CASE STUDY ON

AWS API GATEWAY



**INTRODUCTION**

Amazon **API Gateway** is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the [AWS Cloud](https://aws.amazon.com/what-is-cloud-computing/). As an API Gateway API developer, you can create APIs for use in your own client applications. Or you can make your APIs available to third- party app developers

API Gateway creates RESTful APIs that:

* Are HTTP-based.
* Enable stateless client-server communication.
* Implement standard HTTP methods such as GET, POST, PUT, PATCH, and DELETE.

# API Gateway architecture diagram Architecture of API Gateway

This diagram illustrates how the APIs you build in Amazon API Gateway provide you or your developer customers with an integrated and consistent developer experience for building AWS serverless applications. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.

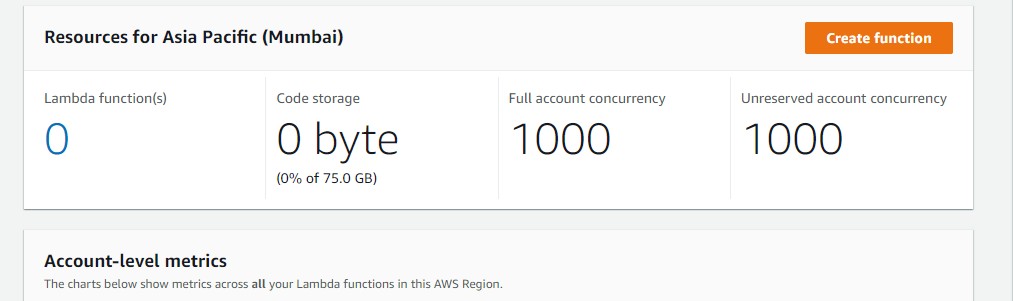
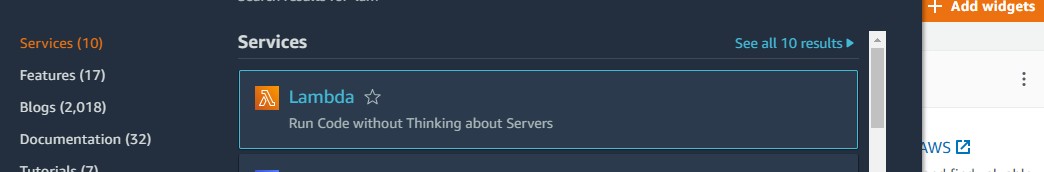
These tasks include traffic management, authorization and access control, monitoring, and API version management.

API Gateway acts as a "front door" for applications to access data, business logic, or functionality from your backend services, such as workloads running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS Lambda, any web application, or real-time communication applications.

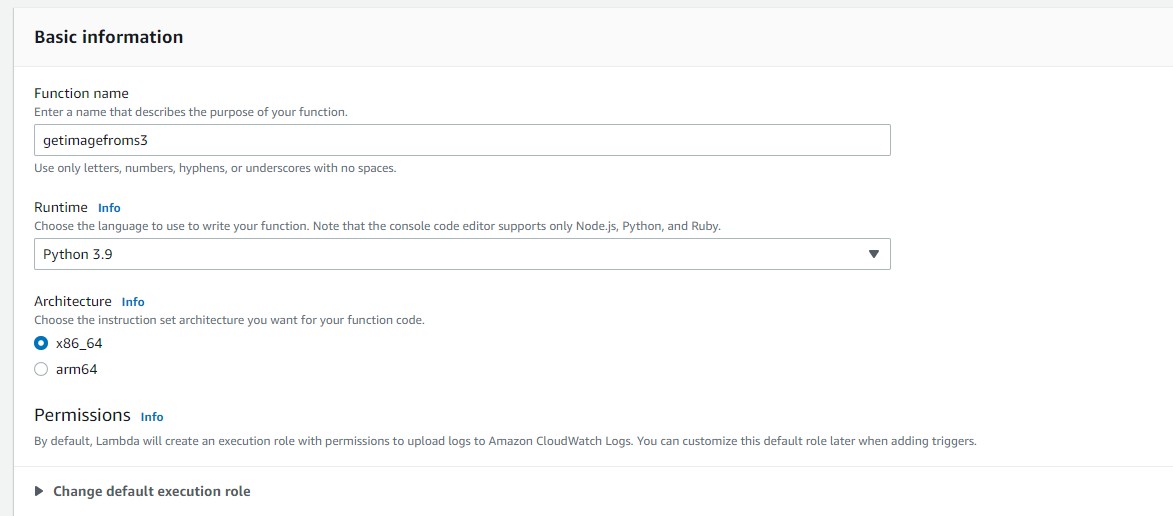
**HANDS ON**

**Steps 1:** In order to extract file from S3 through API you must create bucket and has to have file in that bucket.

**Steps 2:** After completing step 1. Now search for lambda in management console and click it. Below Interface will open. Now Click on **Create Function.**



**Steps 3:** Provide basic information. Choose python 3 as we will be writing lambda code in python. Choose all things as default and click **Create function.**

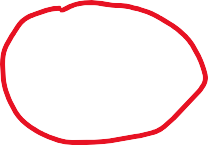
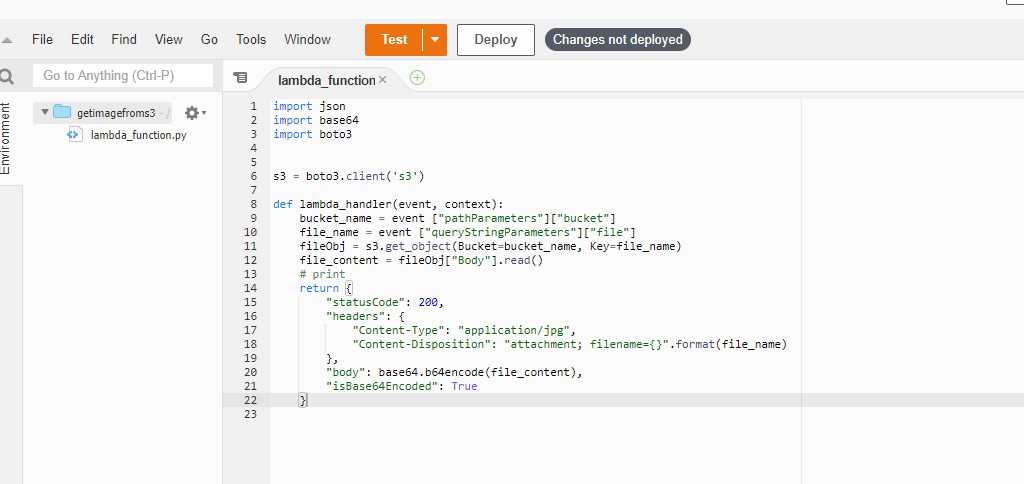


