Table of contents

Intoduction to serverless architecture on AWS :

- Going serverless on cloud is quickly gaining popularity due to below reasons

  - No need to provision or manage servers
  - No idle time and hence no paying for resources which are not working
  - Automated scaling

- When we talk about serverless architecture , it does not mean there are no servers . It simply means they are provisioned in the background where we dont have control or even a visual on the same

- There are few popular choices when it comes to building a serverless architecture

  - S3
  - API gateway
  - Dynamodb
  - Lambda
  - Glue

- While designing the architecture , one needs to be aware of what the service brings to the table compared with other services which might be capable of doing the same job (for ex EC2 vs RDS)

- We'll create a sample architecture consisting api gateway , lambda and dynamodb

What is api gateway

-API gateway is primarily used as a trigger where we would want to invoke services using an api calls like put , post and get.

- There are various concepts in api gateway , lets look at few basic terminologies
  - Rest api : It uses http request to get ,put, post and delete data : cowin , applications where backend data needs to be shared : most commonly used
  - Web socket api : supports 2 way communication : chat application
  - Stages : logical isolation feature for various environments like dev , stage and prod

Creating a sample api resource

- Let us first try and create a simple api trigger which can invoke our lambda function and give a response

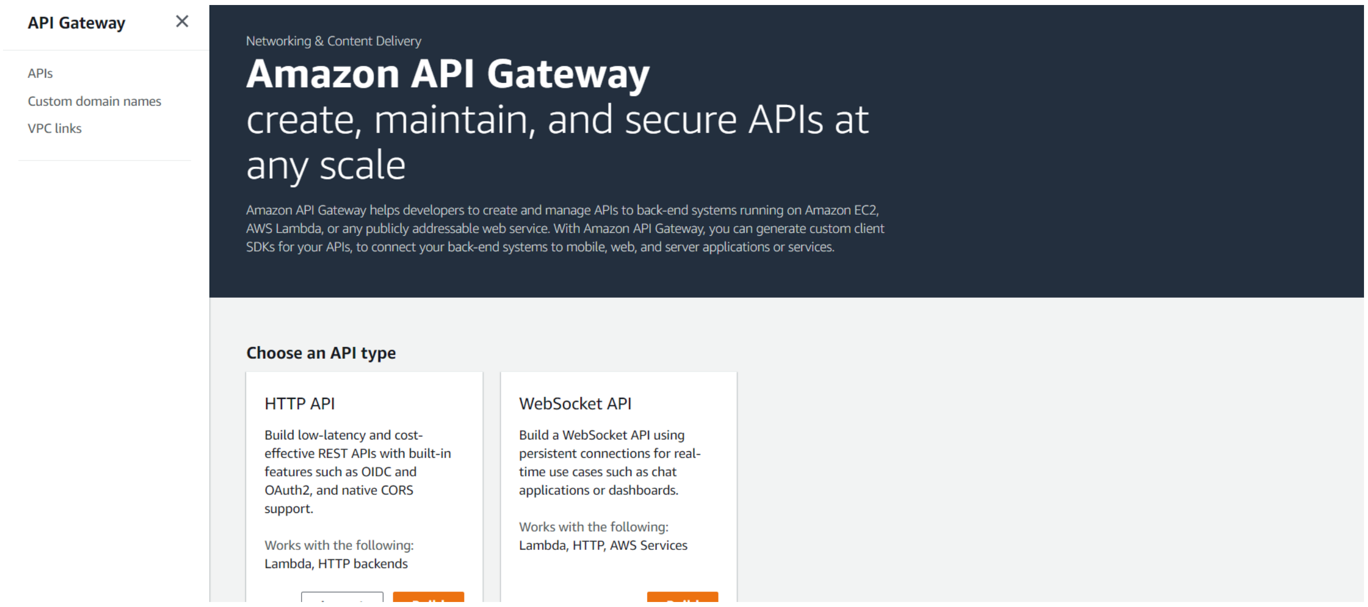
- Create a lambda function with any sample code , it need not be very complex . We can use below code as well for python 3.8

