19      <br>
20      <br>
21      <br>
22      <h1>Hello <span id="my-demo">Online Ucator</span></h1>
23      <button onclick="myFunction()">Click me</button>
24      <br>
25      
26    </div>
27  </body>
28 </html>
```

Lambda Function URL & S3 bucket URL

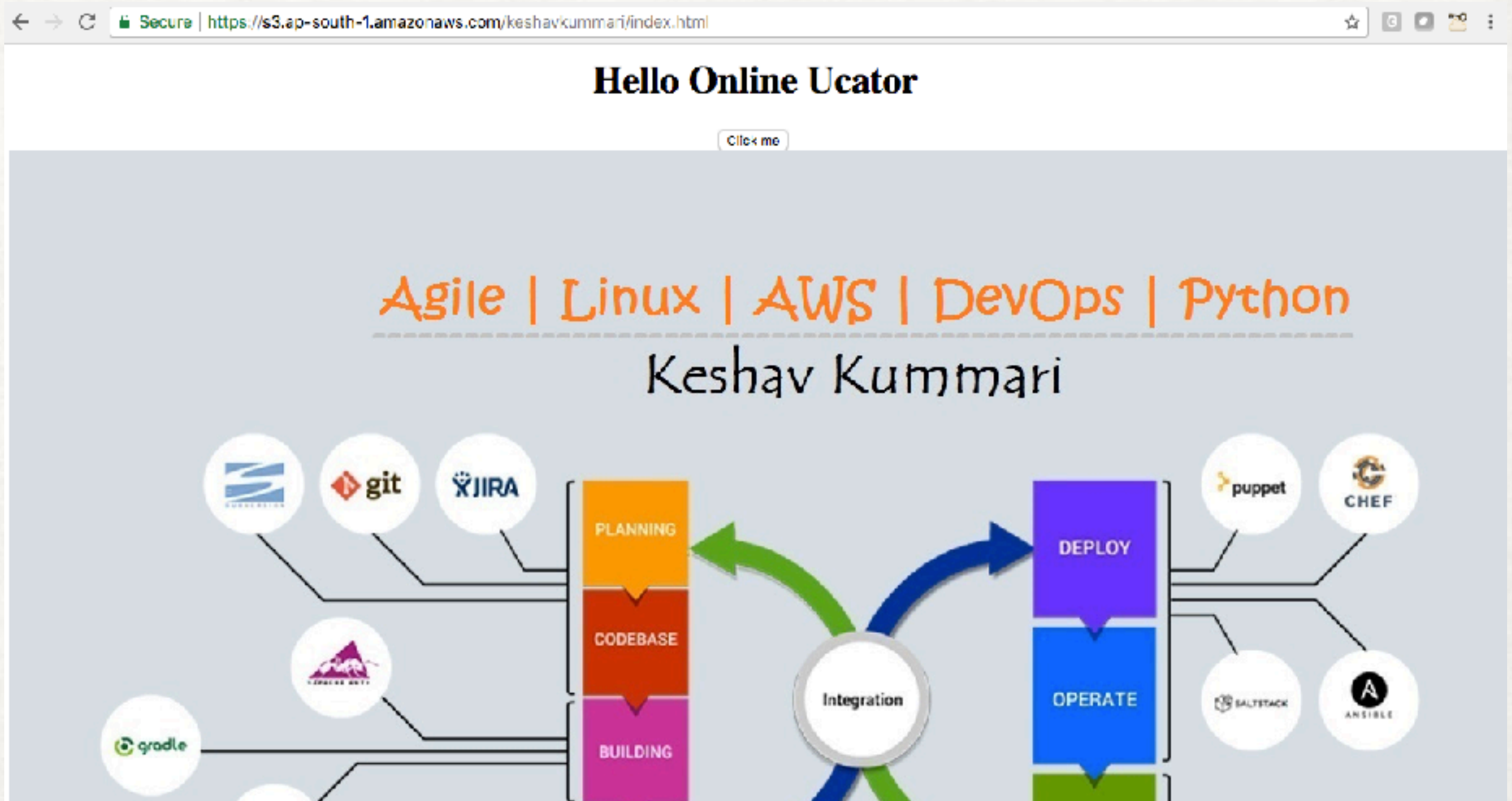
STEP - 20 : UPLOAD INDEX.HTML FILE AGAIN



As we added S3 bucket image url

STEP - 21 : COPY S3 BUCKET INDEX.HTML URL & CROSS CHECK IN THE BROWSER

- Then Click on "Click Me" so, that Lambda Function will execute



S3 & Lambda Function Execution

STEP - 22 : CLICK ON CLICK ME

- Lambda Function has been executed & results are printed



Now, we see Lambda Function has been expected