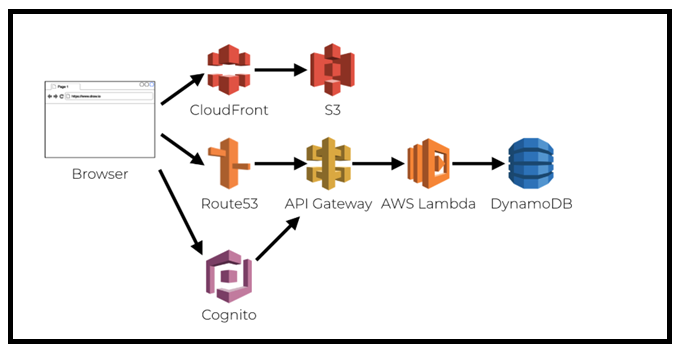
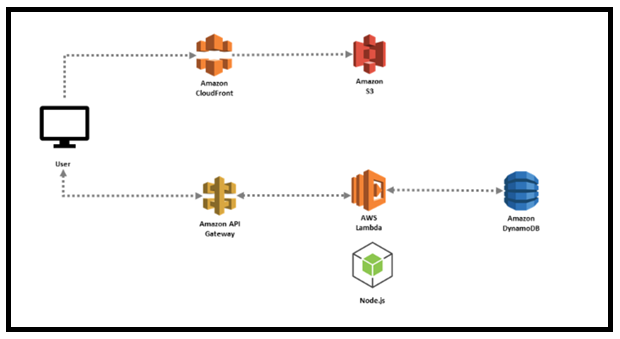
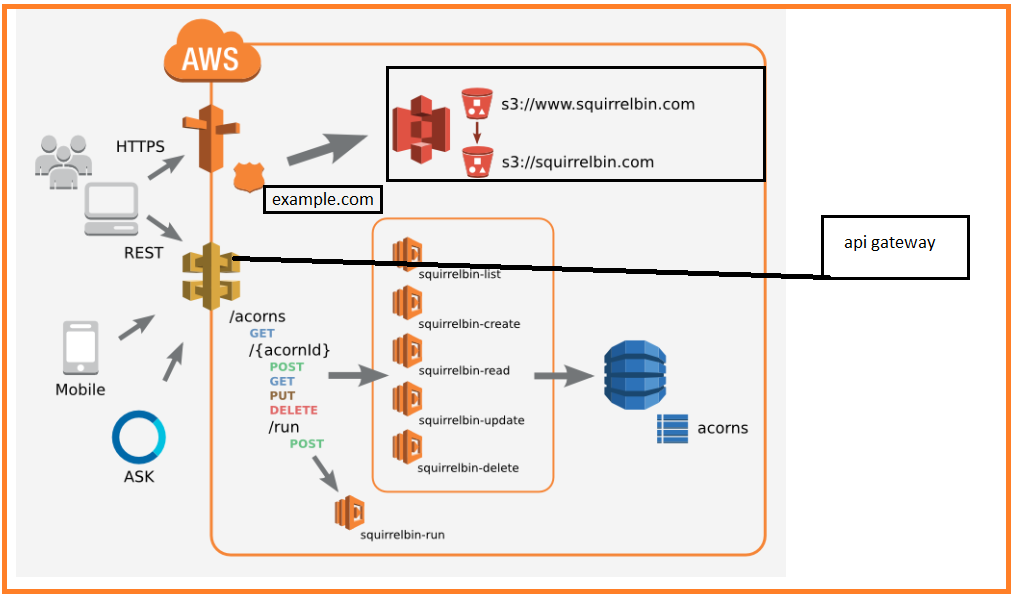
SERVERLESS  
1. History of Cloud computing.  
  
1.1 Data Center : First we moved from a PC to Data Center. We don’t have to be bothered about switching it on and off , n/wing etc. This still takes time to place order and get the hardware, configure the n/w, install OS etc.  
  
1.2 IAAS : Amazon released EC2. This was the birth of infrastructure as a service. We could provision machines using API calls. Aws is basically iaac, and you can commission a machine anywhere in the world with an API call.  
We still have to worry about infrastructure, still it could crash/hacked/goes offline.  
  
1.3 PAAS : Then Amazon’s version of paas is called Elastic Bean Stalk was released. It’s a way of saying, give me your code and we will provision the machine for you and push your code into it and run it for you.  
In this case we did not have to worry about the architecture. We still have the server that we needed to control.

1.4 Containers : Containers are not related to cloud but are popular in cloud environments. Containers are isolated and lightweight, But they still are needed to be deployed to a server. They are an excellent solution when dependencies are in play. We still have to worry about keeping our containers running and scaling.

1.5 Lambda : Birth of server less computing. Basically lambda allows you to take your code and run it with out any need to provision servers/ install softwares / deploy containers/ and no need to worry about any low level details.  
And your code can be run in massive parallelized way in response to events. The cloud provider takes care of provisioning and management of infrastructure.  
  
2. Sample Introductory Use cases of Server less architecture.  
  
   
  
  
 &&&&&&&  
  
   


3. Lambda – Theory.

What is Lambda : It is a compute service where you can upload your code and create a lambda function.  
Aws lambda takes care of provisioning and managing the servers that you use to run the code. And so we don’t have to worry about Operating systems, patching, scaling etc.

Lambda can be used in the following ways :   
Lambda can be used as an event driven compute service, where aws lambda runs your code in response to events.  
These events could be – ‘changes’ to the data in S3 bucket / or change to your data in the dynamo DB table, these are called triggers.

You can also use Lambda as a compute service to run your code in response to HTTP requests using Amazon API Gateway.

4. Languages supported by Lambda   
  
Node C#  
Java Go  
Python

5. Notes - lambda.  
Lambda Scale OUT not UP : You can have millions of functions running in parallel.  
Lambda functions are independent : 1 event = 1 function.  
Lambda : is a compute service + a serverless service .  
  
What other services are serverless : API Gateway + S3 + Lambda + DynamoDB etc.  
  
Lambda functions can trigger other function.  
And 1 event can trigger many lambda functions.

Aws X Ray : To debug lambda.

Lambda can do things globally : Backup s3 buckets to other s3 buckets. Not confined to one region.

5. API GATEWAY – Theory.  
There are two are kinds of API’s.  
REST APIs : Representational State Transfer.  
SOAP APIs : Simple object Access Protocol. Uses XML.

What is api gateway : API gateway is a **fully managed** (serverless) service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale.

API gateway exposes http endpoints to define a restful api. And it gives us a https address that we can make calls to and we can configure how to respond to those calls.

It basically helps you to serverlessly connect to services like lambda or DynamoDB. You don’t have to worry about any EC2 instances.

And we can **send one end point to** a lambda **and other endpoint** to Dynamo DB or EC2 etc...

Api gateway **scales effortlessly**- don’t have to worry about autoscaling groups etc.