**Steps 4:** Provide basic information. Choose python 3 as we will be writing lambda code in python. Choose all things as default and click **Create function.**

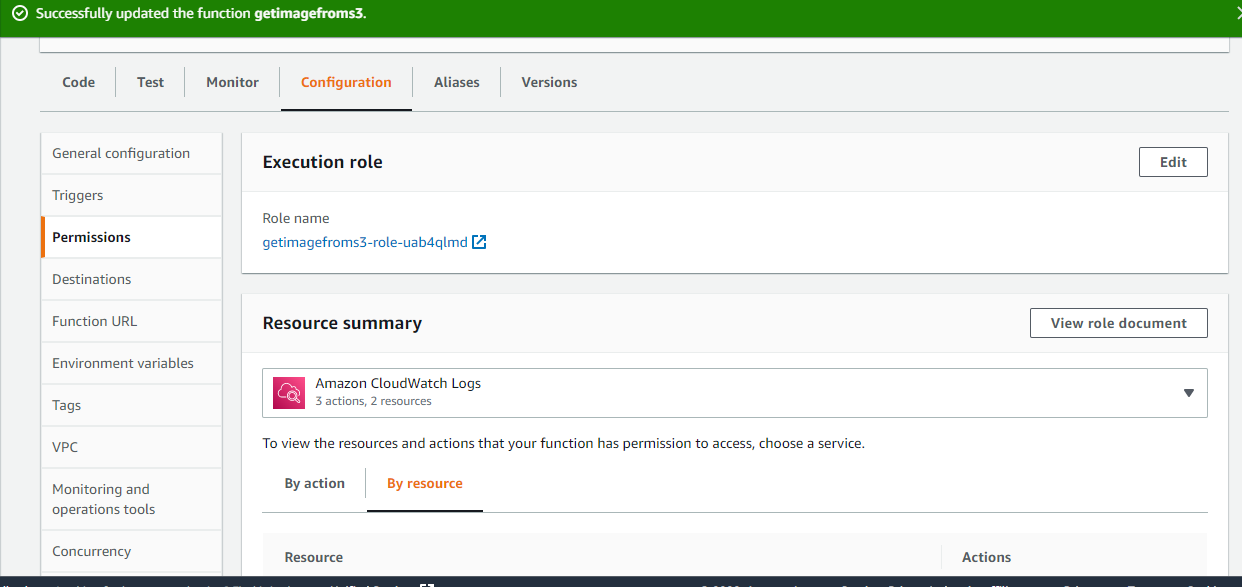
**Steps 5:** Once function is Create write the below code

Or we can get the code by clicking in this link. [Lambda/get\_picture\_from\_s3.py at master · saha- rajdeep/Lambda · GitHub](https://github.com/saha-rajdeep/Lambda/blob/master/get_picture_from_s3.py)

After writing the code click on Deploy. To save the changes made

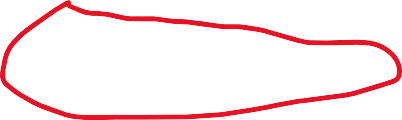
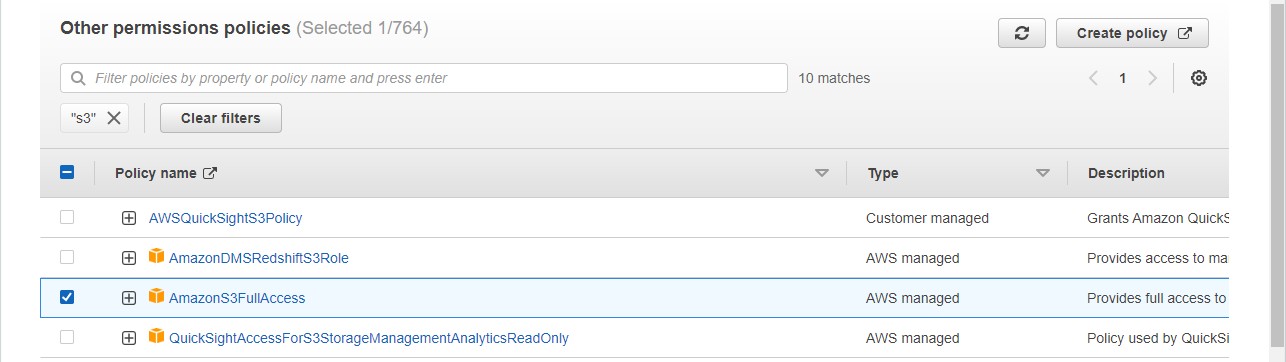
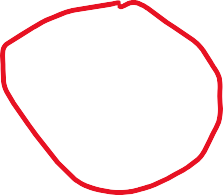
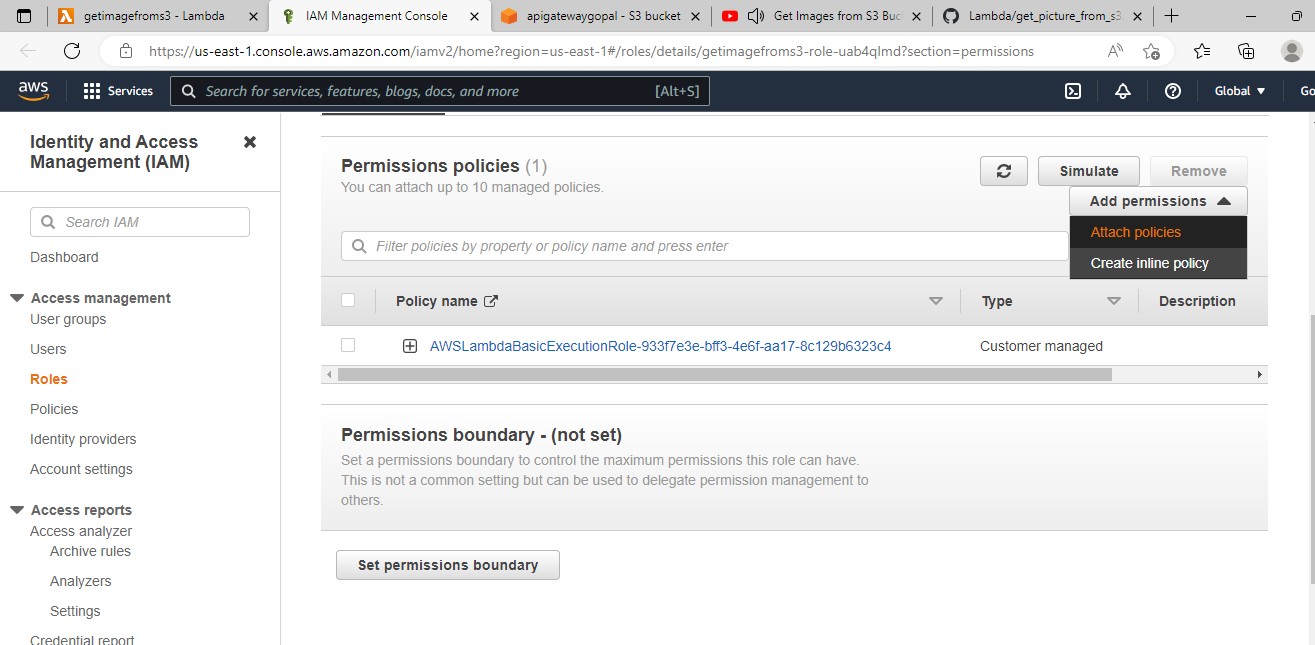


