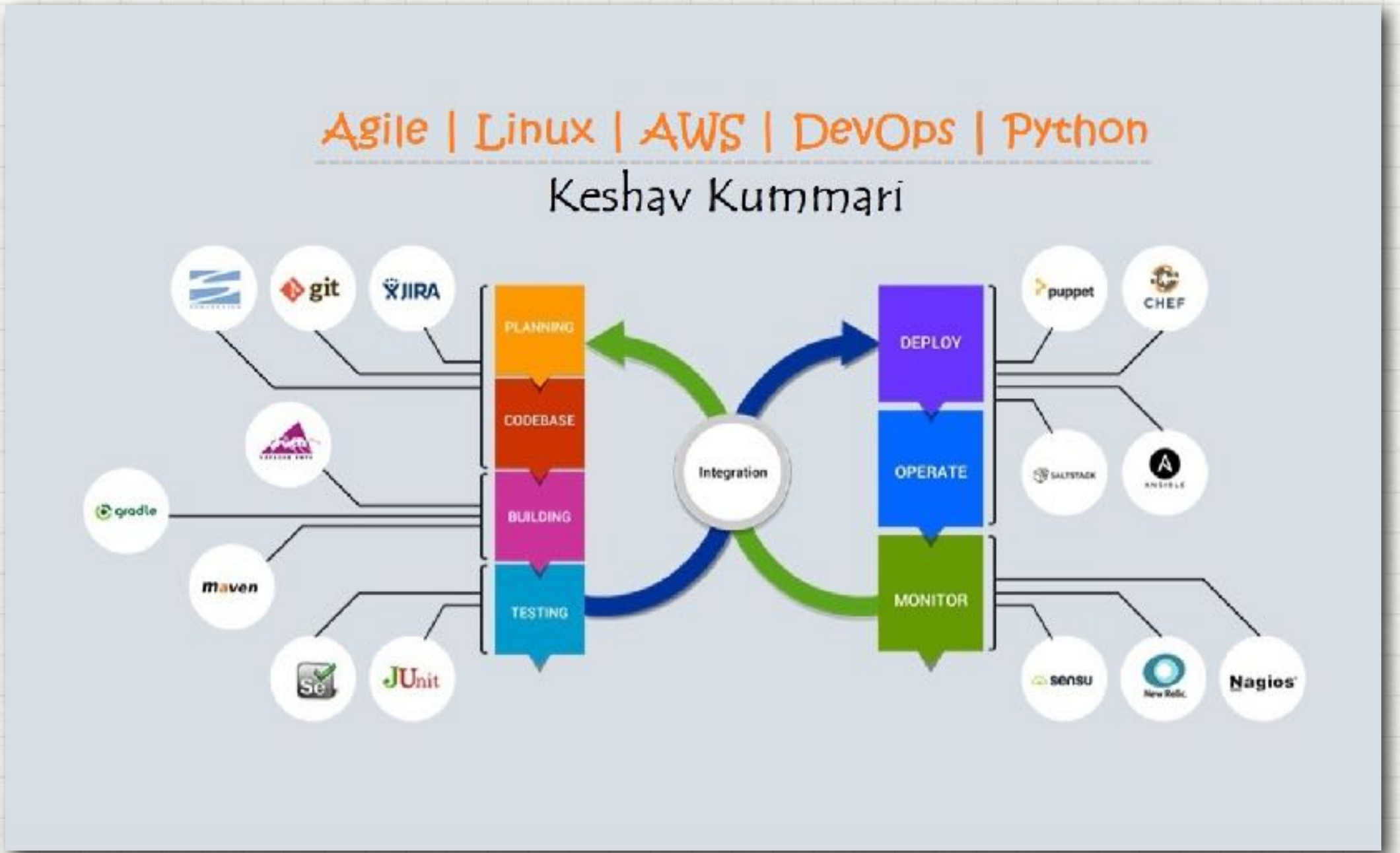
# AWS

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.

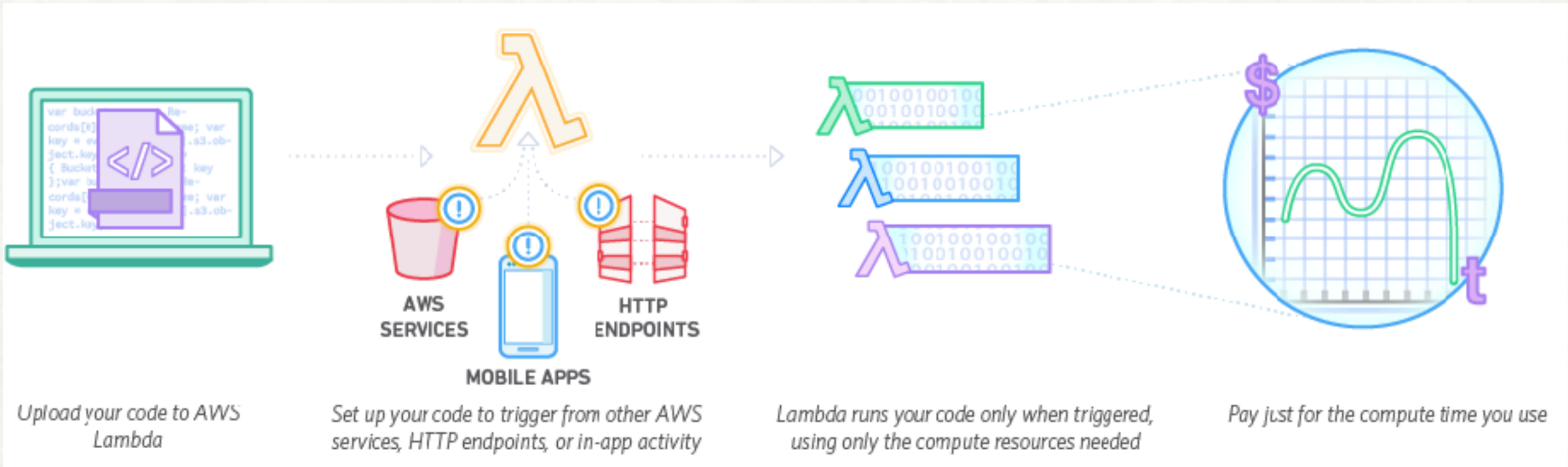
### SUBSECOND METERING

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.

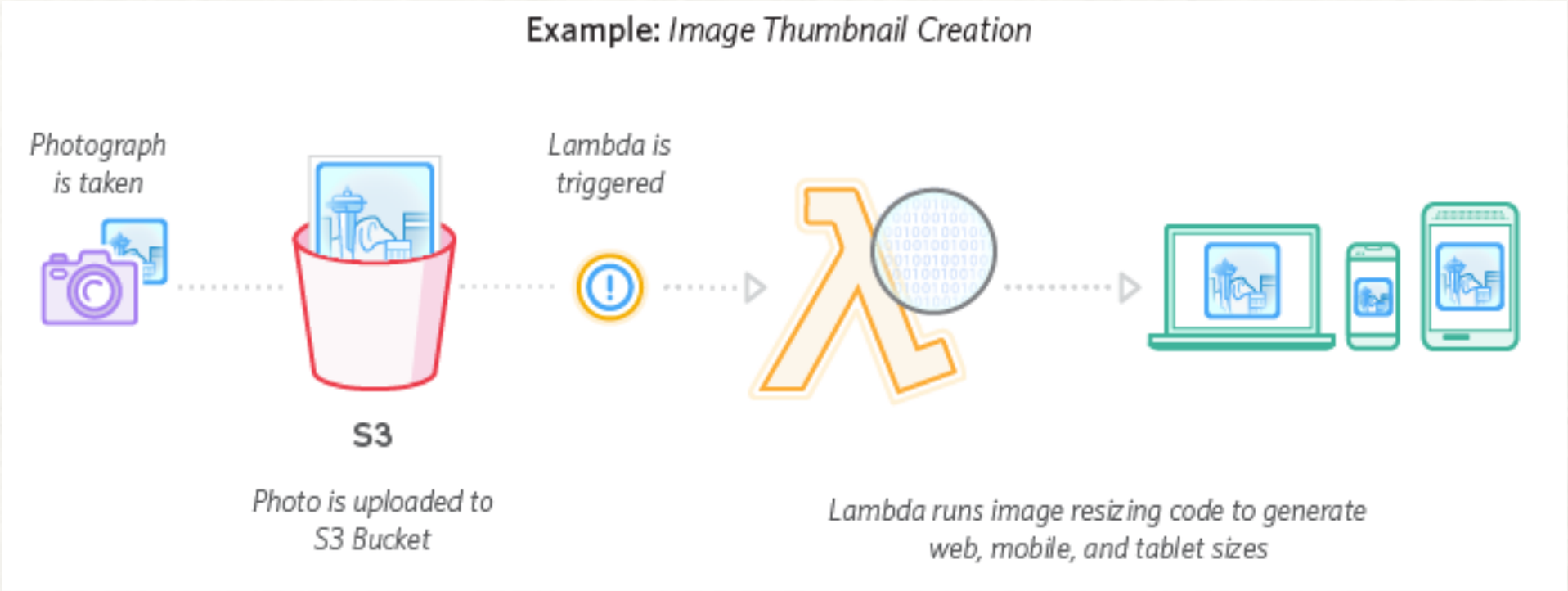
- This allows you to build a variety of real-time serverless data processing systems.