'''python import json

def lambda_handler(event, context): # TODO implement a = 10 b = 20 c = a+b return c

'''

- As we know lambda needs to invoked by some trigger , we have already seen invocation via cloudwatch events as well as s3 events .
- Let us now create an api gateway which can act as a trigger to invoke our lambda
- Navigate to api gateway



Networking & Content Delivery

Amazon API Gateway
create, maintain, and secure APIs at any scale

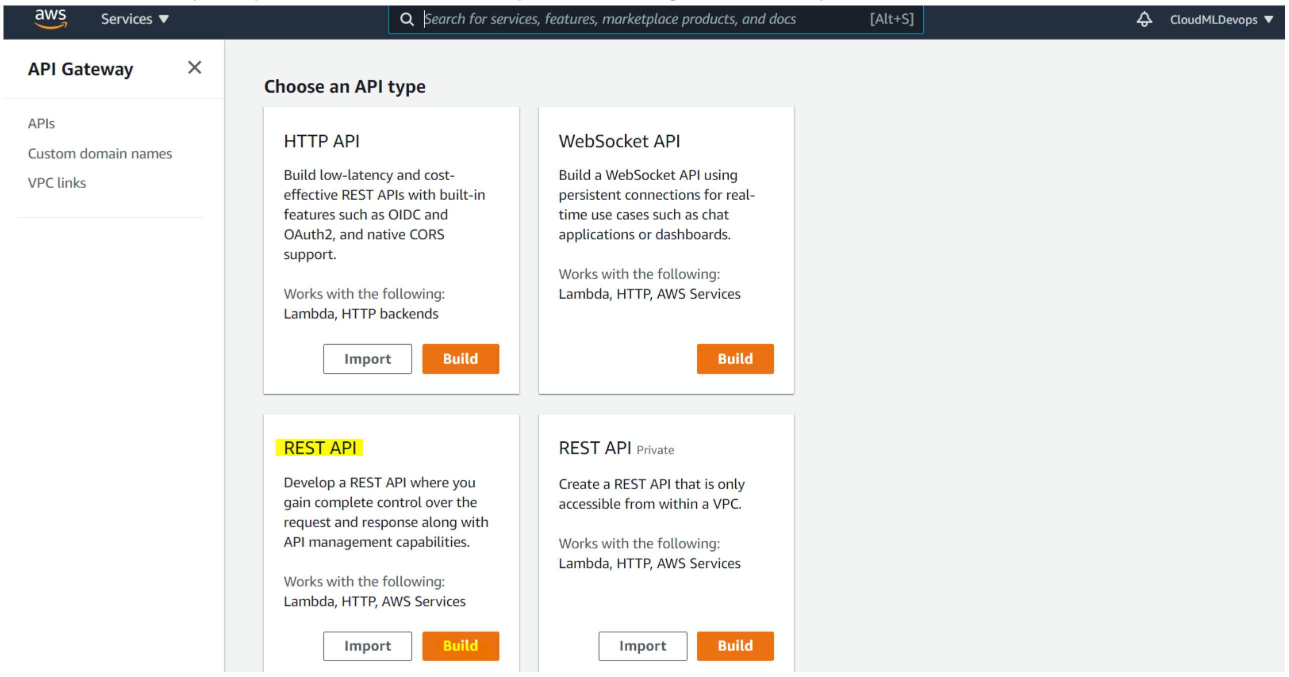
Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services.