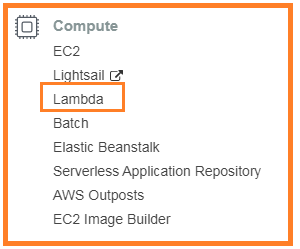
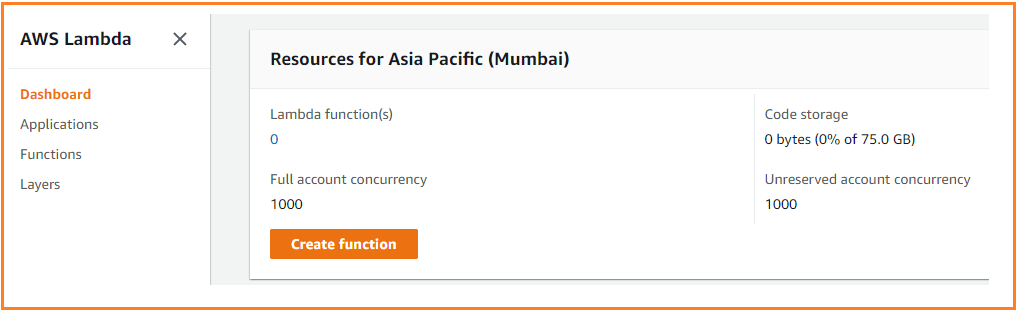
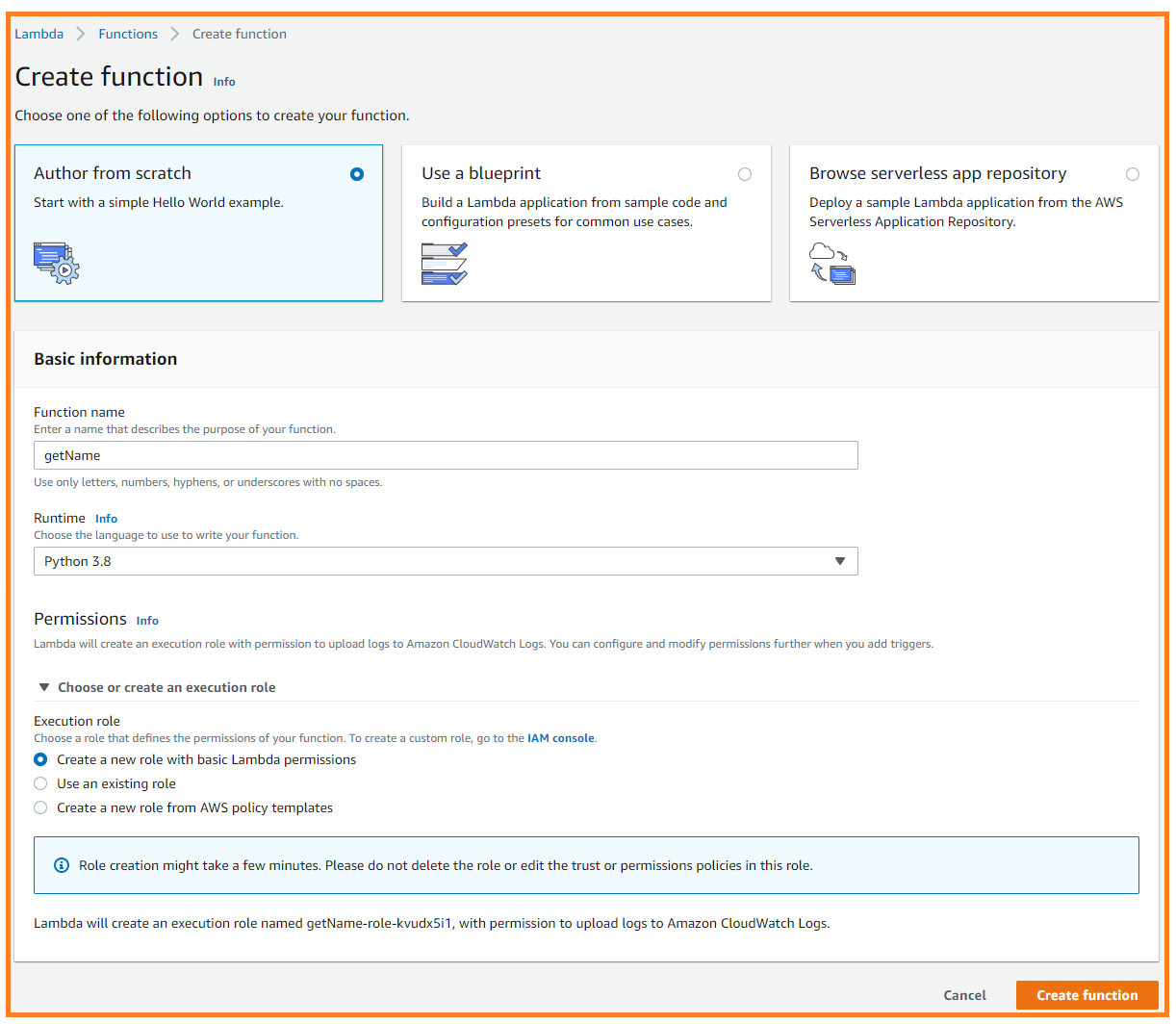
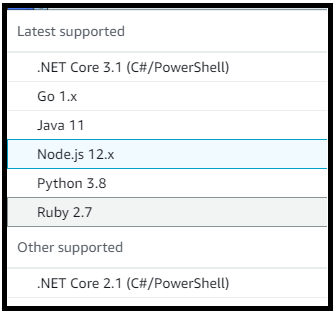
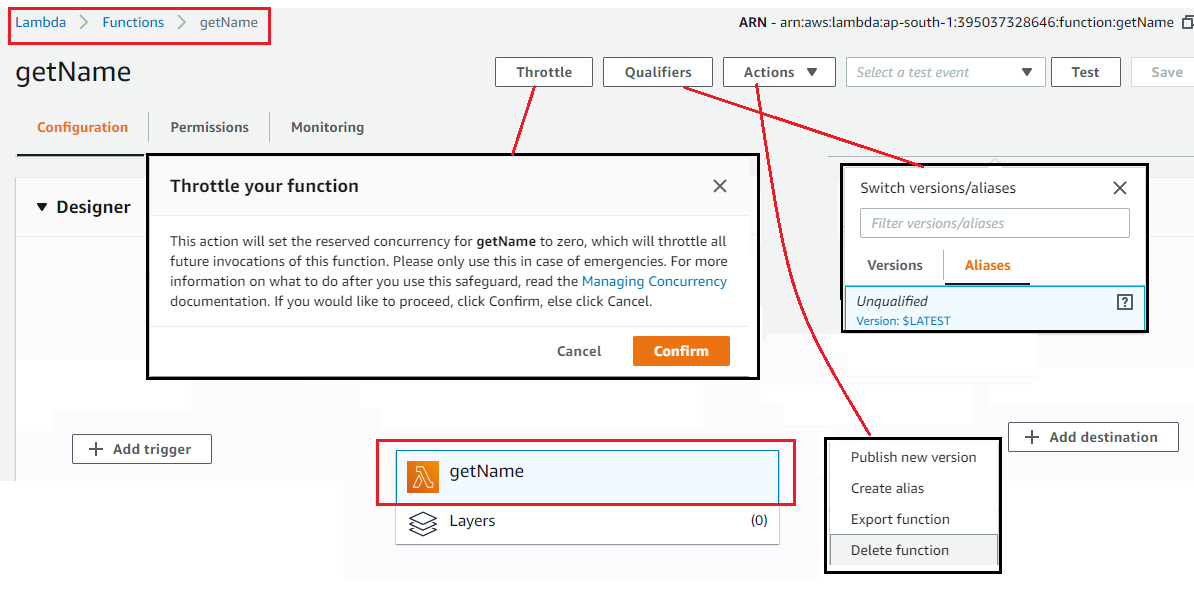
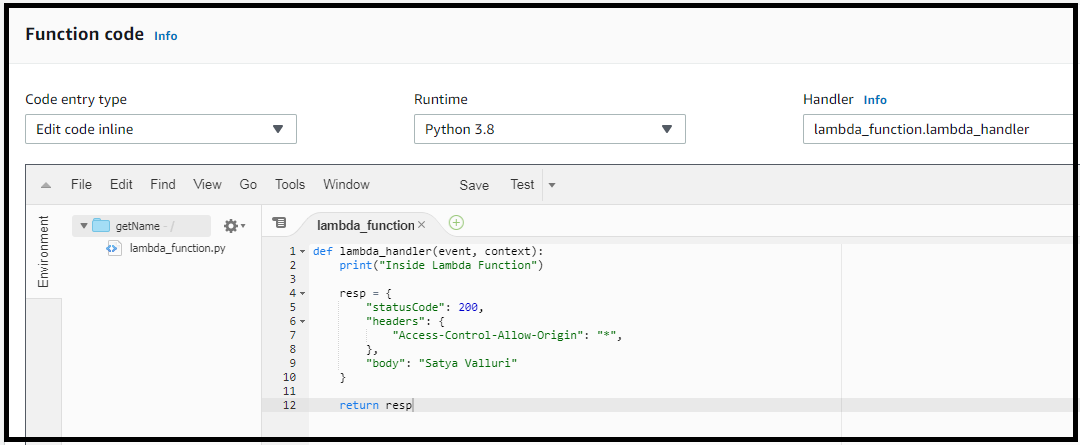
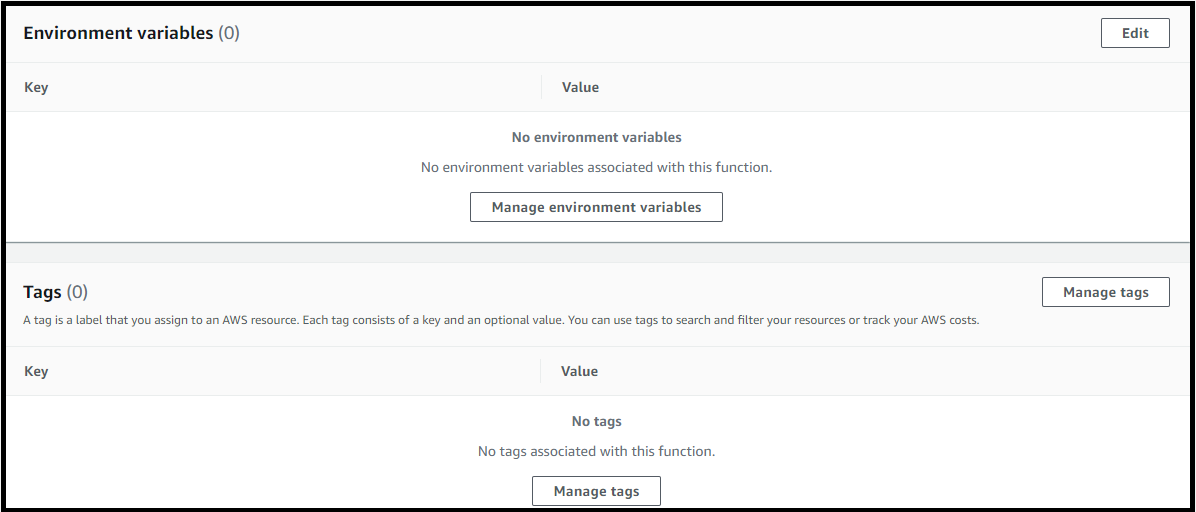
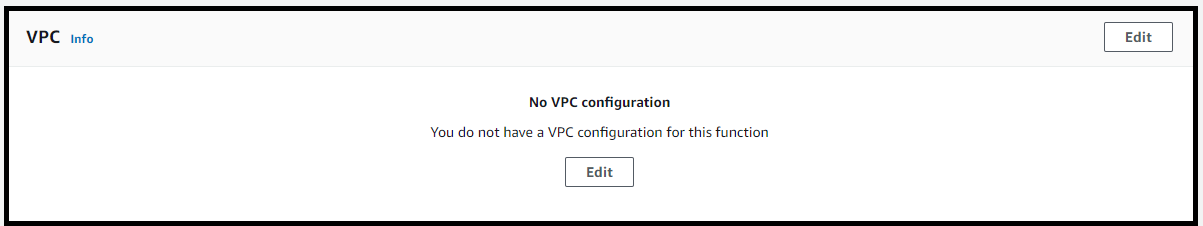
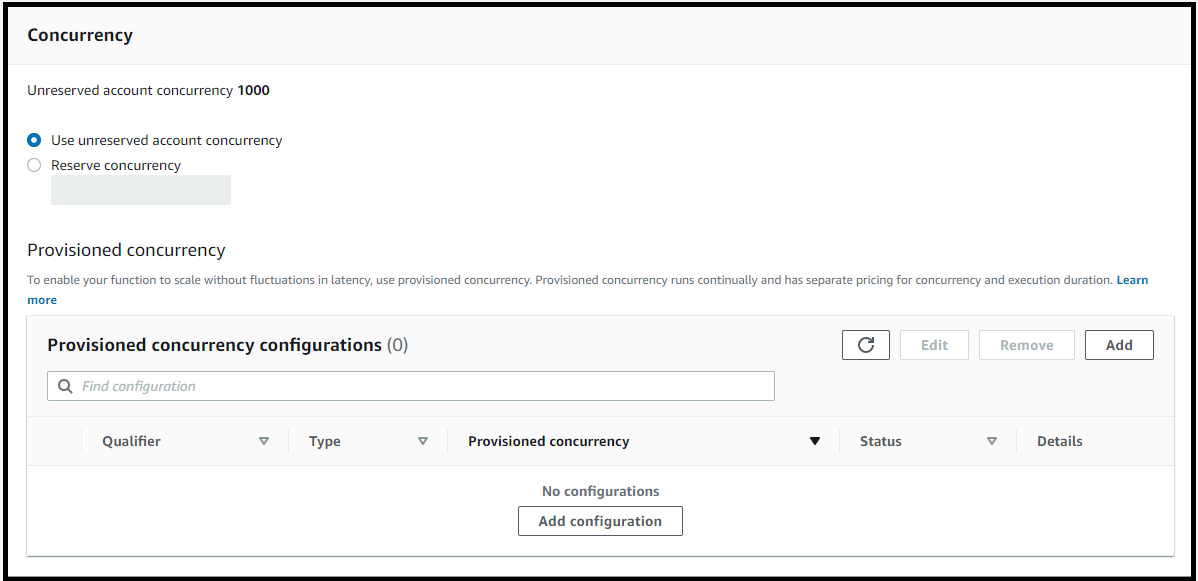
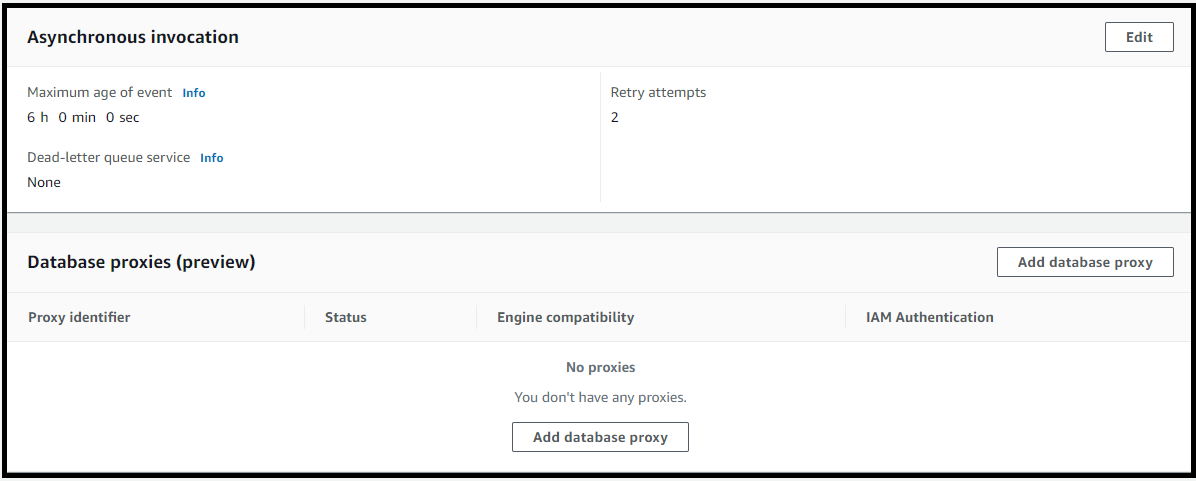
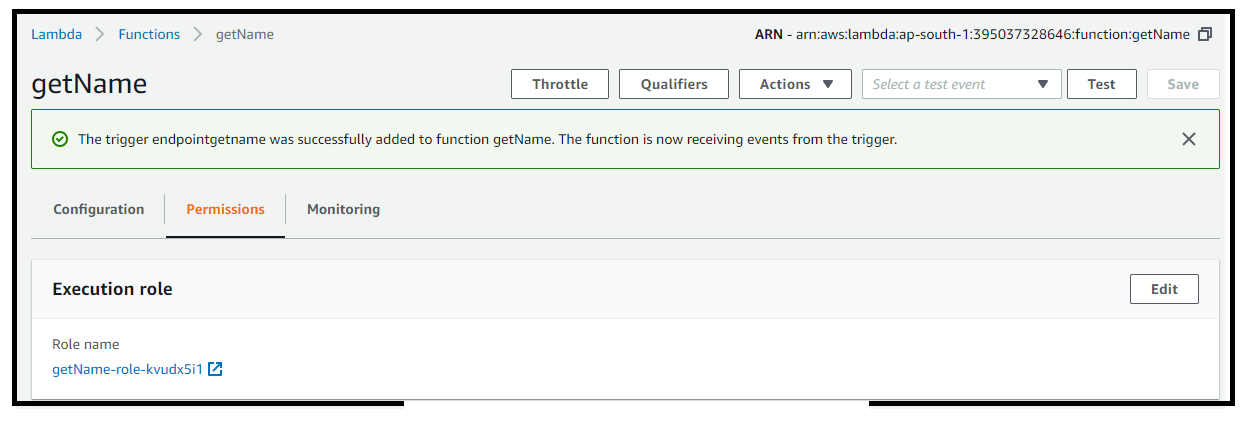
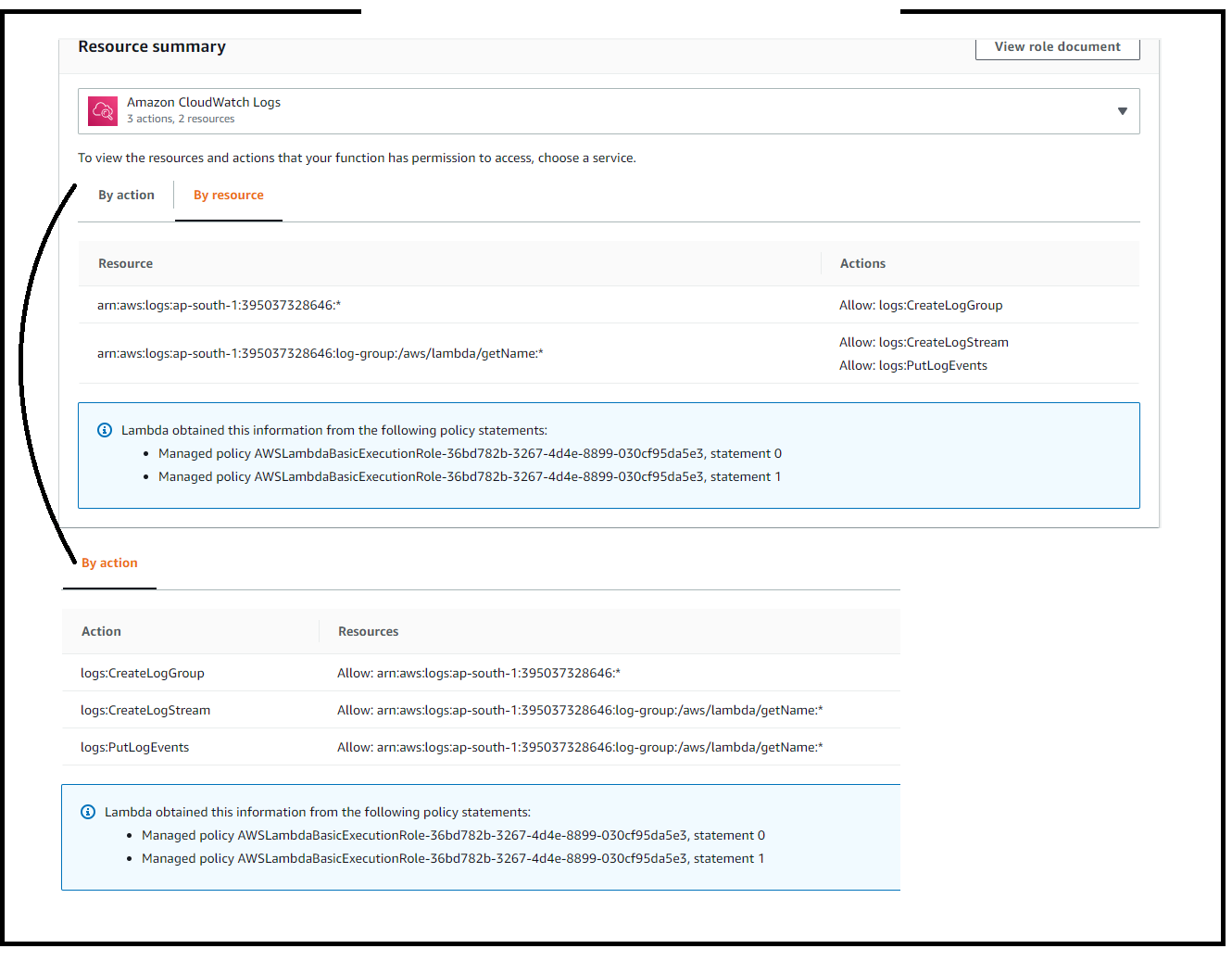
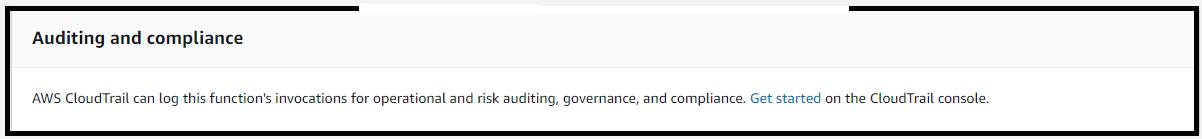
We can **throttle** requests to prevent attacks.