# 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.

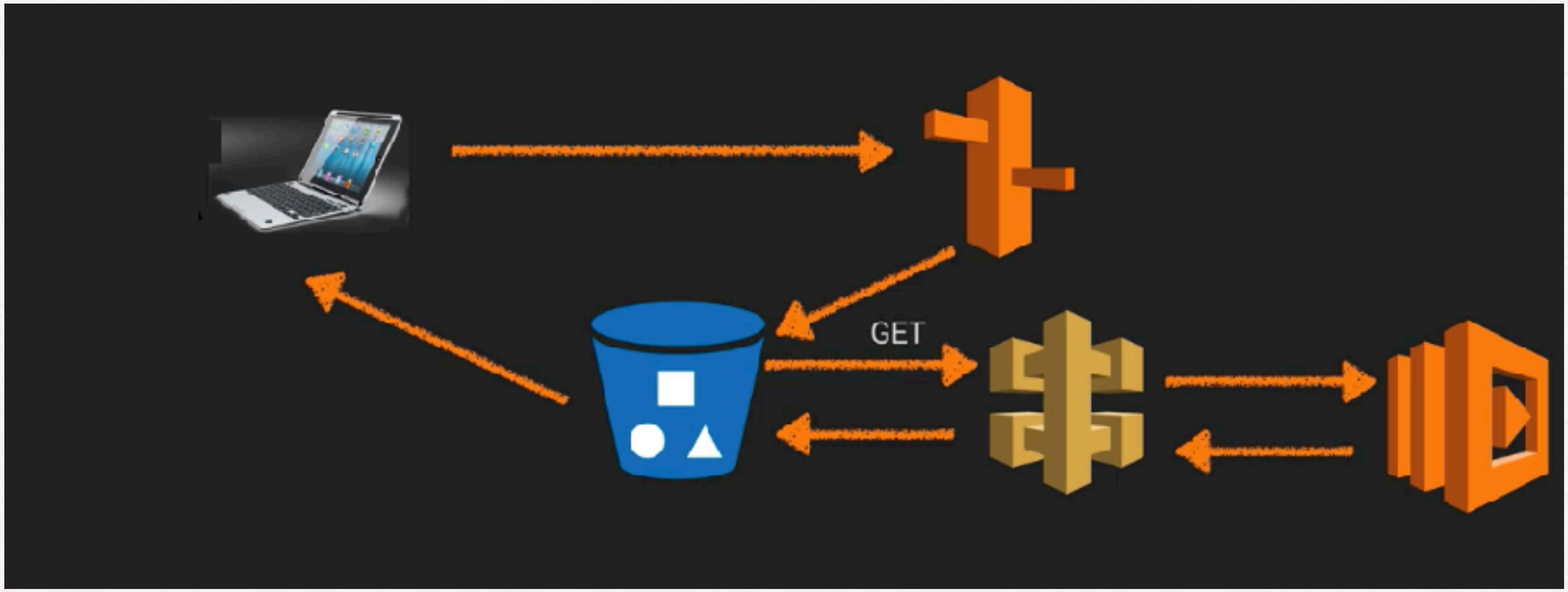
- Example




Example of AWS Lambda

# LAMBDA FUNCTION EXAMPLE

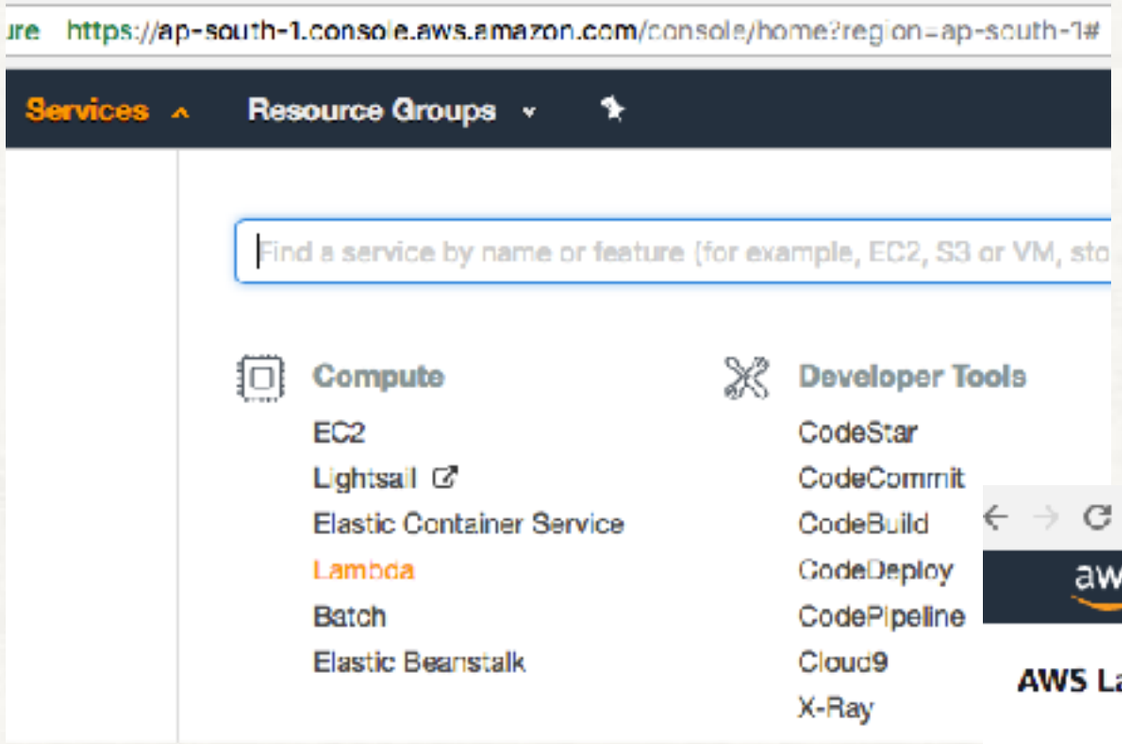
- Sample Lambda Flow Diagram



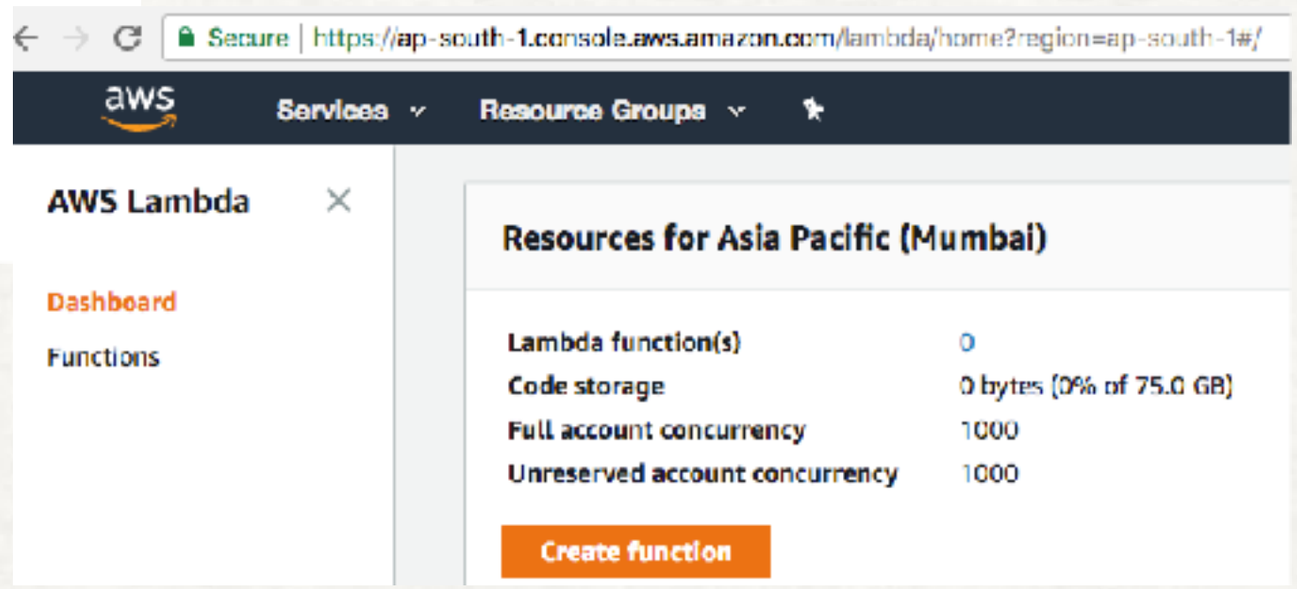

User, Route 53, S3 Bucket, API Gateway & Lambda Function

# STEP - 1 : GO TO AWS MGMT CONSOLE

- Click on Lambda




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.