**Steps 6:** Now we have to add permission to lambda function so that it can see files in S3 bucket.So click on **Configuration** then **permissions** then click on the link under **role name**



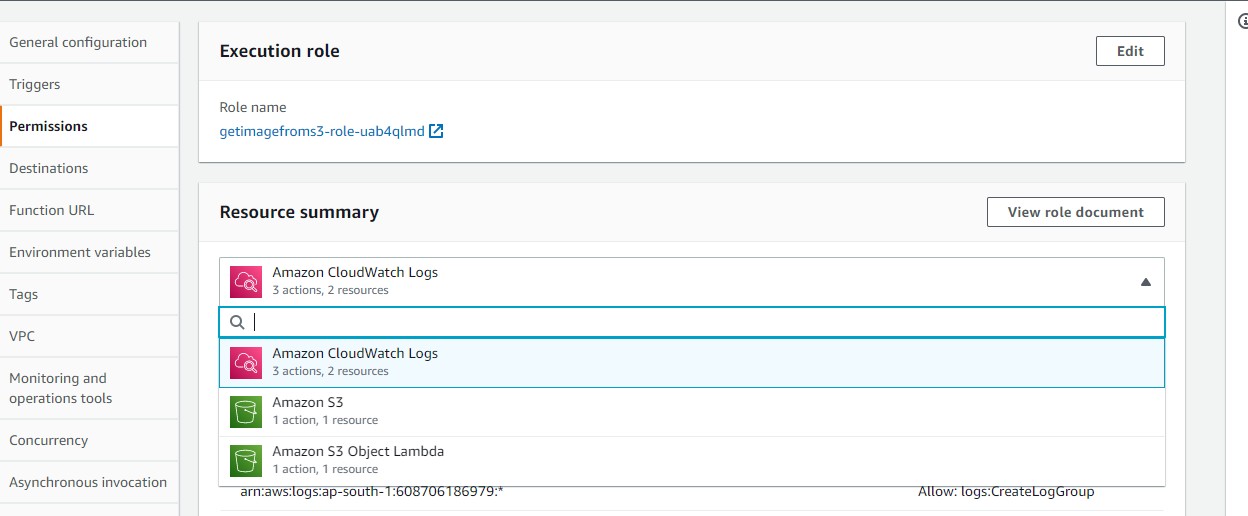
**Steps 7:** Click on **Add permission** and **then attach**