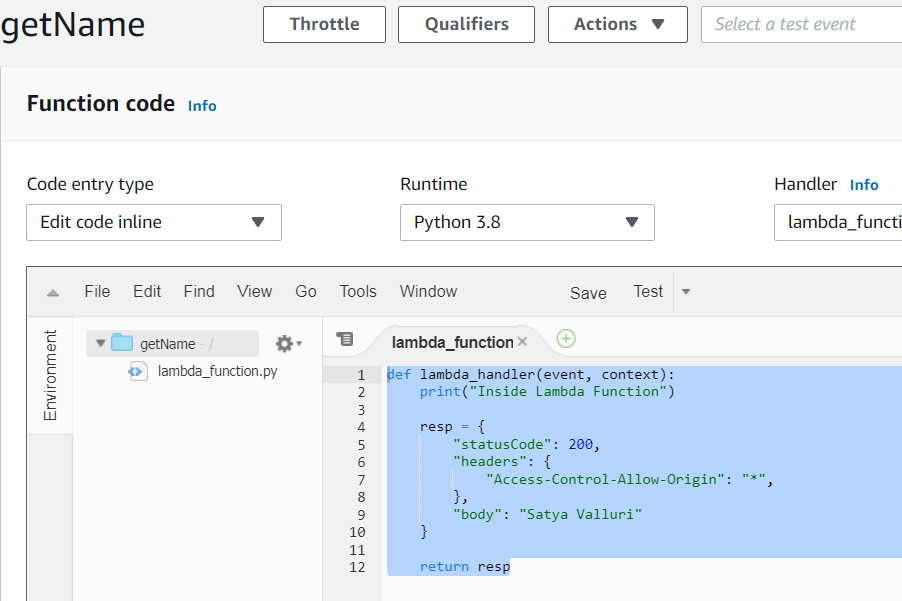
We can connect to **CloudWatch to log all requests for monitoring**.