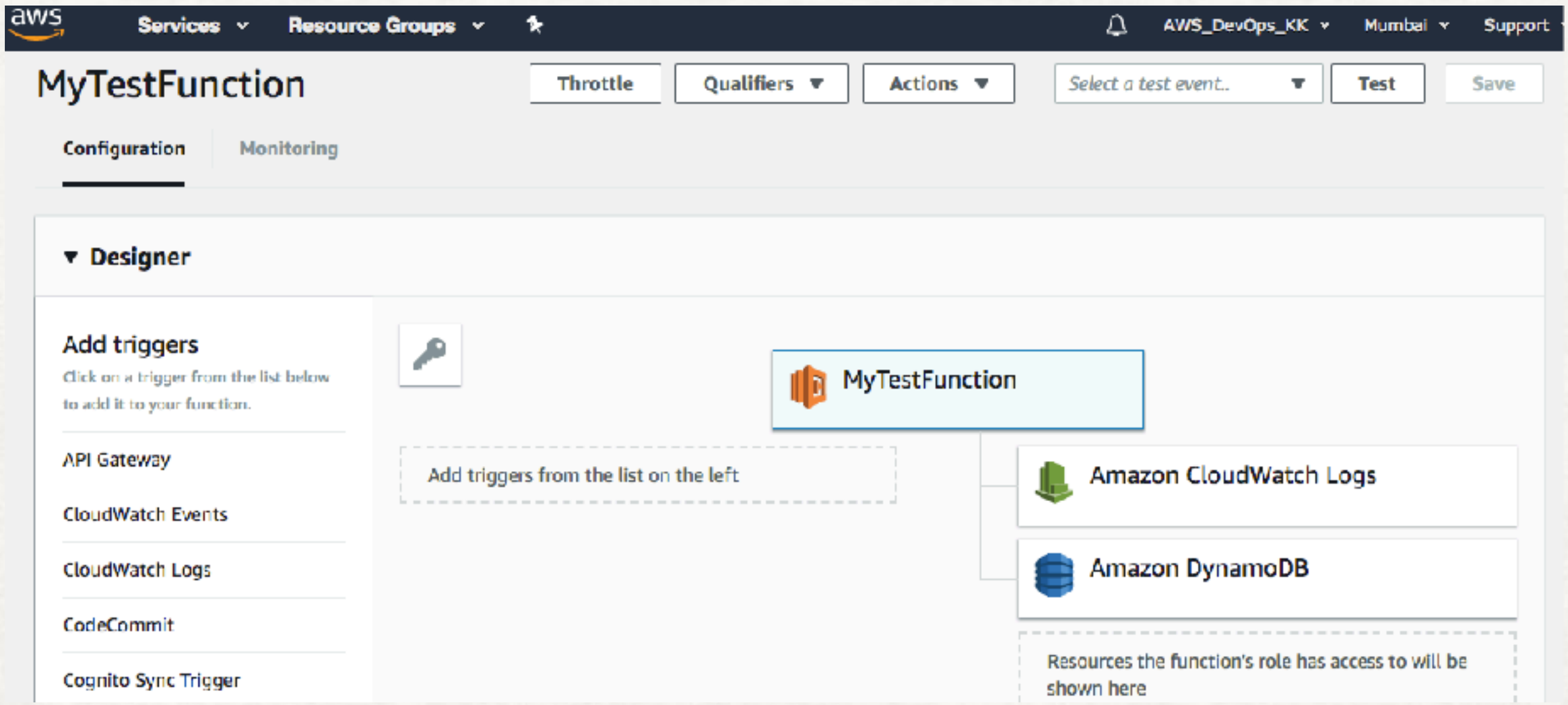
Simple Microservice permissions ✕

Cancel | Create function

Click on Create Function

# STEP-3 : CROSS CHECK REQUIRED DETAILS

- Scroll Down & Check required details




Scroll Down & Add Description under Basic Settings

# STEP-4 : SCROLL DOWN & ADD DESCRIPTION

- Add Description

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

aws Services Resource Groups AWS_DevOps_KK Mumbai Support

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

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

aws Services Resource Groups AWS_DevOps_KK Mumbai Support

MyTestFunction

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

**Function code** Info

Code entry type: Edit code inline

Runtime: Python 3.6

Handler Info: lambda_function.lambda_handler

File Edit Find View Goto Tools Window

Environment

MyTestFunction

lambda_function.py

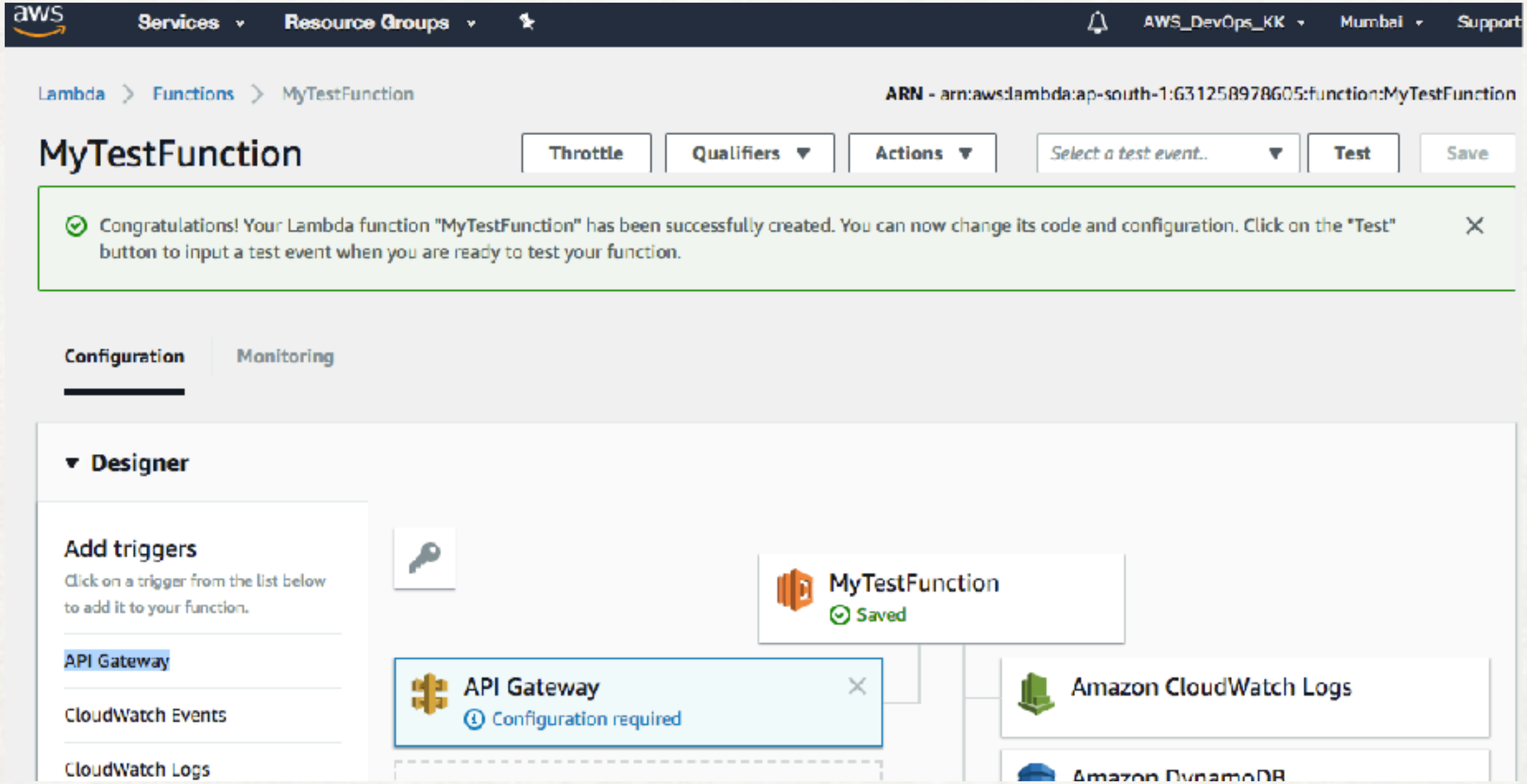
lambda_function

```
def lambda_handler(event, context):
    print("In lambda handler")

    resp = {
        "statusCode": 200,
        "headers": {
            "Access-Control-Allow-Origin": "*",
        },
        "body": "Keshav Kumari"
    }

    return resp
```

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”.




Scroll Down & Fill the required fields

# STEP- 7 : CONFIGURE THE TRIGGERS

## ADD THE REQUIRED FIELDS ACCORDINGLY

aws Services Resource Groups AWS_DevOps_KK Mumbai

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

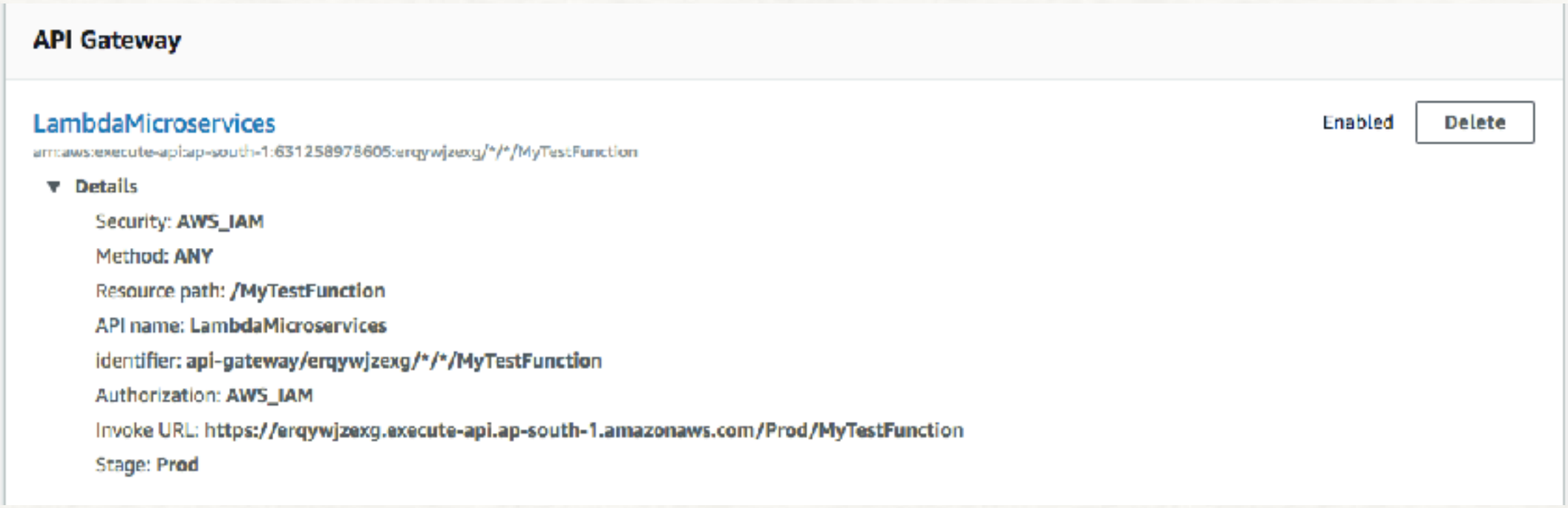
API name
Enter a name to uniquely identify your API.
LambdaMicroservices

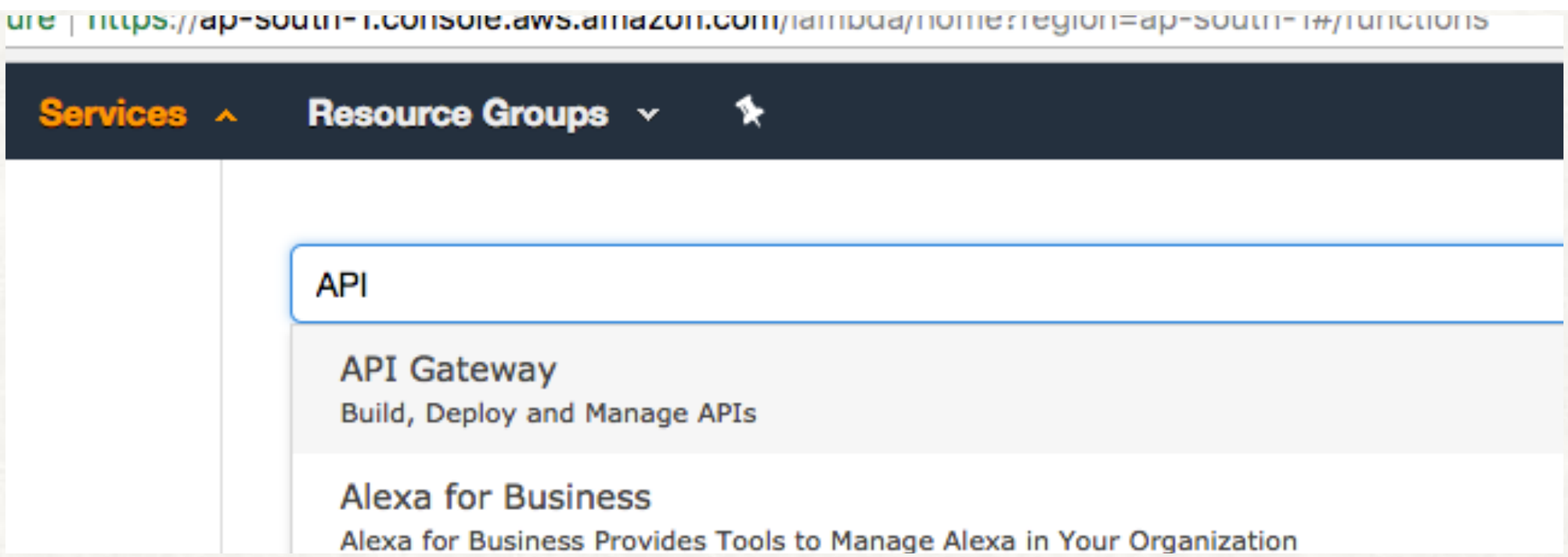
Deployment stage
The name of your API's deployment stage.
Prod