**policies,**search for S3 and tick **s3fullaccess.**

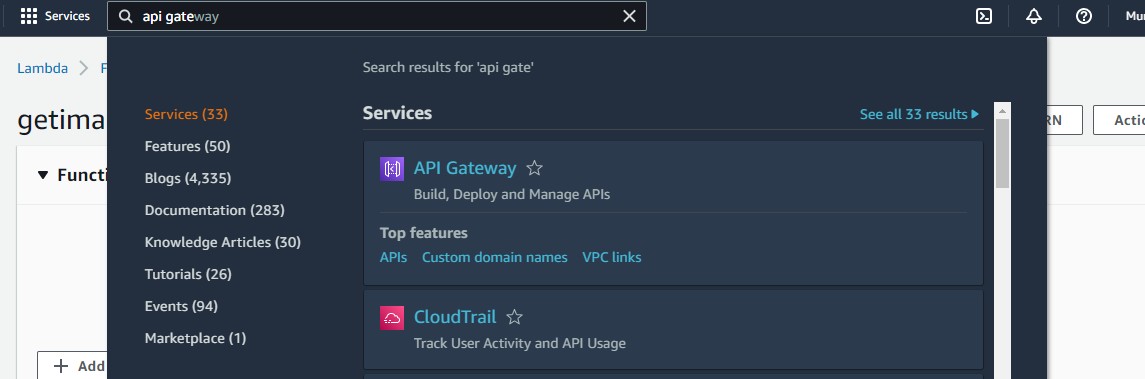


**Steps 8: O**nce it is attached you can see it in resource

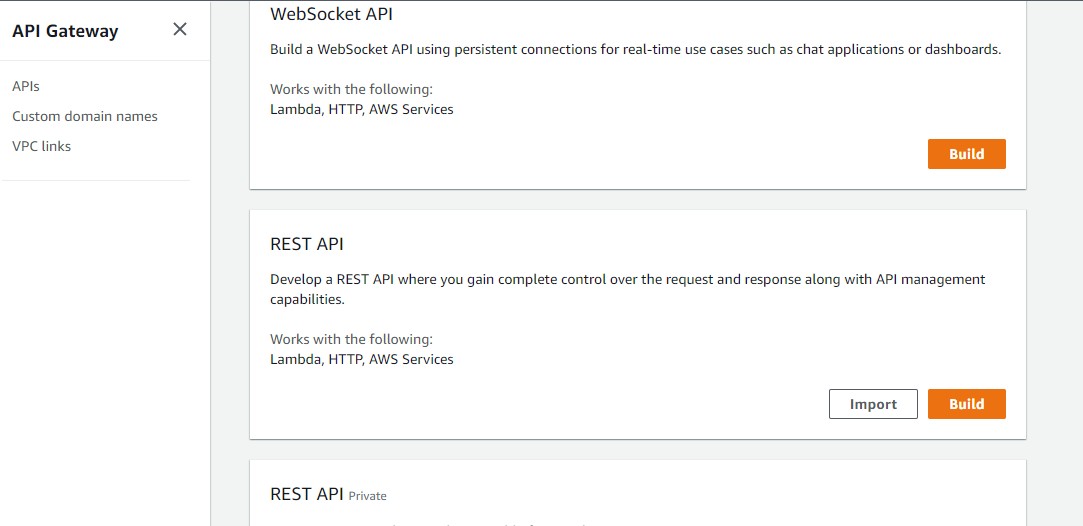
summary



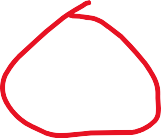
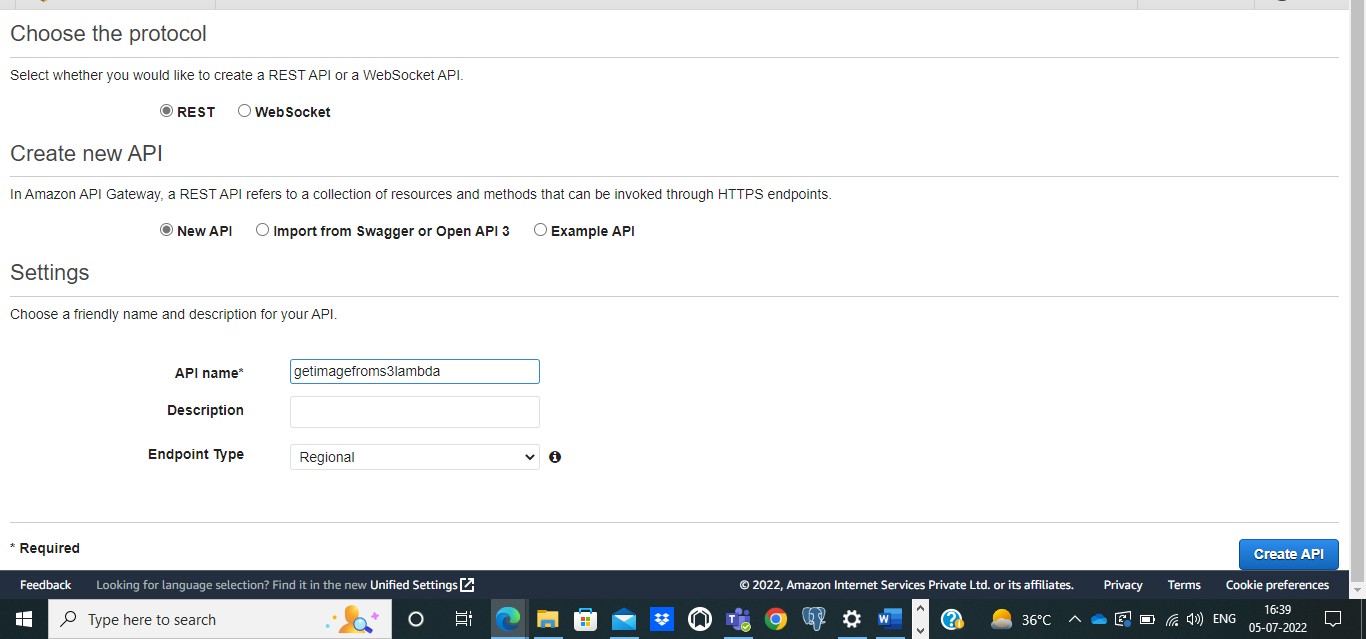
**Steps 9:** Now we have to create API .search Api gateway in management console.



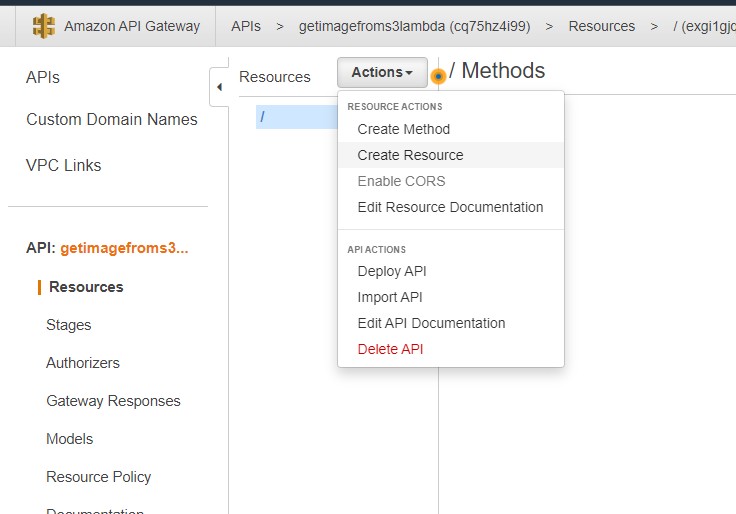
**Steps 10:** select **rest api** and click on **build.**