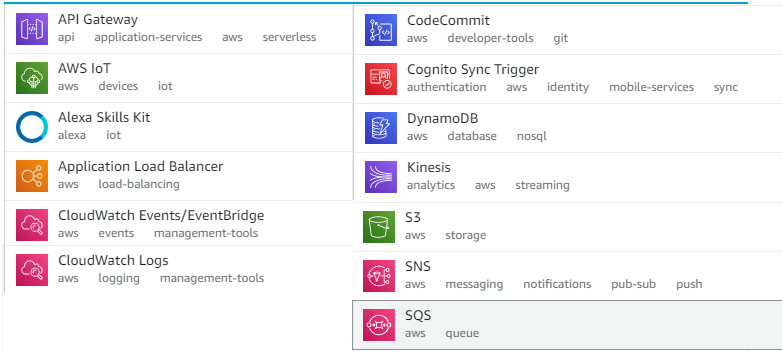
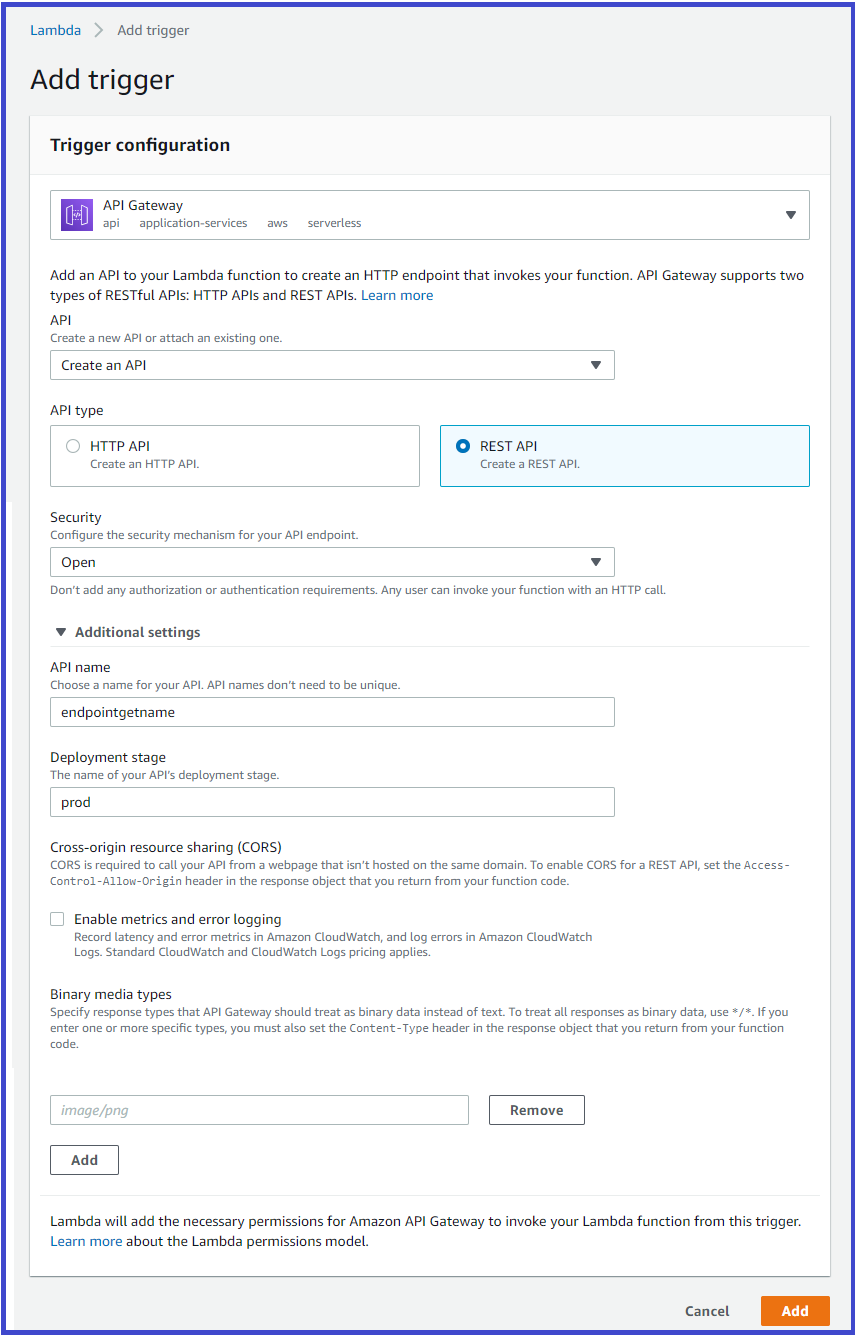
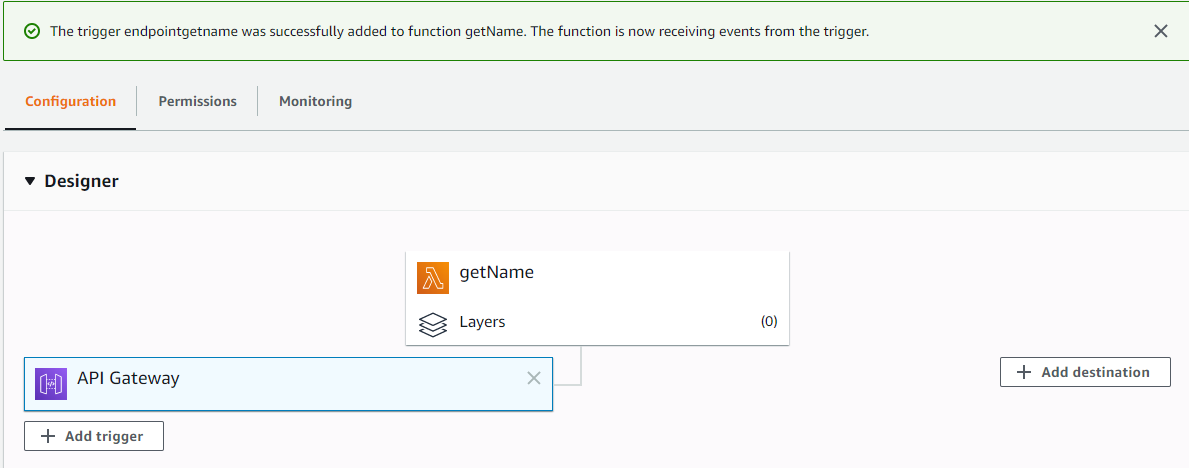
We can also maintain multiple versions of our API.