Security
Configure the security mechanism for your API endpoint.
AWS IAM

Click on Add & Save the changes

# STEP - 8 : CROSS CHECK THE API GATEWAY

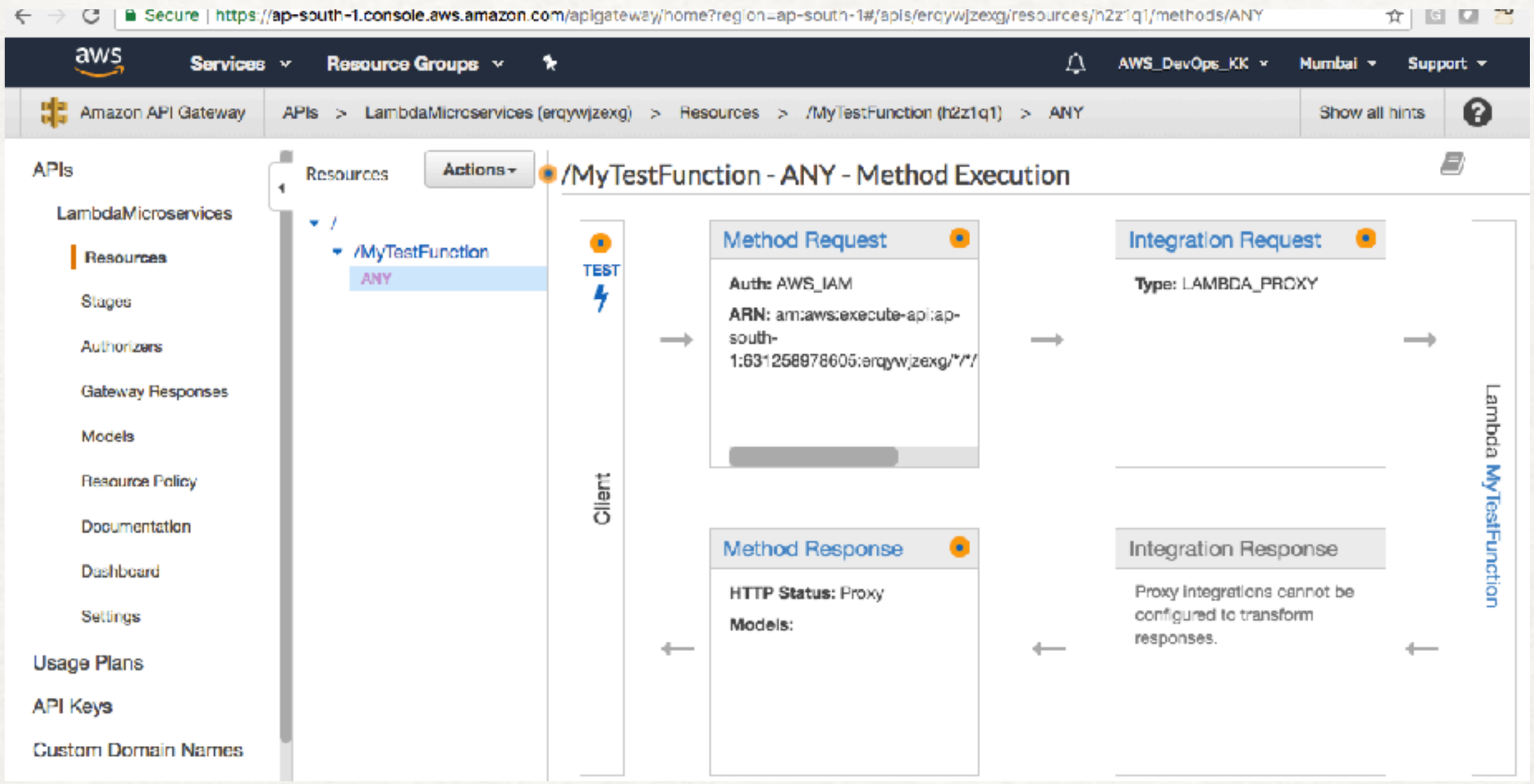



Go to AWS Mgmt Console & Look for API Gateway

# STEP - 9 : GO TO API GATEWAY

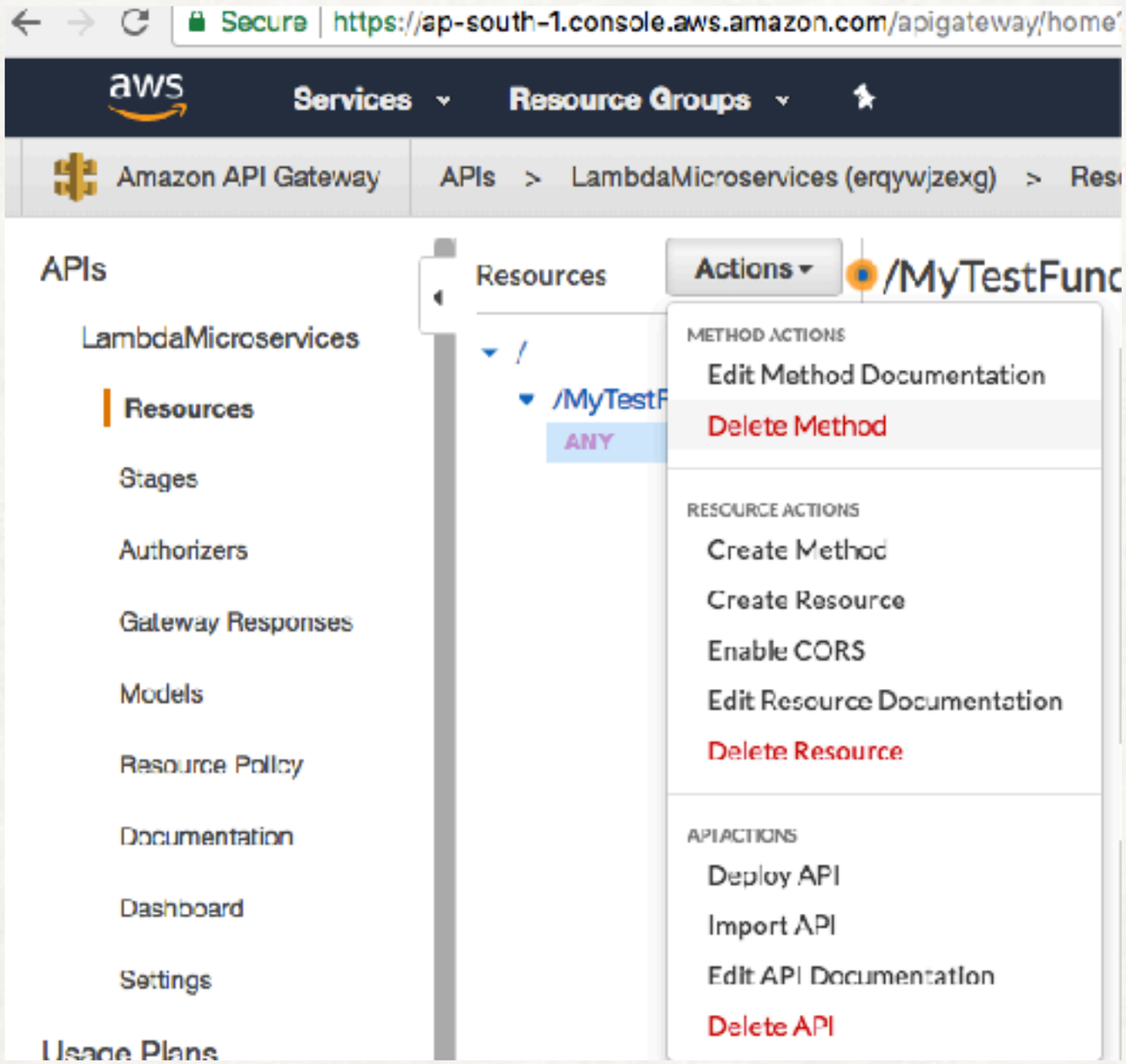
## CLICK ON RESOURCES & CLICK ON "ANY" METHOD