Choose an API type

HTTP API
Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.
Works with the following:
Lambda, HTTP backends

WebSocket API
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.
Works with the following:
Lambda, HTTP, AWS Services

- You'd see multiple options to create the api . Select regional rest api and click on build



Choose an API type

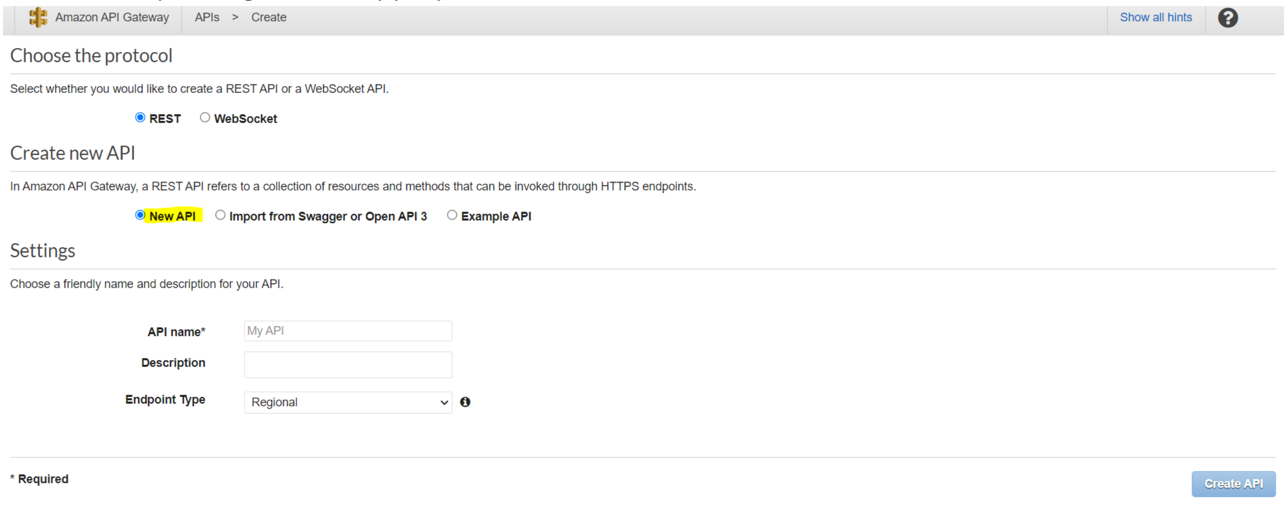
HTTP API
Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.
Works with the following:
Lambda, HTTP backends

WebSocket API
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.
Works with the following:
Lambda, HTTP, AWS Services

REST API
Develop a REST API where you gain complete control over the request and response along with API management capabilities.
Works with the following:
Lambda, HTTP, AWS Services

REST API Private
Create a REST API that is only accessible from within a VPC.
Works with the following:
Lambda, HTTP, AWS Services