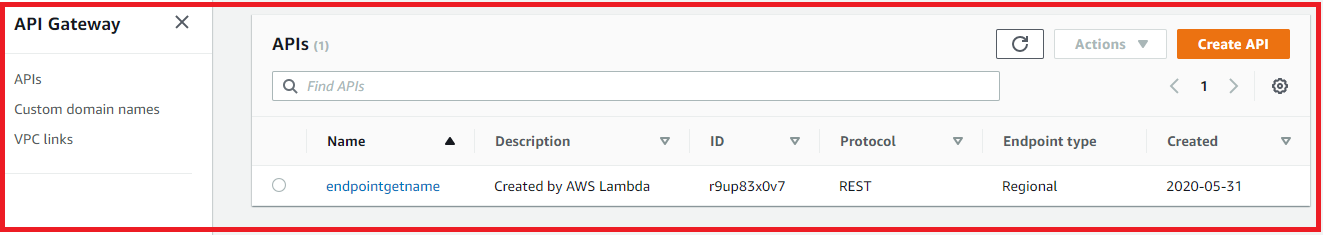
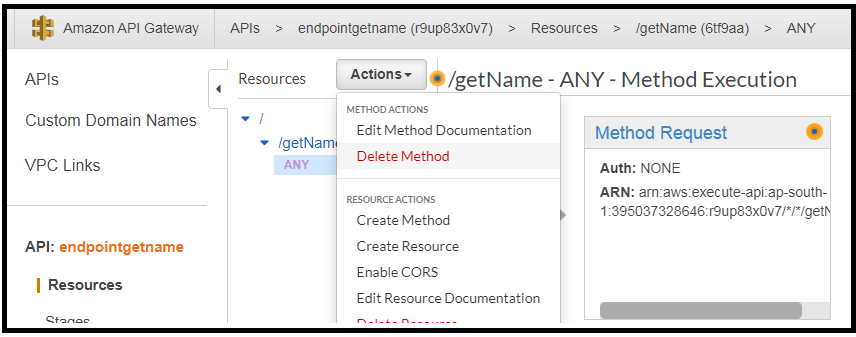
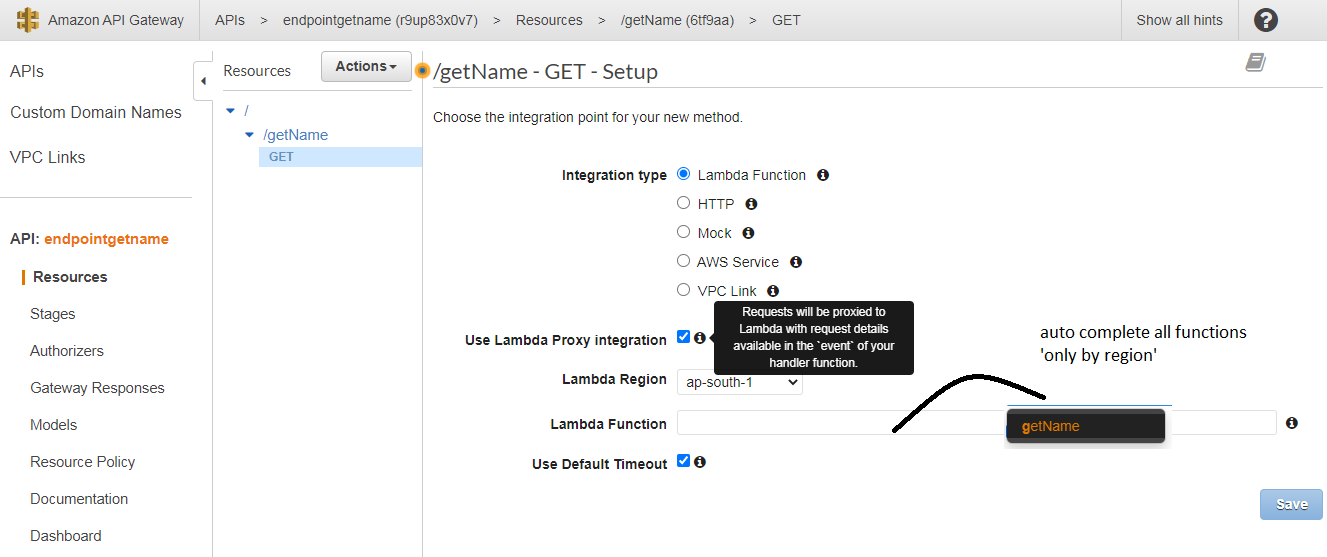
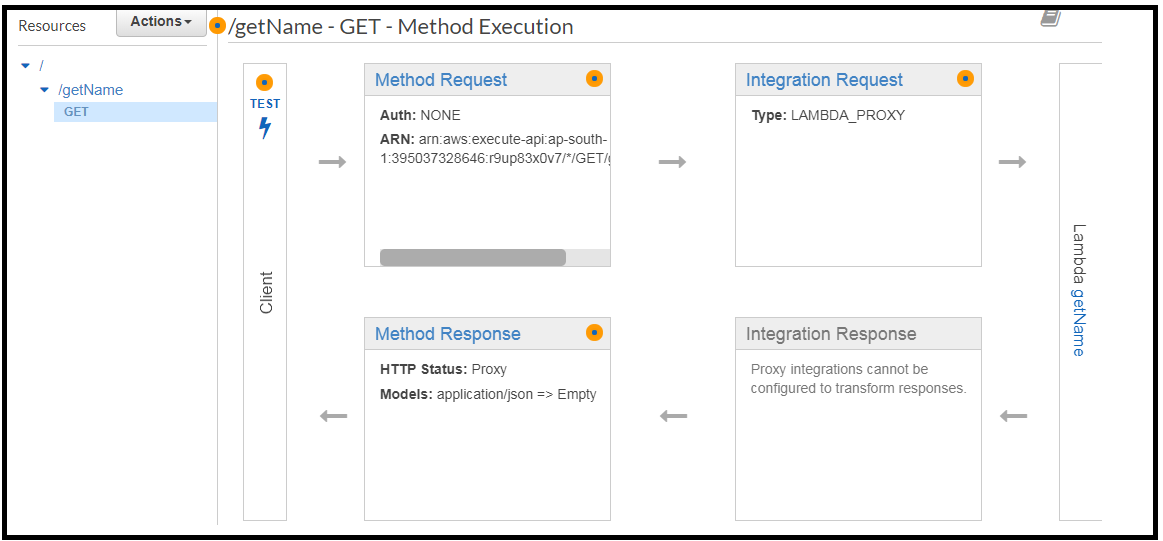
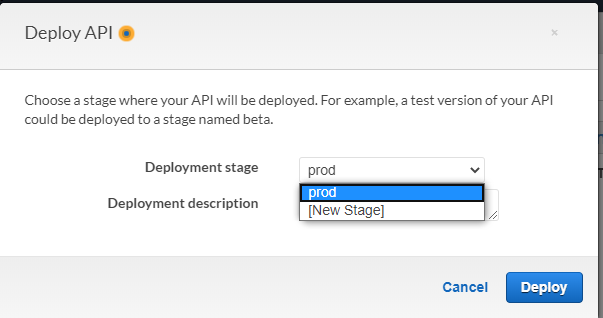
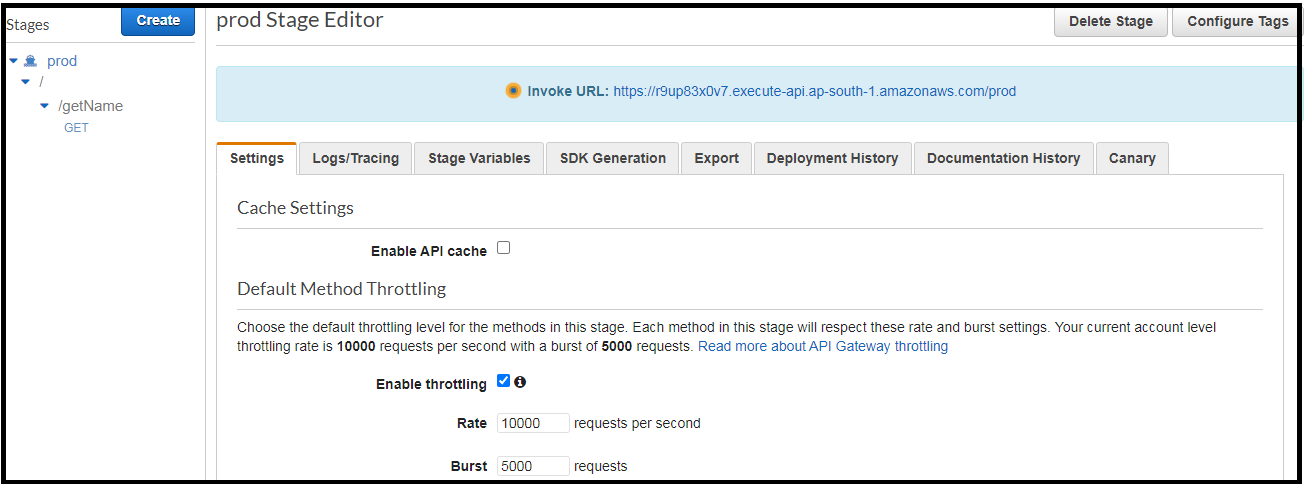
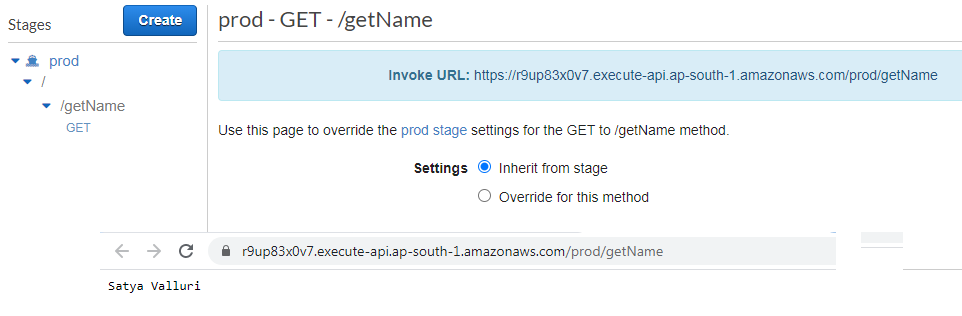
Apigateway uses its own domain by default, but we can use our own domain.

Apigateway supports aws certificate managers.