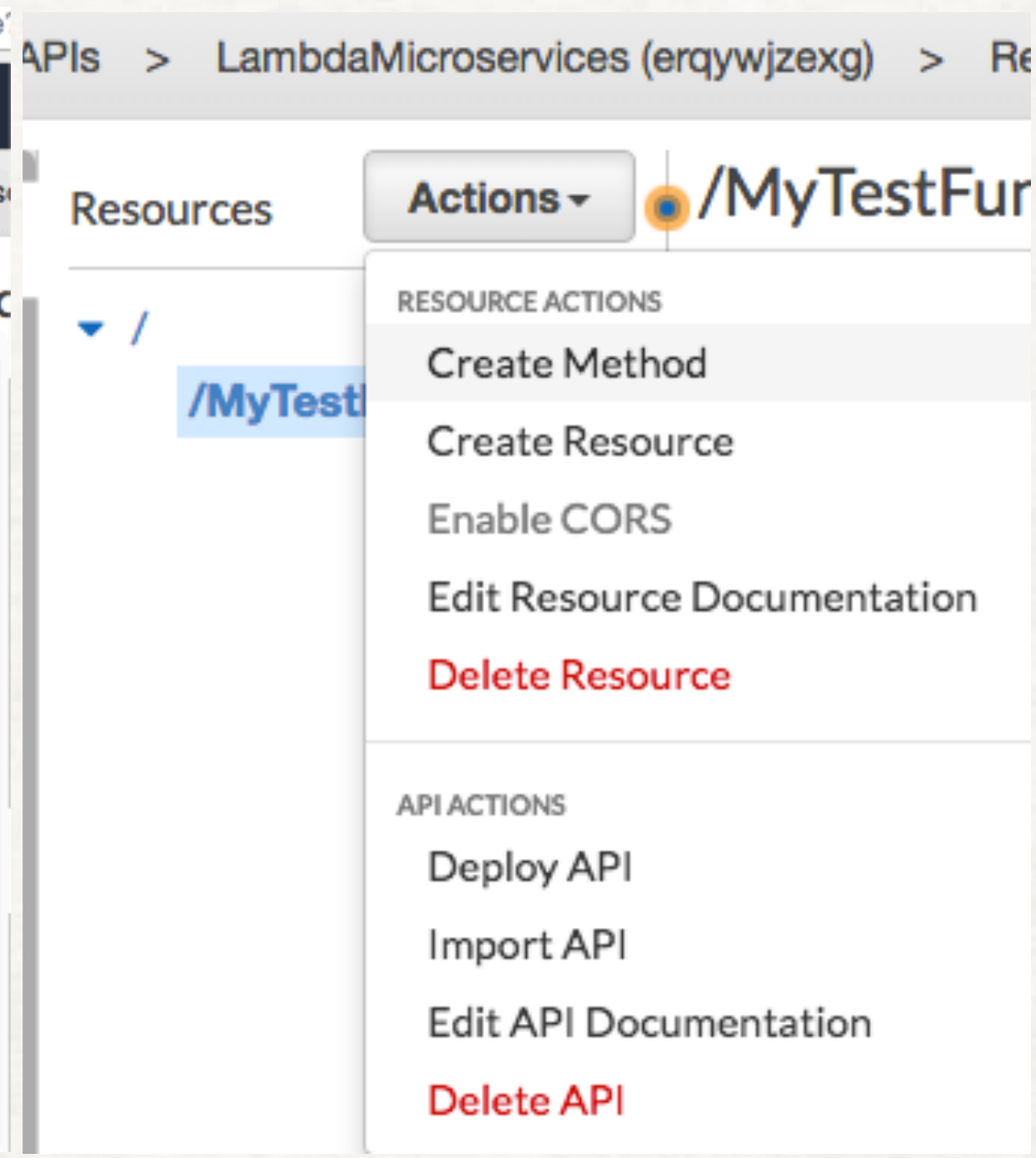

Cross check the details

# 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

/MyTestFunction - GET - Setup

Choose the integration point for your new method.