**Steps 11:** Click on New API and give api name then click on create api

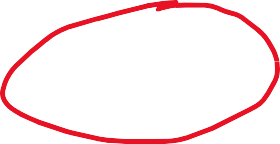
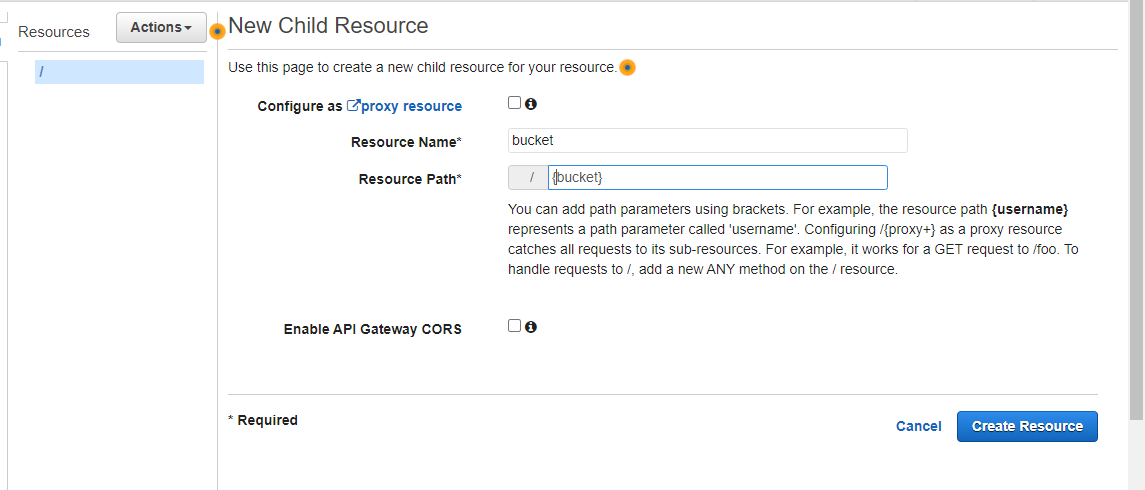


**Steps 12:** Click on Action then create resources



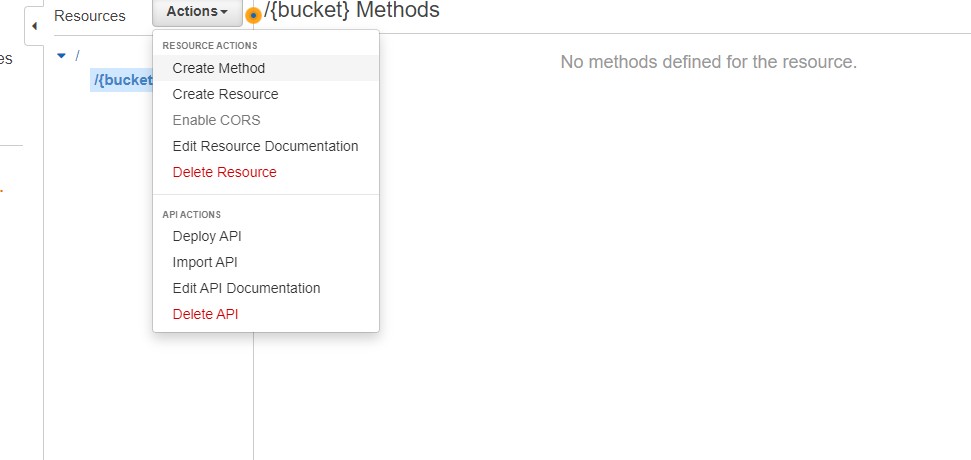
**Steps 13:** Give Resource as **bucket** and resource path as

# {bucket}

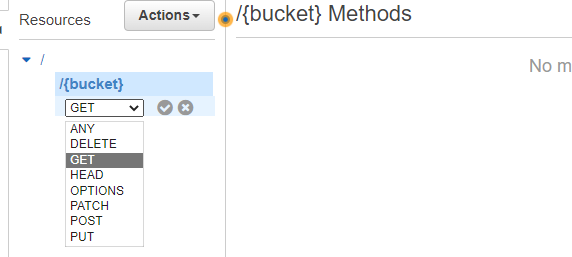


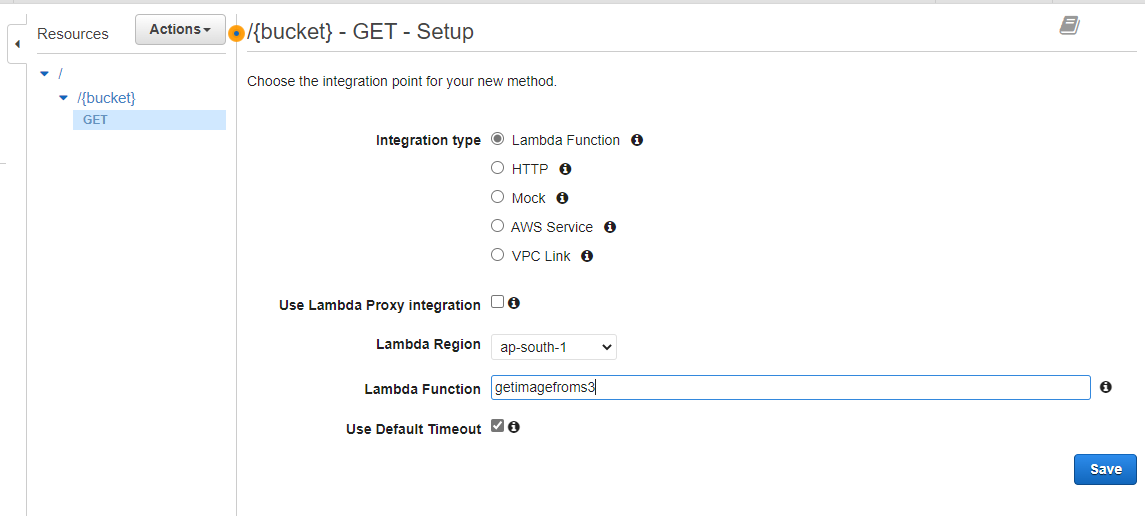
**Steps 14:** Once resource it create again click on Action and