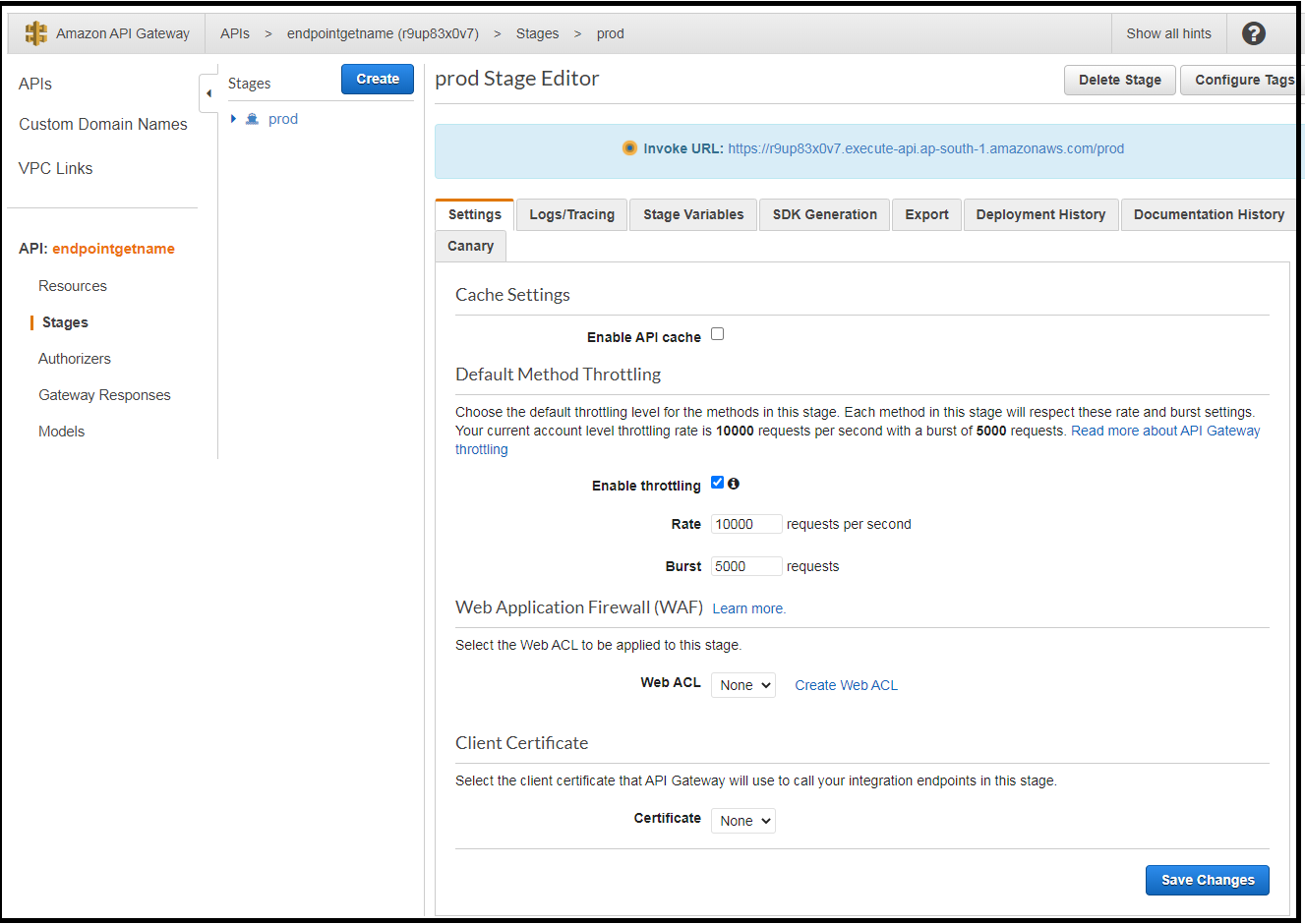
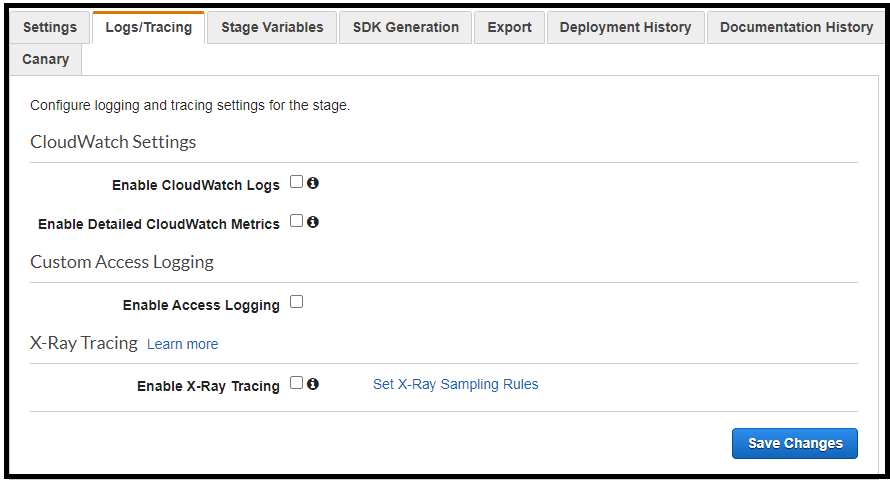
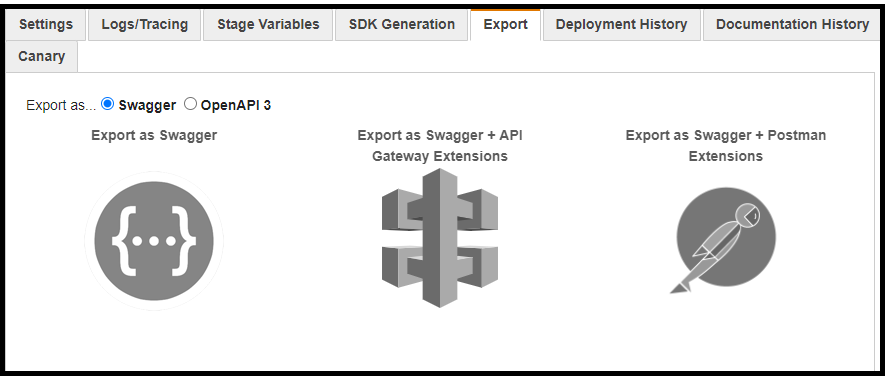
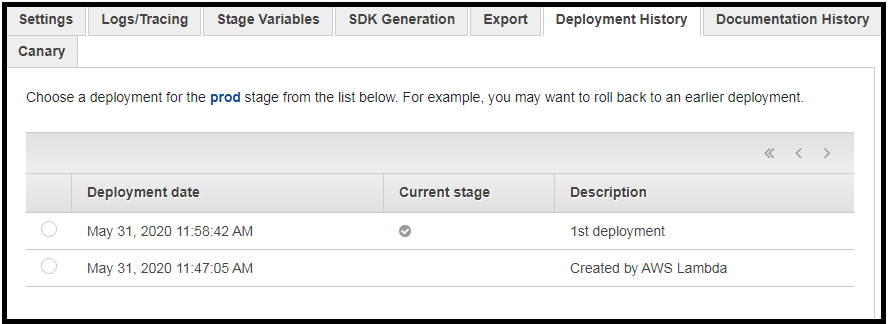
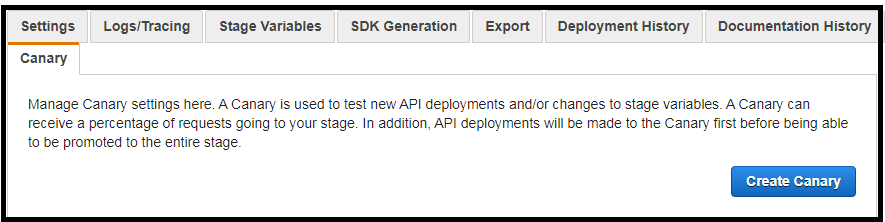
6. Api Gateway – Lambda – S3Example 1.  
In this example we create a lambda function which will be triggered by a api gateway. This api gateway will be called from a html file hosted in a S3 bucket.  
  
 **6.1** Lets create a lambda function.  
  
   
  
The AWS Lambda dashboard looks like below.  
  
  
  
Create a function . A role could be automatically created with permissions to upload logs to cloud watch.  
  
  
  
  
**6.2** What does dashboard looks like after creating the function.  
  
 Configuration Tab /Designer section  
  
  
Configuration Tab / function Code section  
  
  
Configuration Tab / **Environmental Variables** and **Tags** section.  
  
  
Configuration Tab / **VPC** section.  
  
  
  
Configuration Tab / **Concurrency** section.  
  
  
  
Configuration Tab / **Asynchronous invocation** and **Database proxies** section.  
  
  
  
Permission Tab / Execution Role.  
  
Permission Tab / Resource Summary.  
  
  


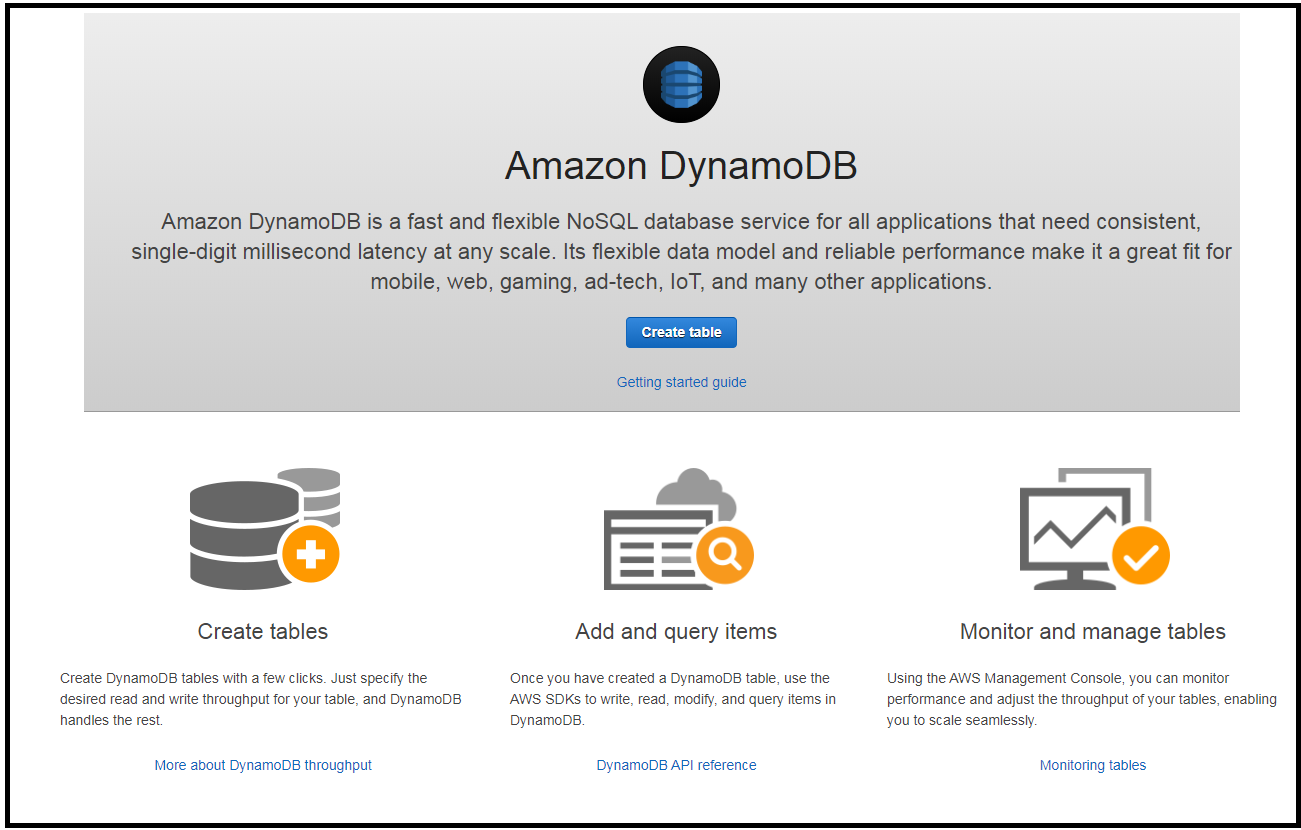
6.3 Add code to the lambda function and save it.  