**Integration type**
- (•) Lambda Function
- ( ) HTTP
- ( ) Mock
- ( ) AWS Service
- ( ) VPC Link

**Use Lambda Proxy integration** [x]

**Lambda Region** ap-south-1

**Lambda Function**

MyTestFunction

**Use Default Timeout** [x]

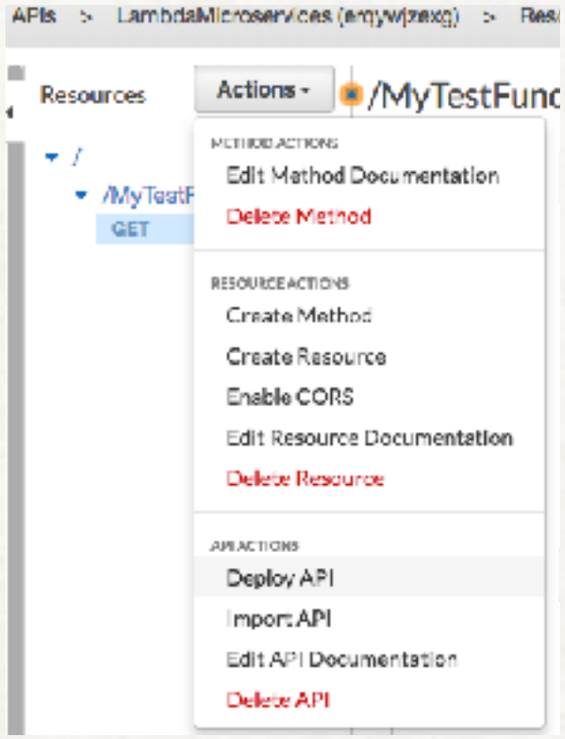
Save

Click on SAVE

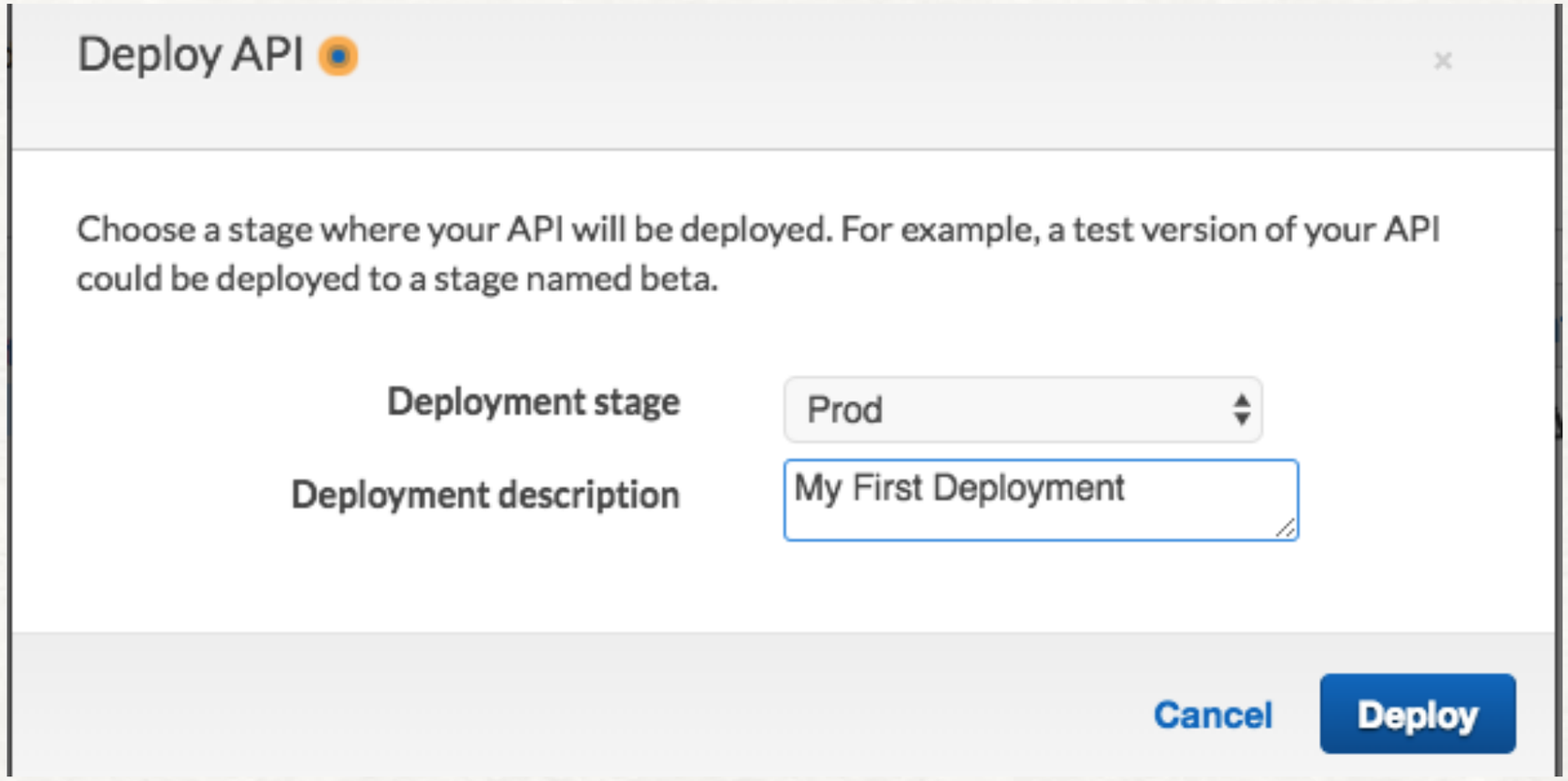
# STEP - 12 : DEPLOY

## GET METHOD

- Go to Actions & Deploy API



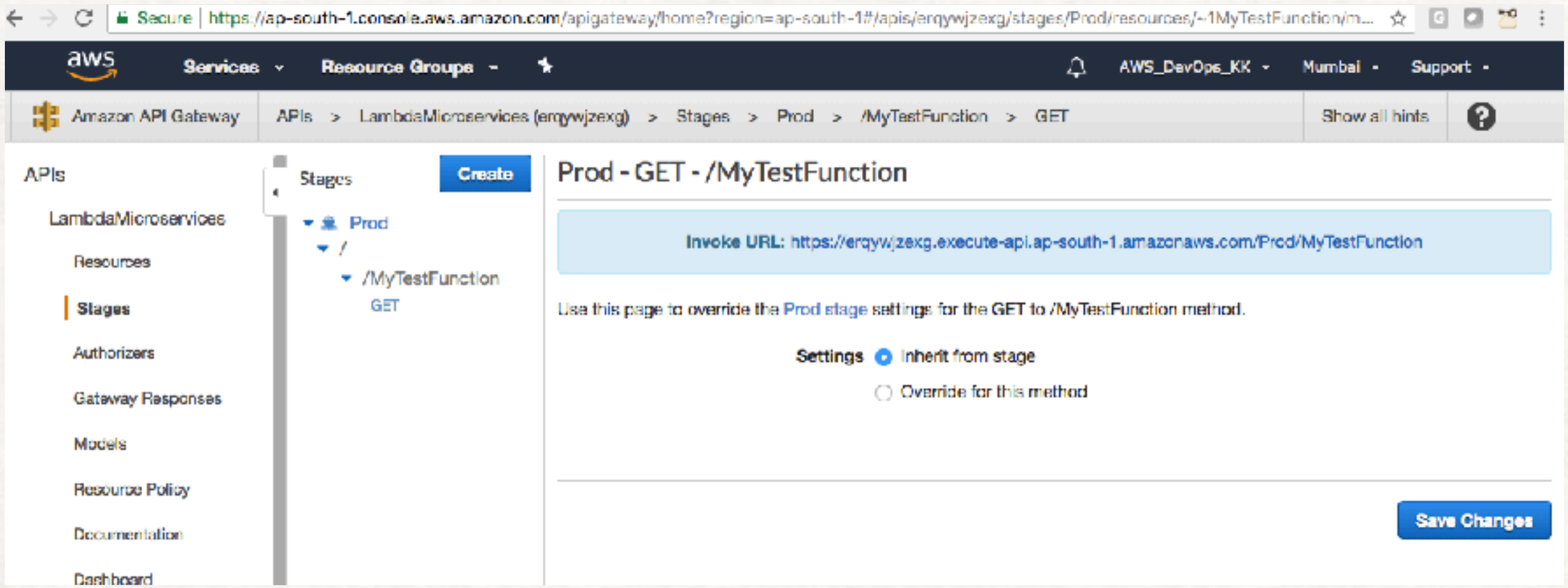

Click on Deploy API



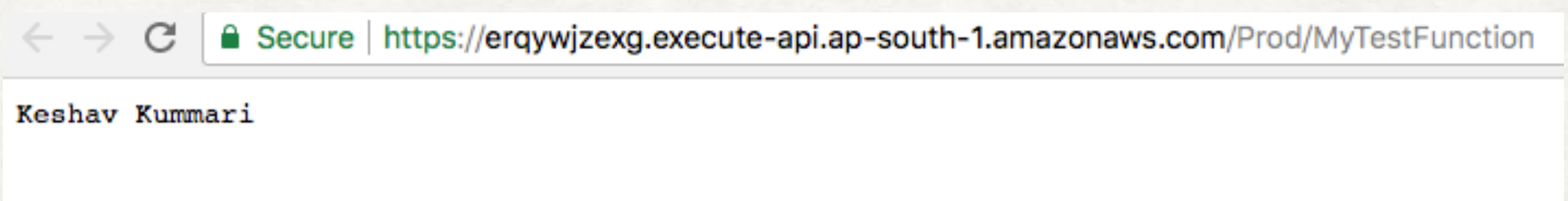

Click on Deploy

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

- Click on Stages & Click on GET method




Copy the URL & go to the browser & Check