- Select new api and give it an appropriate name



Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

[✓ REST] WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

[✓ New API] Import from Swagger or Open API 3 Example API

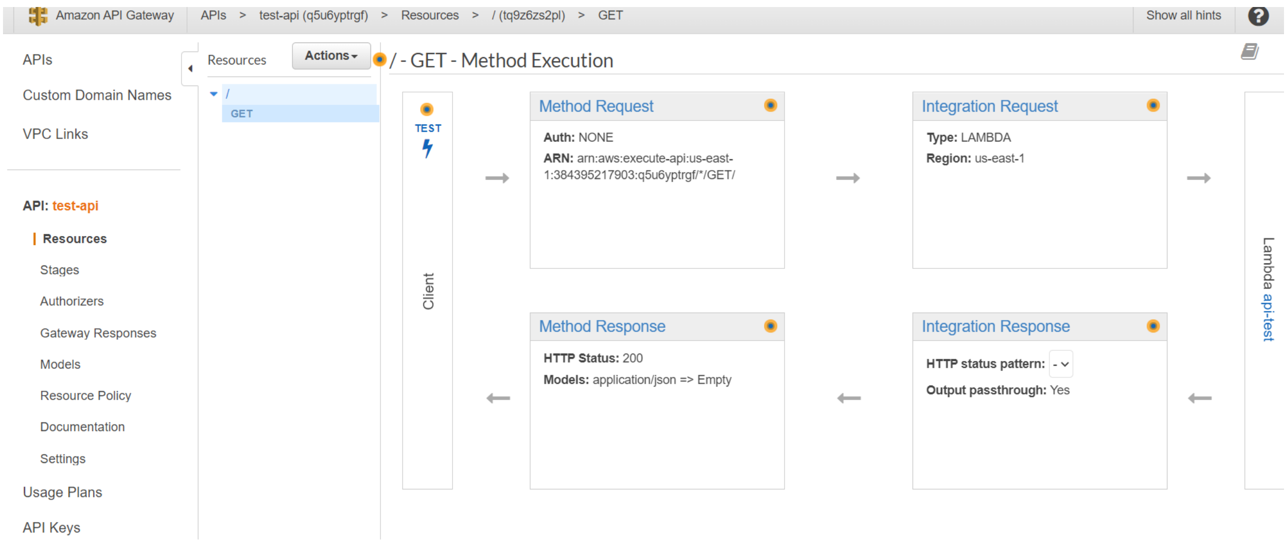
Settings

Choose a friendly name and description for your API.

API name*	My API
Description	
Endpoint Type	Regional

* Required

- The resources tab will be automatically selected . These resources are kind of subdomains to your api call . Default one is root denoted with "/"
- Under resorces we can create multiple methods which can be used as trigger to invoke various aws resources
- Click on actions and select create method
- Select get method
- Under integration , keep the default option of lambda , and use the funcion name that we had created earlier
- Click on save



/ - GET - Method Execution

Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:384395217903:q5u6yptrgf/*/GET/

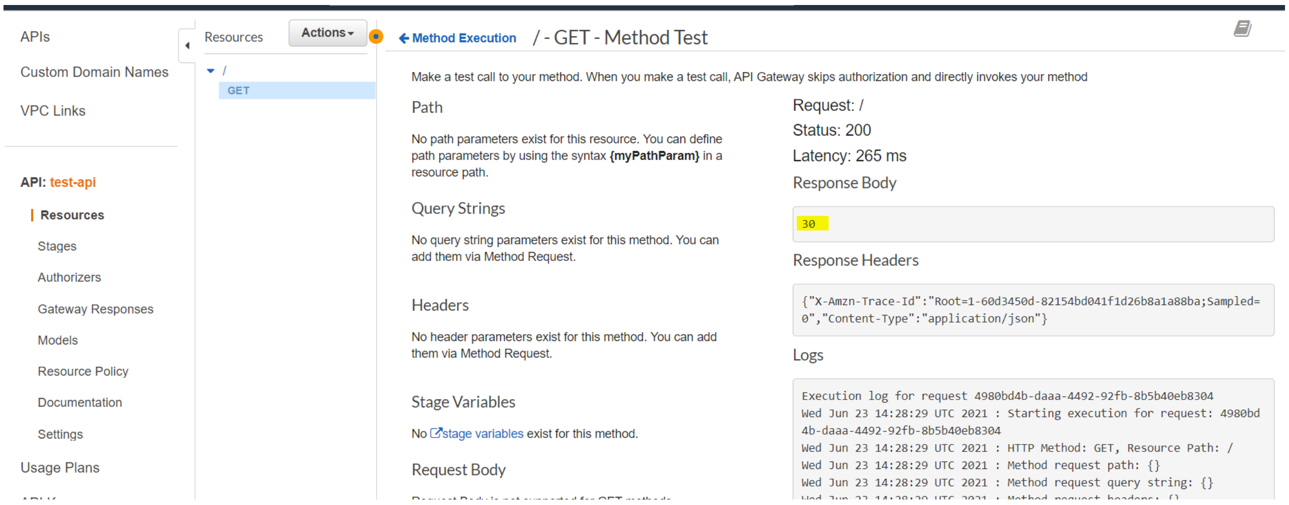
Integration Request
Type: LAMBDA
Region: us-east-1

Method Response
HTTP Status: 200
Models: application/json => Empty

Integration Response
HTTP status pattern: -
Output passthrough: Yes

- You can see the flow of request above in the method .