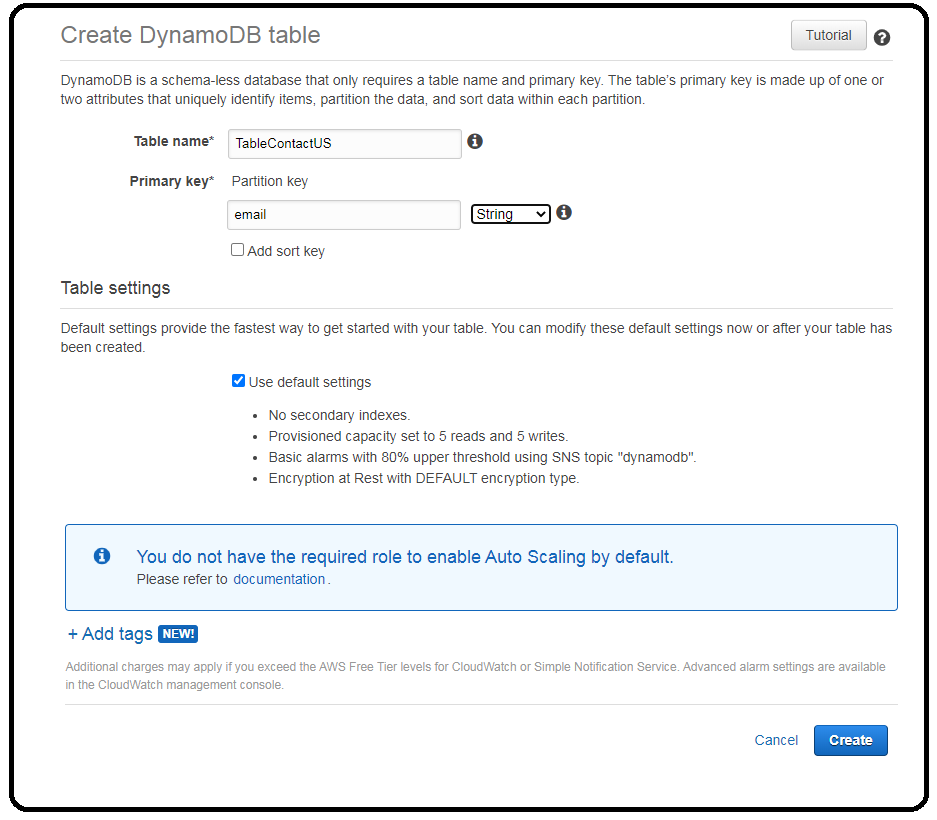
6.5 Add an Api Gateway trigger to the code.  
  
Below are the list of acceptable triggers.  
  
  
  
  
  
  
6.6 After adding the trigger we see as below.  
  
  


6.7   
Configure the API gateway.  
  
  
The end point supports all http methods by default, Lets add only a GET method to the end point.  
  
  
  
  
  
After creating the GET method we get the below view.  
  
  
  
Deploy the api.  
  
  
After deploying the api.  
  
  
We get the invocation URL.  
  