select CREATE METHOD

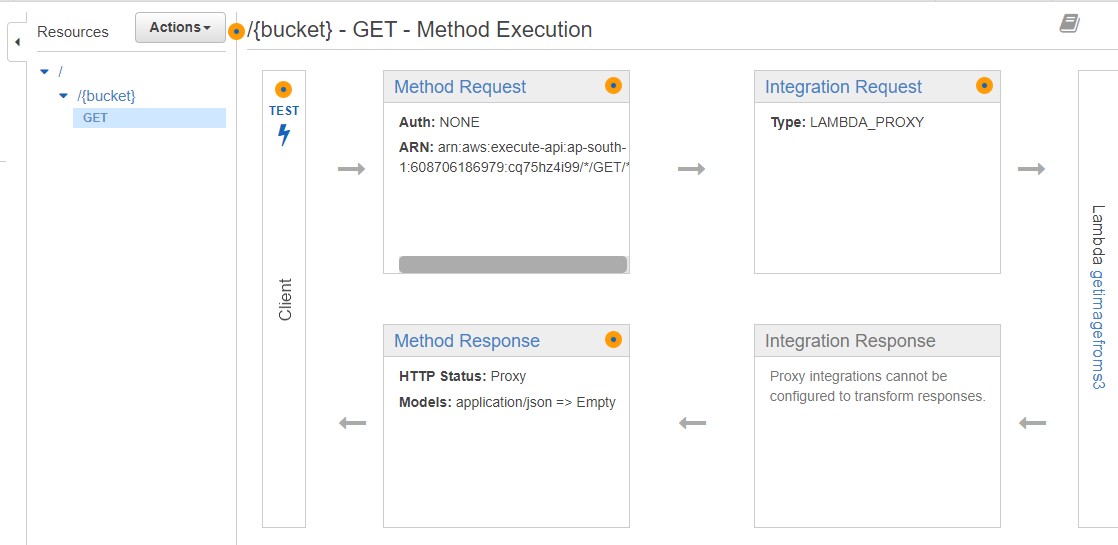


**Steps 15:** Select **get method** tick the check mark

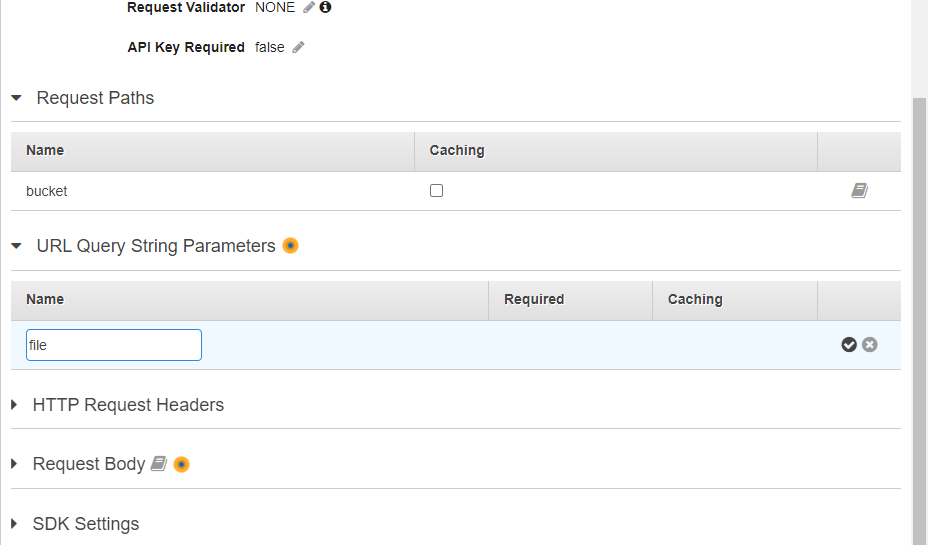


**Steps 16:** Give the name of your lambda function and then save

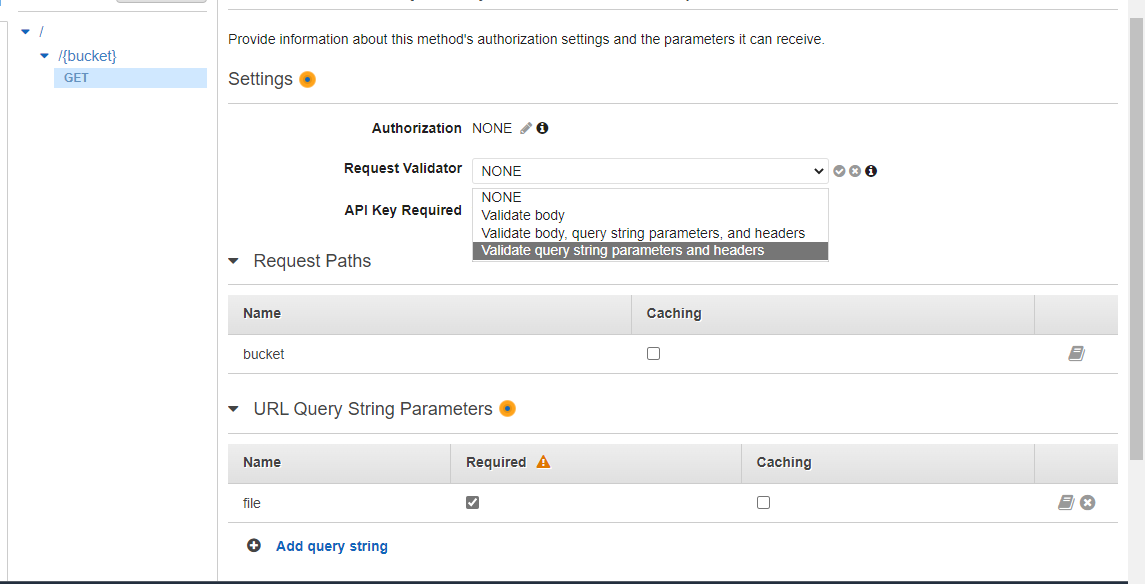
**Steps 17:** Now click on Method Request