- You can simply click on test and see the invcation result of the lambda



/ - GET - Method Test

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

No query string parameters exist for this method. You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No stage variables exist for this method.

Request Body

Request: /
Status: 200
Latency: 265 ms

Response Body

```
30
```

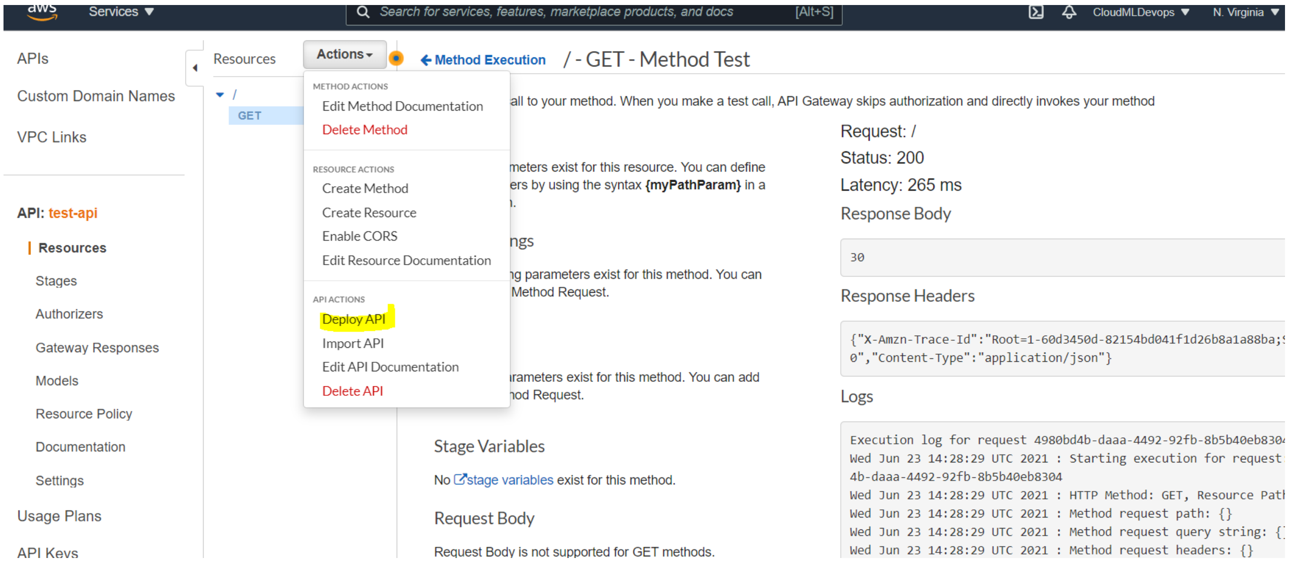
Response Headers

```
{"X-Amzn-Trace-Id":"Root=1-60d3450d-82154bd041f1d26b8a1a88ba;Sampled=
0","Content-Type":"application/json"}
```

Logs

```
Execution log for request 4980bd4b-daaa-4492-92fb-8b5b40eb8304
Wed Jun 23 14:28:29 UTC 2021 : Starting execution for request: 4980bd
4b-daaa-4492-92fb-8b5b40eb8304
Wed Jun 23 14:28:29 UTC 2021 : HTTP Method: GET, Resource Path: /
Wed Jun 23 14:28:29 UTC 2021 : Method request path: {}
Wed Jun 23 14:28:29 UTC 2021 : Method request query string: {}
```

- Once we are satsified with the required result , we can deploy the changes to a stage . This is to ensure that the lambda is now exposed to internet



/ - GET - Method Test

Request: /
Status: 200
Latency: 265 ms

Stage Variables

No stage variables exist for this method.