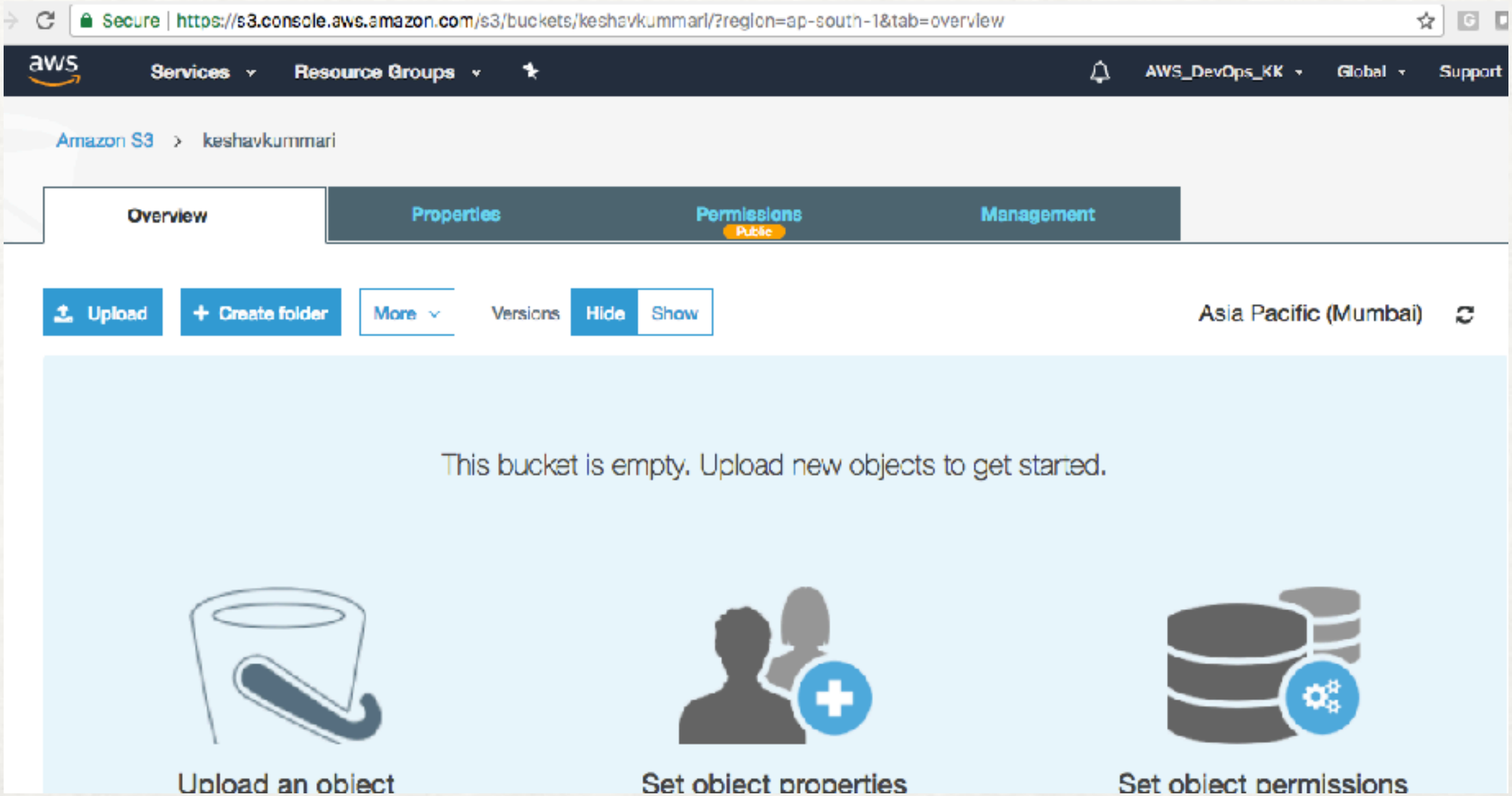



Lambda has been launched a server less website

# STEP - 14 : GO TO S3 BUCKET

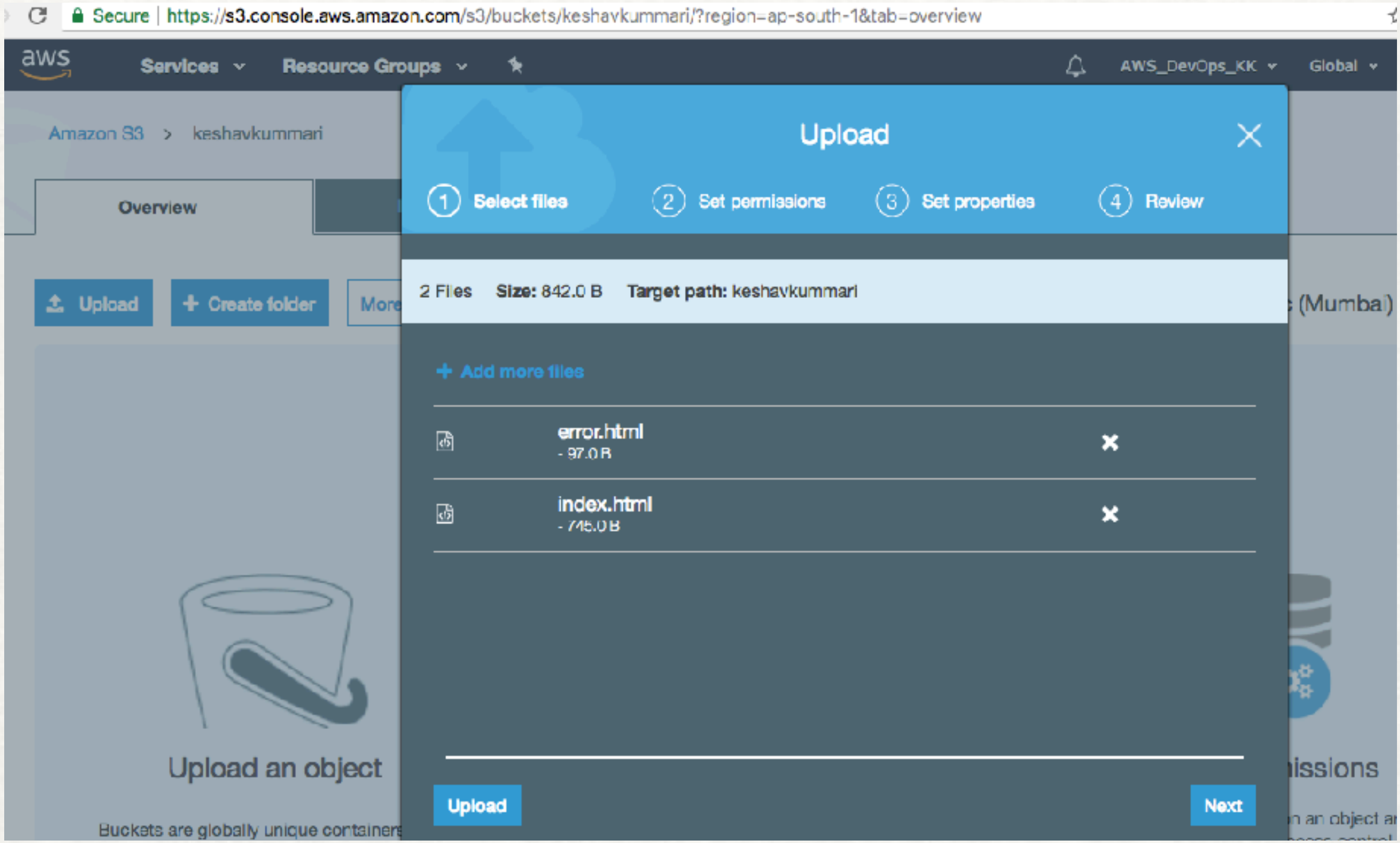
## CREATE A BUCKET I.E. KESHAVKUMMARI

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




S3 Bucket

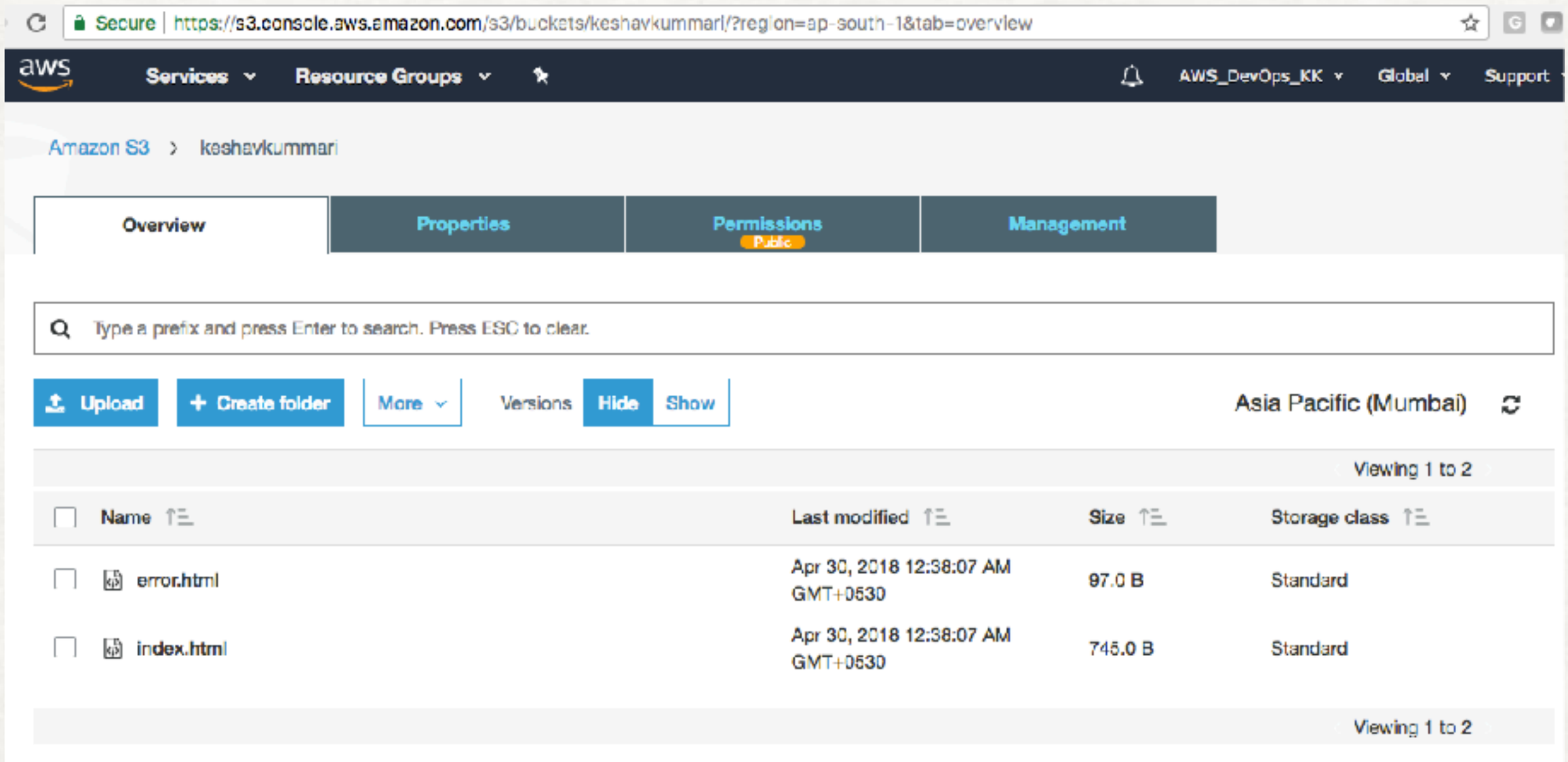
# STEP - 15 : UPLOAD FILES




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

# STEP - 16 : FILES ARE UPLOADED

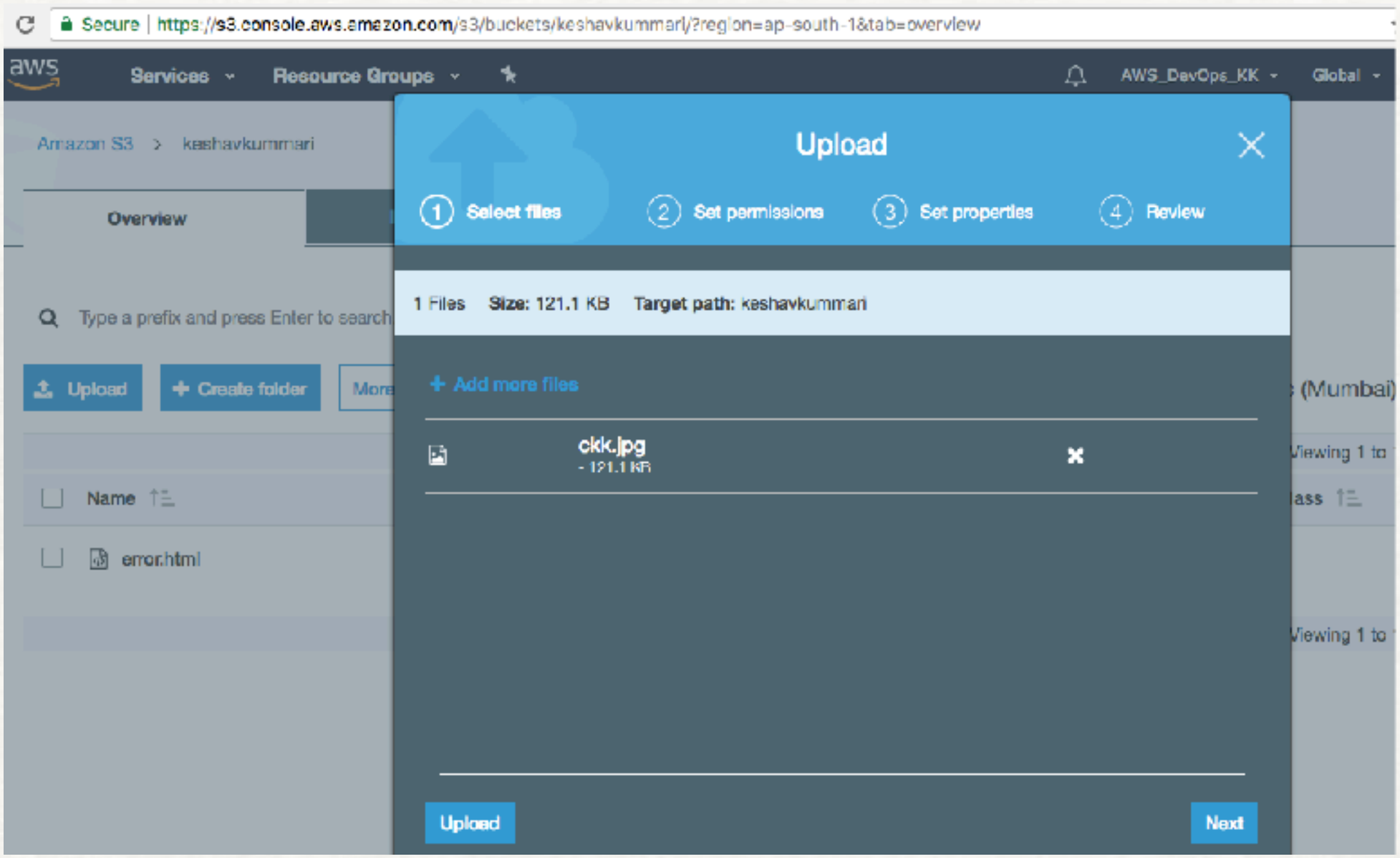
- Cross check those two files access




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

# STEP - 17 : UPLOAD A IMAGE

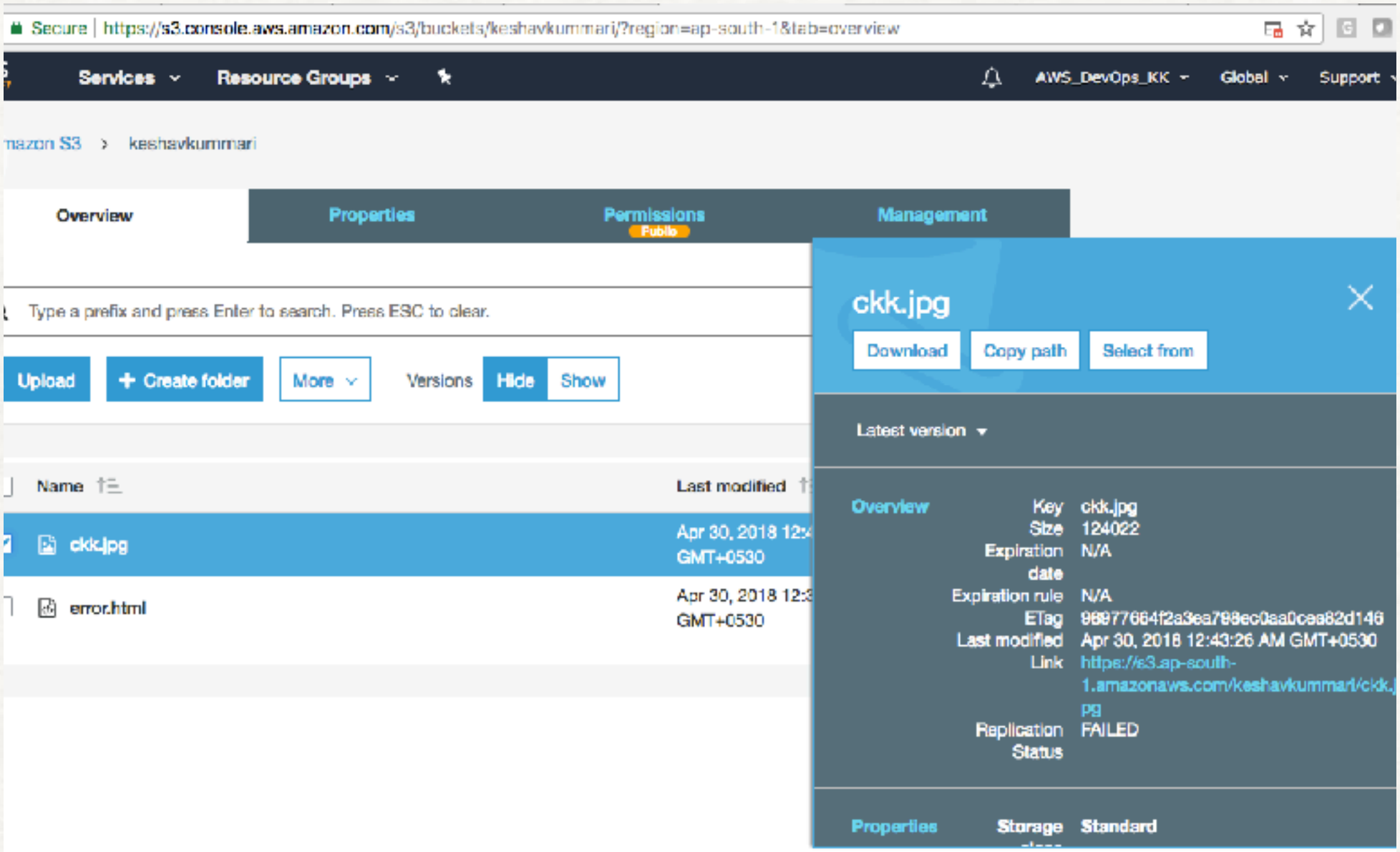
- Grant Public Read Access & Copy the URL & place in index.html file