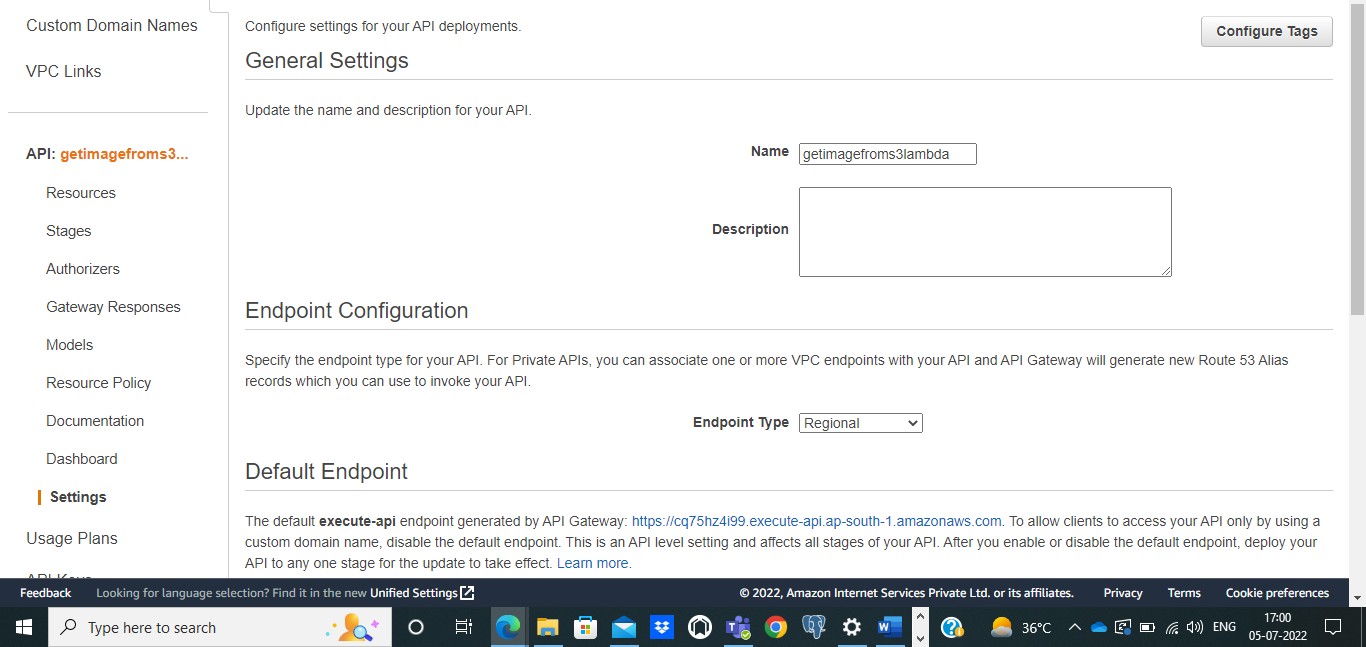
**Steps 18:** Now click on **add query string** and give the name as **file** then check the tick mark.



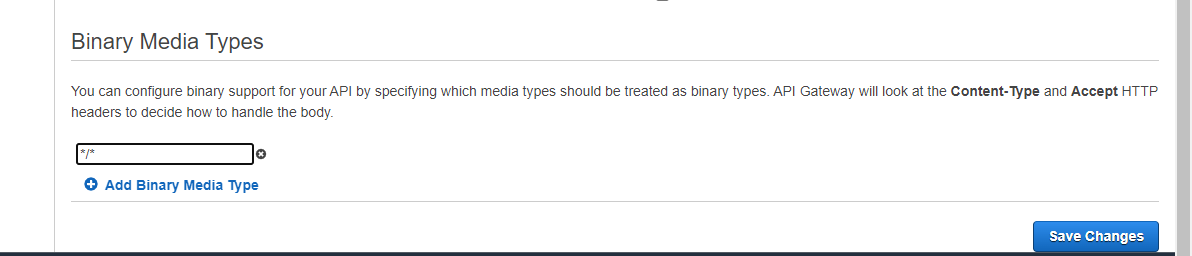
**Steps 19:** change request validator from **none** to **validate query string and headers**

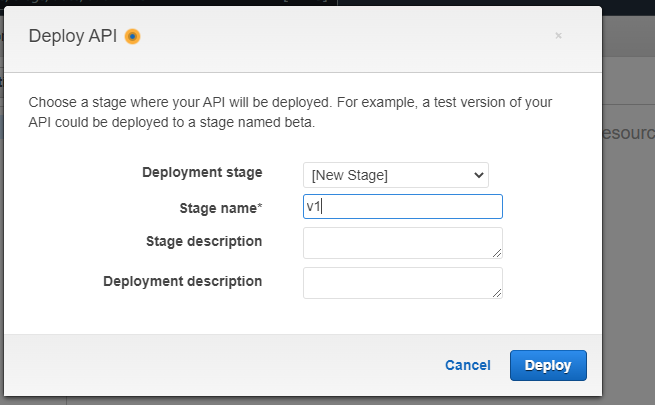
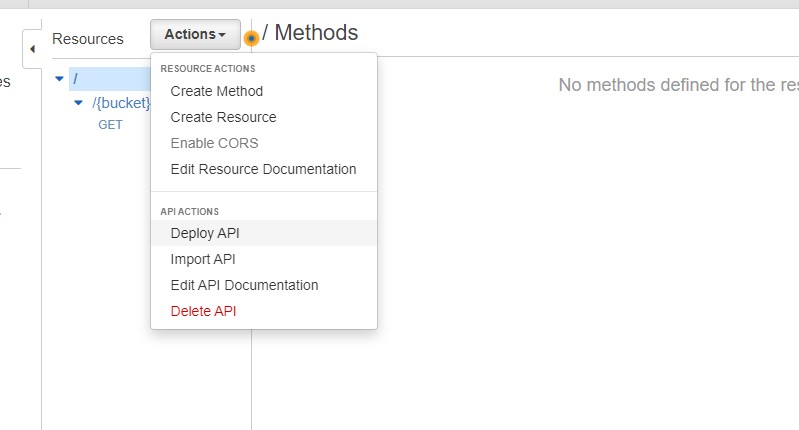