Request Body

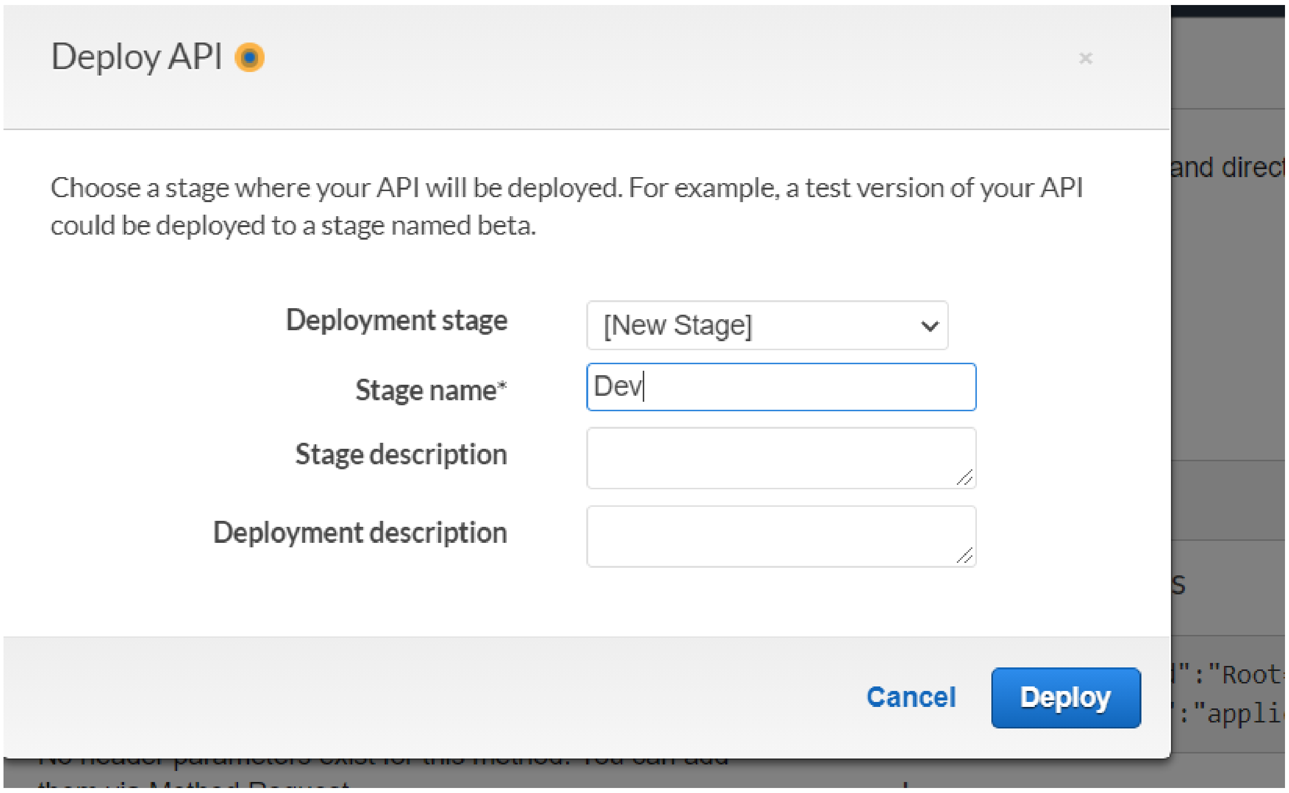
Request Body is not supported for GET methods.