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

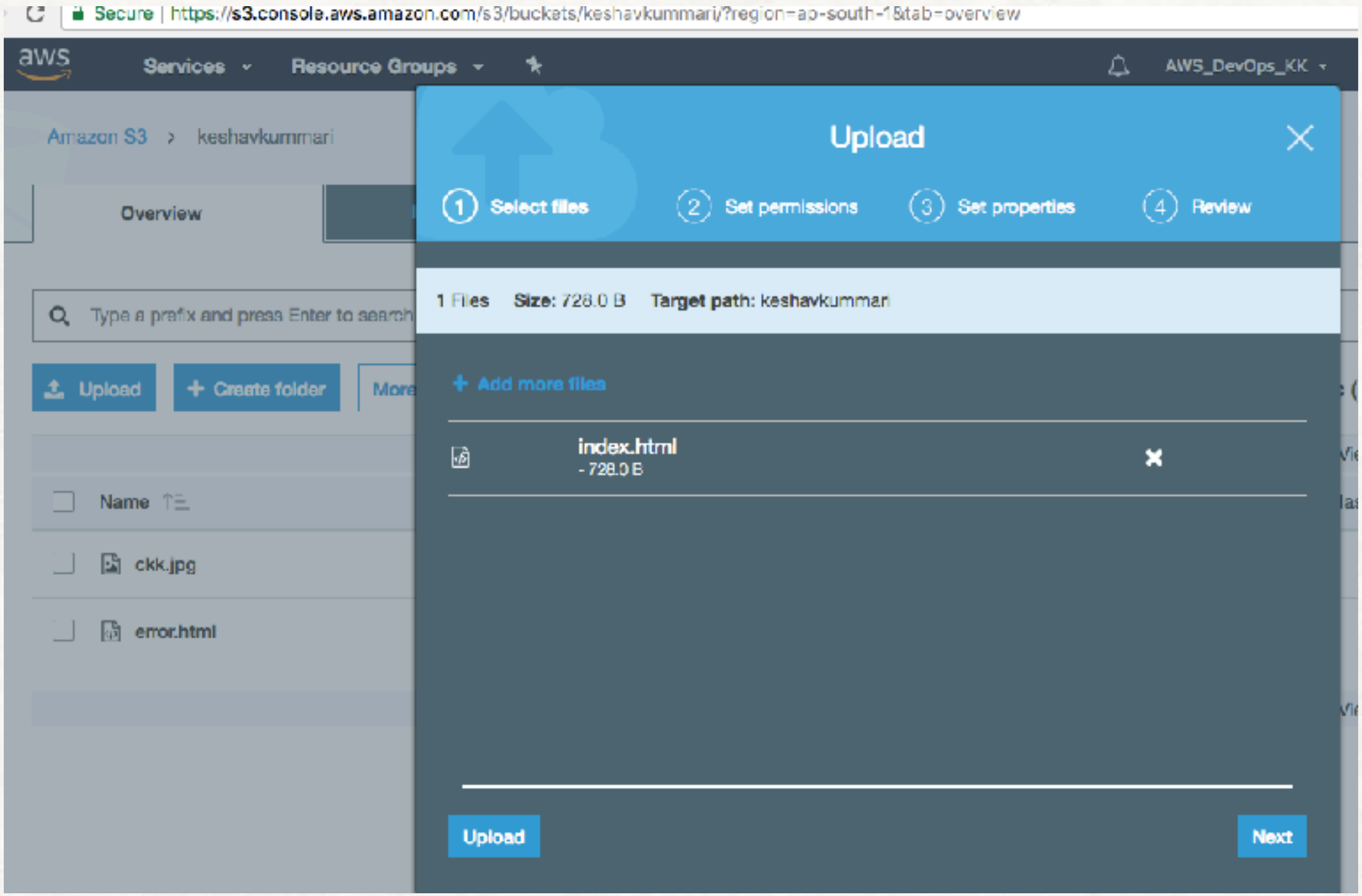
- Go to index.html & paste in it

index.html

```html
<html>
    <head>
        <script>
            function myFunction() {
                var xhttp = new XMLHttpRequest();
                xhttp.onreadystatechange = function() {
                    if (this.readyState == 4 && this.status == 200) {
                    document.getElementById("my-demo").innerHTML = this.responseText;
                    }
                };
                xhttp.open("GET", "https://erqywjzexg.execute-api.ap-south-1.amazonaws.com/Prod/MyTestFunction", true);
                xhttp.send();

            }

        </script>
    </head>
<body>
    <div align="center">
        <br>
        <br>
        <br>
        <br>
        <h1>Hello <span id="my-demo">Online Ucator</span></h1>
        <button onclick="myFunction()">Click me</button>
        <br>
        <img src="https://s3.ap-south-1.amazonaws.com/keshavkumari/ckk.jpg">
    </div>

    </body>
</html>
```

Lambda Function URL & S3 bucket URL

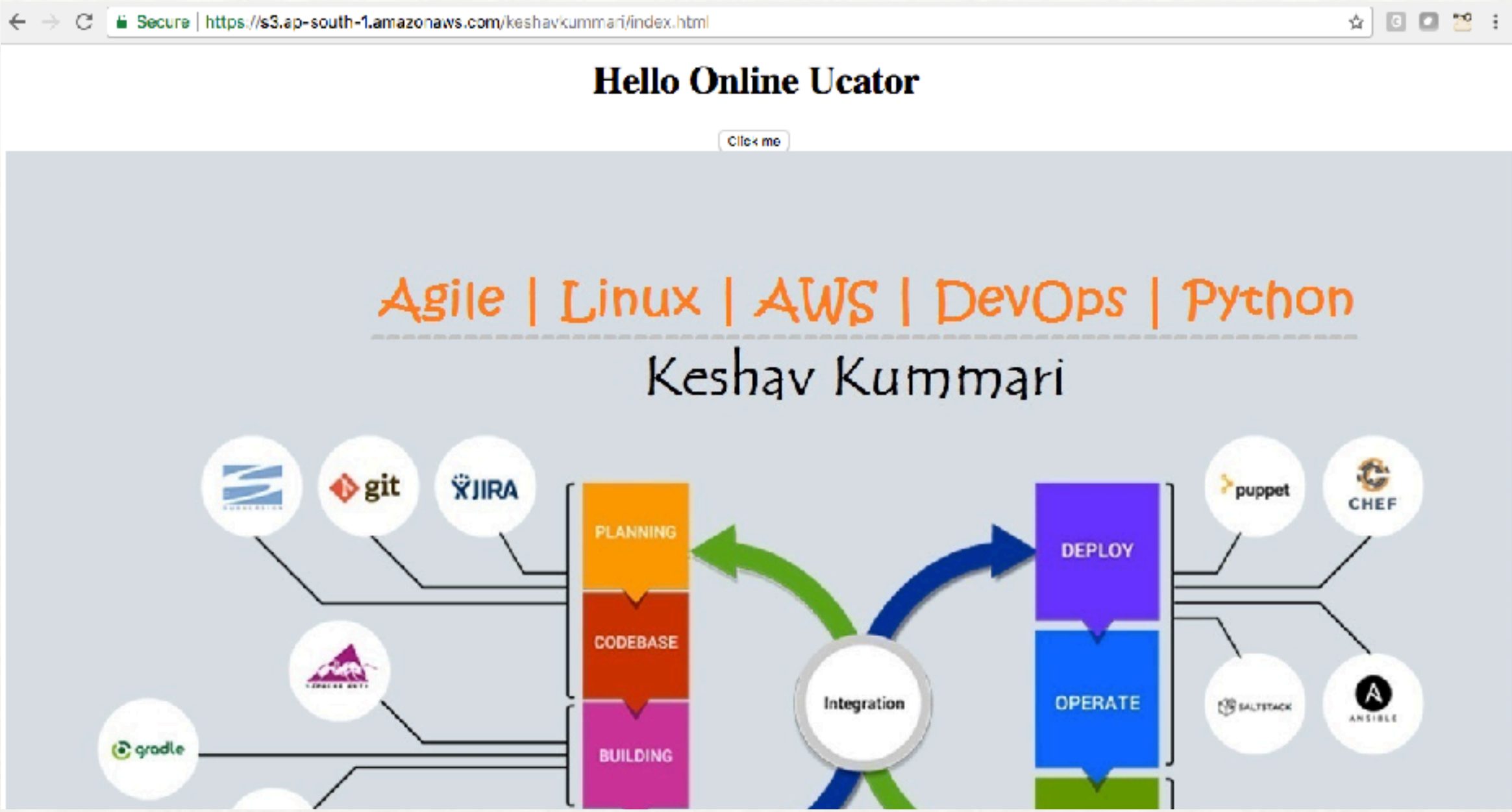
# STEP - 20 : UPLOAD INDEX.HTML FILE AGAIN




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