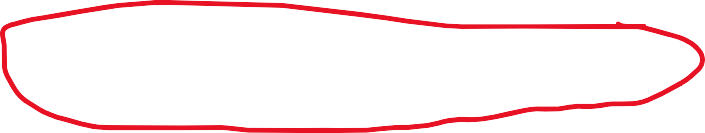
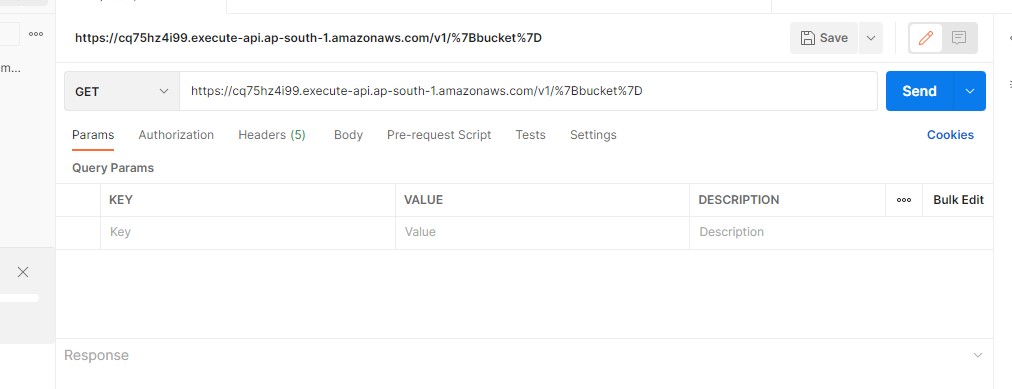
**Steps 20:** After that go the **setting under api**.



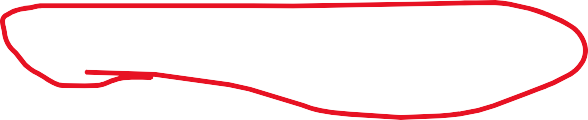
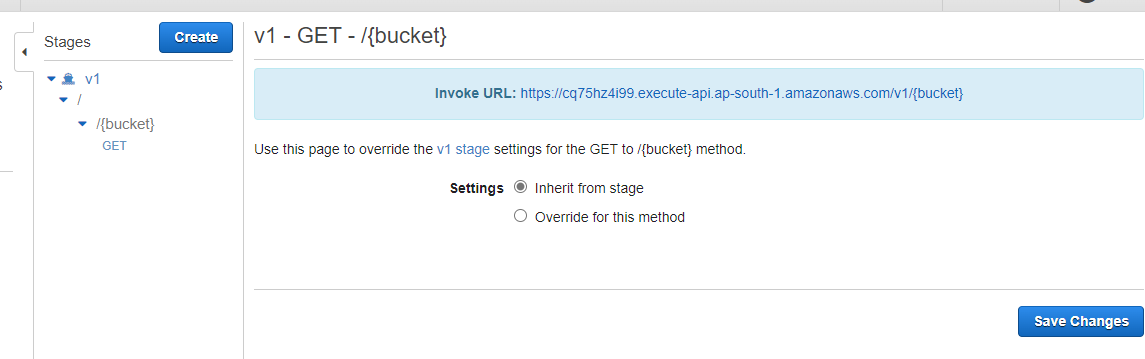
**Steps 21:** File the binary media types as **\*/\*** and save changes.



**Steps 22:** Now we have to deploy our api. For that click on **action** then click on **Deploy API** then select **new stage** and give stage name then **deploy**

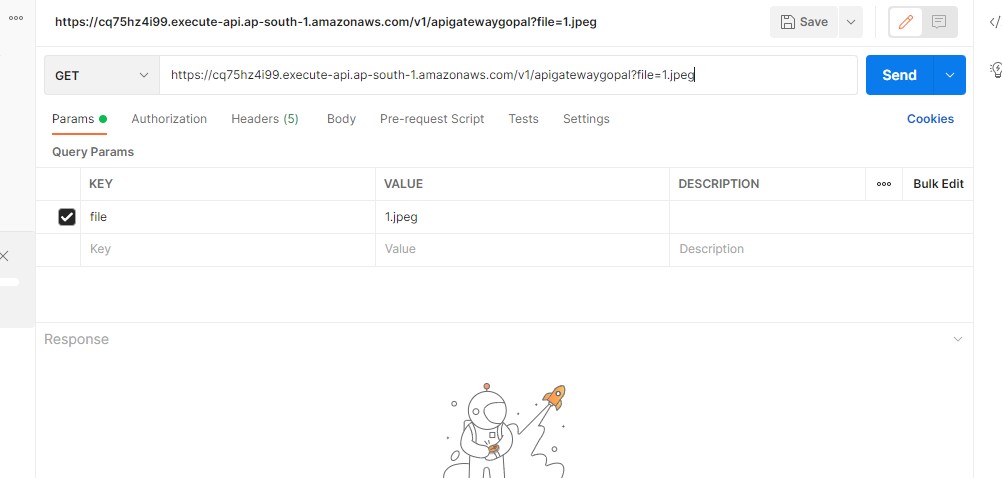


**Steps 23:** in oder to get api click on **get** then copy the URL.



**Steps 24:** Now open your postman copy this URL on that.

**Steps 25:** Now make some modification in your URL. After V1/ give your bucket name followed by ?file=yourfilename. **Then click on send.**



**Steps 26:** once you click on click image file will optain as binary format as shown below, you can save that file in you system by clicking on save references.