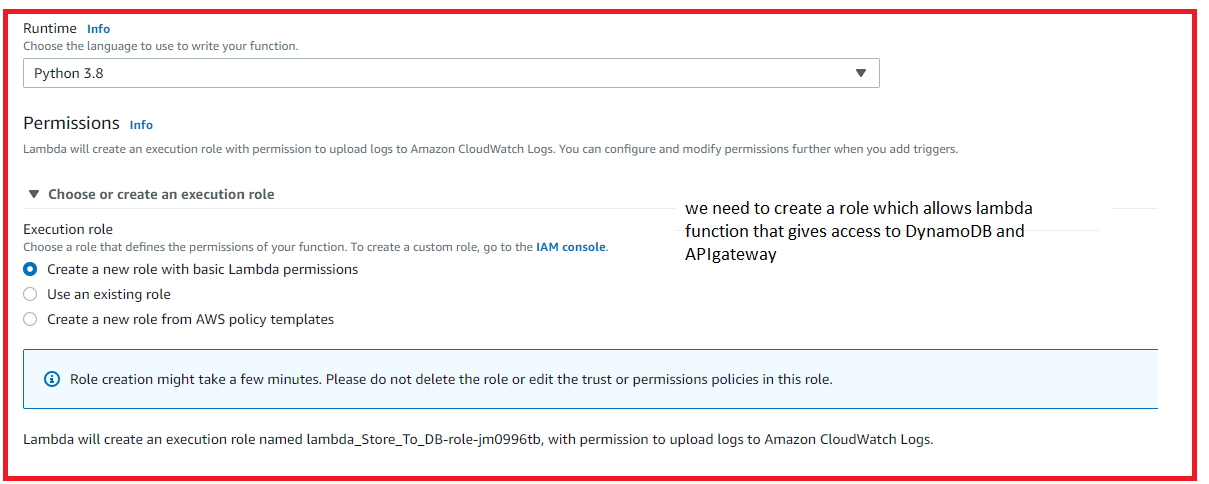
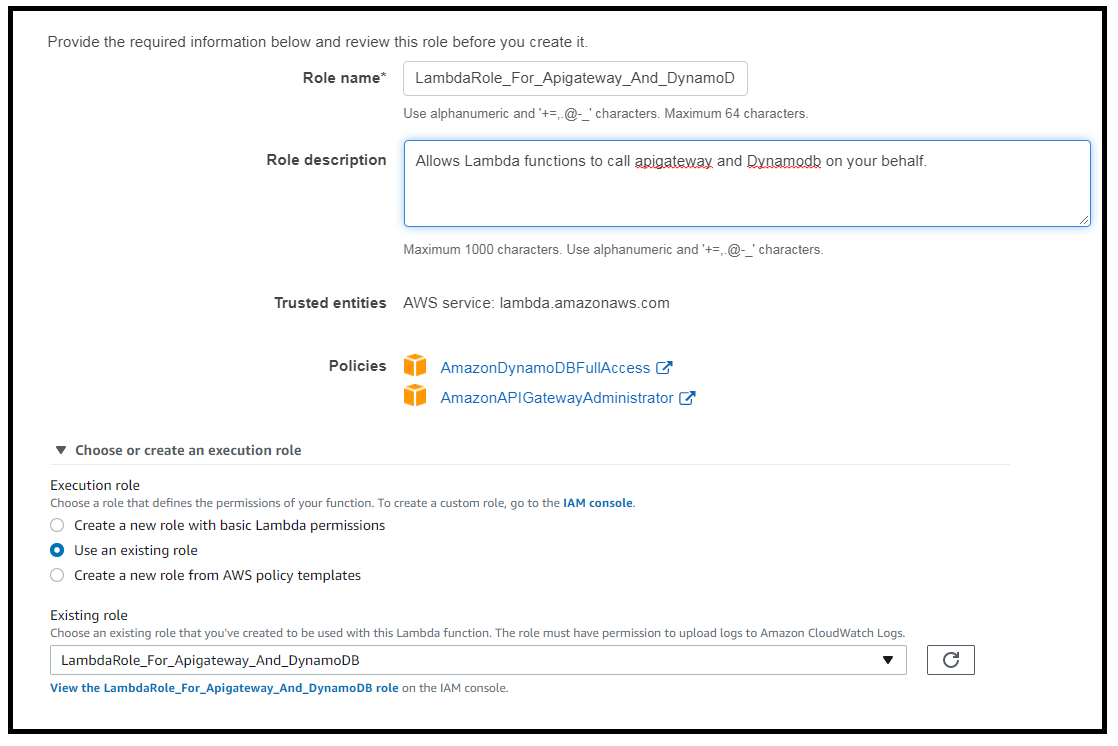
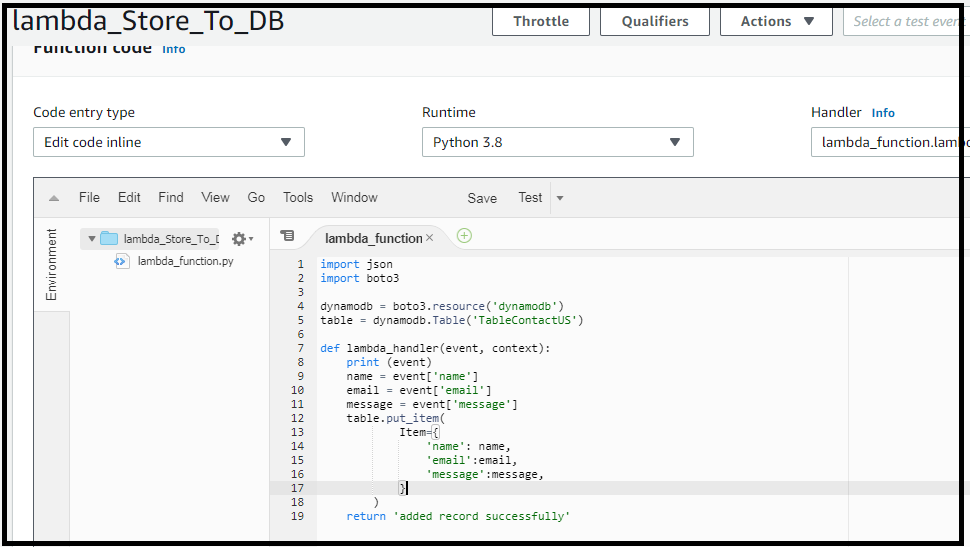
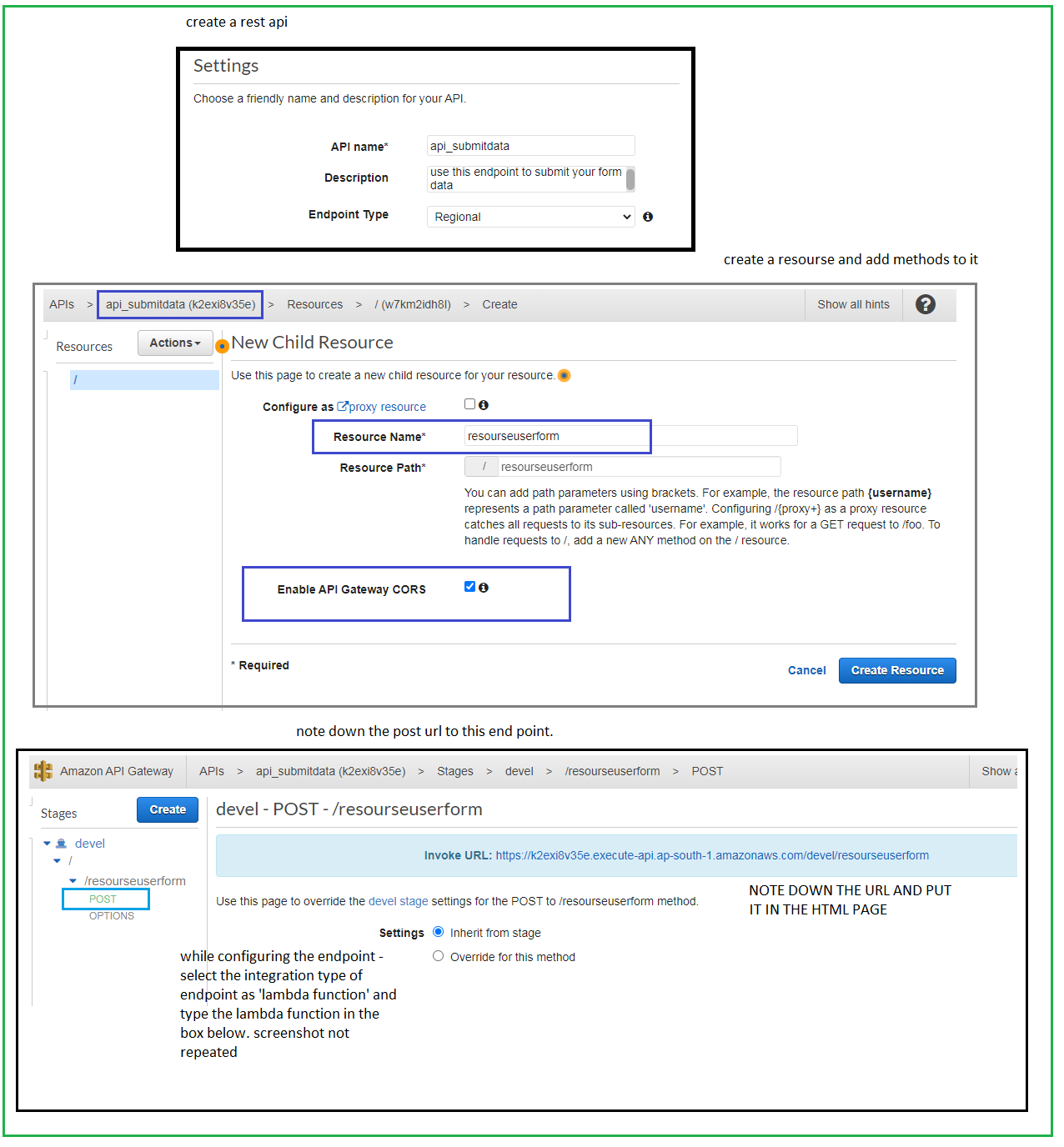
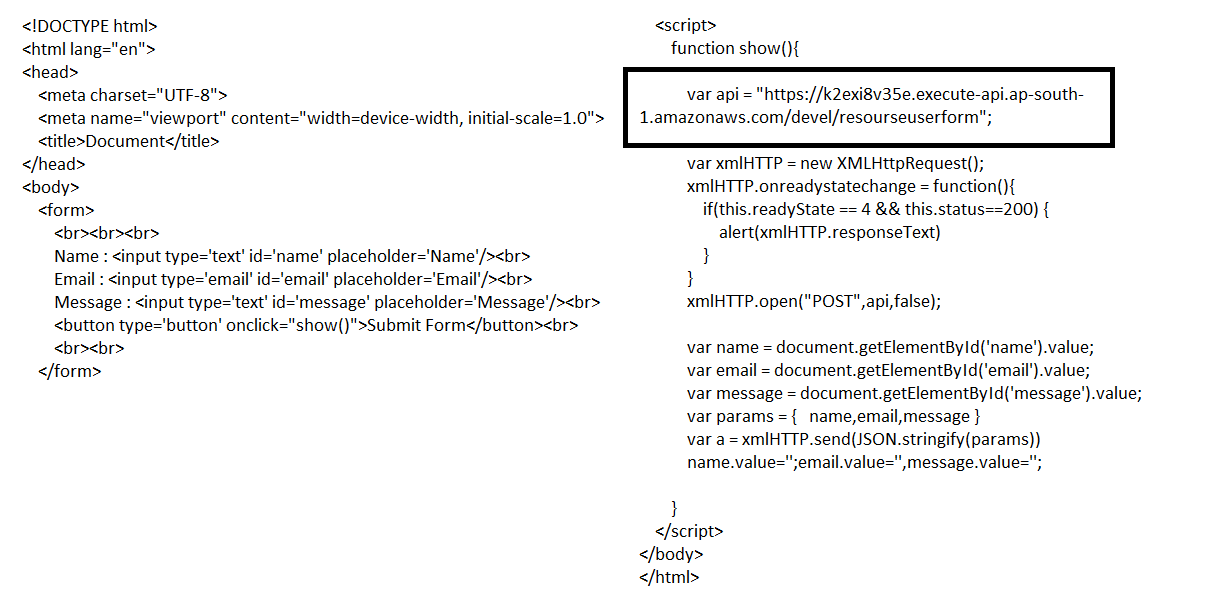
6.8 Configure the HTML (as below )that you upload into S3 [Static website hosting], to use the endpoint we created above.  
  
  
   
Access the website which make a call to the api gateway, Upon which the Lambda function will be triggered.

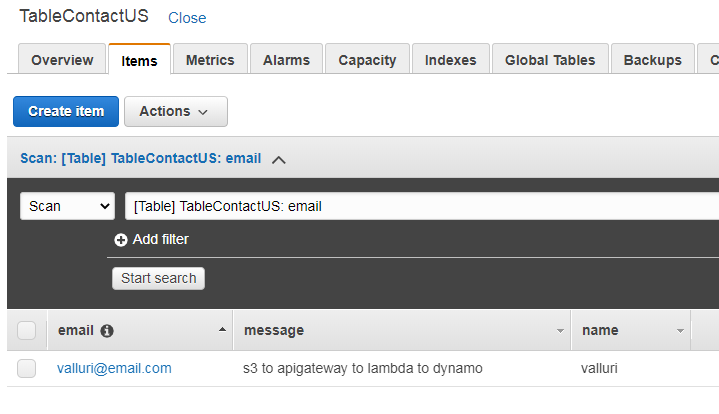
**7. What does a endpoint Stages options look like.  
7.1 Settings tab.  
  
7.2 Logs tab.  
  
7.3 Export Tab.  
  
7.4 Deployment History.  
  
7.5 Canary Tab.  
**

**8. Another Serverless Example – S3 – Api Gateway – Lambda – DynamoDB**We will fill a form hosted in S3, and when we submit the form we send a request to an api gateway endpoint, which will trigger a lambda function.  
This Lambda Function will store the data into a Dynamo DB data base.  
  
  
8.1 Dynamo DB dash board****

8.2 create the table.  
****

8.3 Leave the table as is. The table name is “TableContactUS”  
****

8.4 Create a lambda function that stores data to your Database.  
  
  
  
Create an IAM role as below and assign it to Lambda.  
  
  
****  
8.5 Create an Endpoint and configure the end point to use the above lambda .  
Don’t forget to enable cors and deploy the api.  
****8.6 : change the submit form url to the end point url as shown above And Upload it to S3 (static website hosting).****

8.7 after submitting the data the xml http request -> The request reaches the end point.  
The end point calls lambda function which then stores the data in the dynameo db database.  
****