Response Body

```
30
```

Logs

```
Execution log for request 4980bd4b-daaa-4492-92fb-8b5b40eb830
Wed Jun 23 14:28:29 UTC 2021 : Starting execution for request
4b-daaa-4492-92fb-8b5b40eb8304
Wed Jun 23 14:28:29 UTC 2021 : HTTP Method: GET, Resource Path
Wed Jun 23 14:28:29 UTC 2021 : Method request path: {}
Wed Jun 23 14:28:29 UTC 2021 : Method request query string: {
Wed Jun 23 14:28:29 UTC 2021 : Method request headers: {}
```

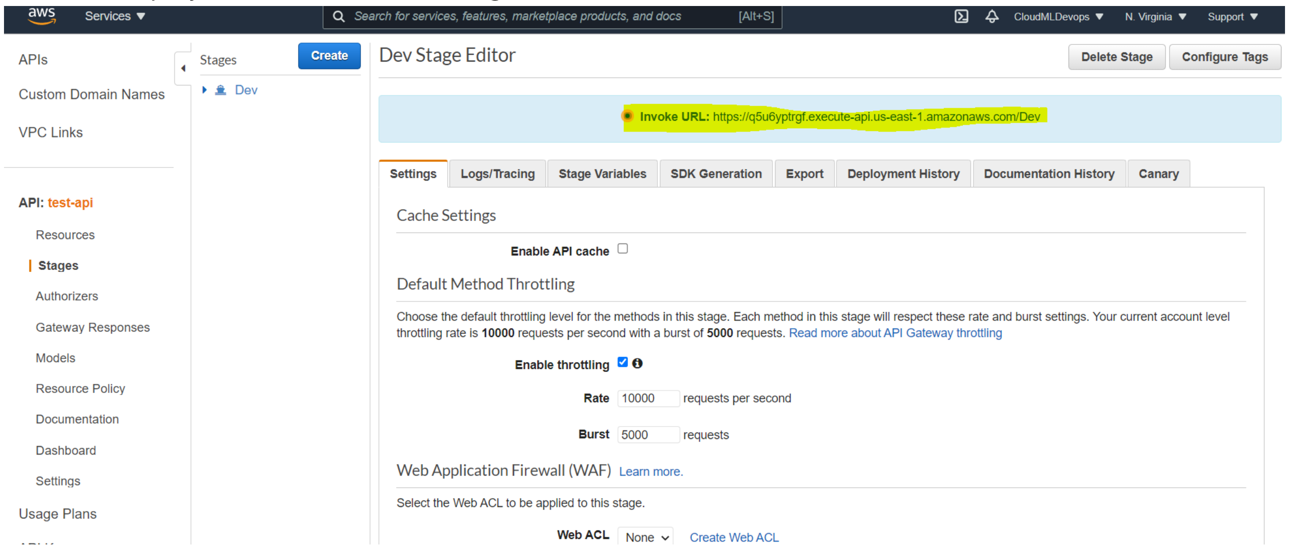


Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage	[New Stage]
Stage name*	Dev
Stage description	
Deployment description	

- Once we deploy the resource to the stage , it will be available to be used



Dev Stage Editor

Invoke URL: https://q5u6yptrgf.execute-api.us-east-1.amazonaws.com/Dev

Cache Settings

Enable API cache ☐

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. Read more about API Gateway throttling

Enable throttling ☑

Rate 10000 requests per second

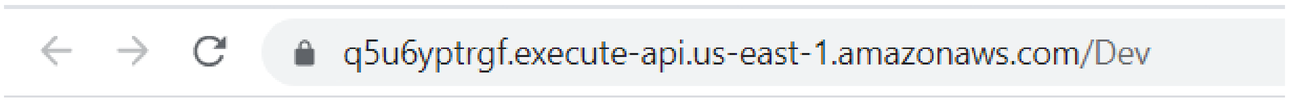
Burst 5000 requests

Web Application Firewall (WAF) Learn more.

Select the Web ACL to be applied to this stage.

Web ACL None Create Web ACL

- The invcation url is used to invoke this api



q5u6yptrgf.execute-api.us-east-1.amazonaws.com/Dev

```
30
```

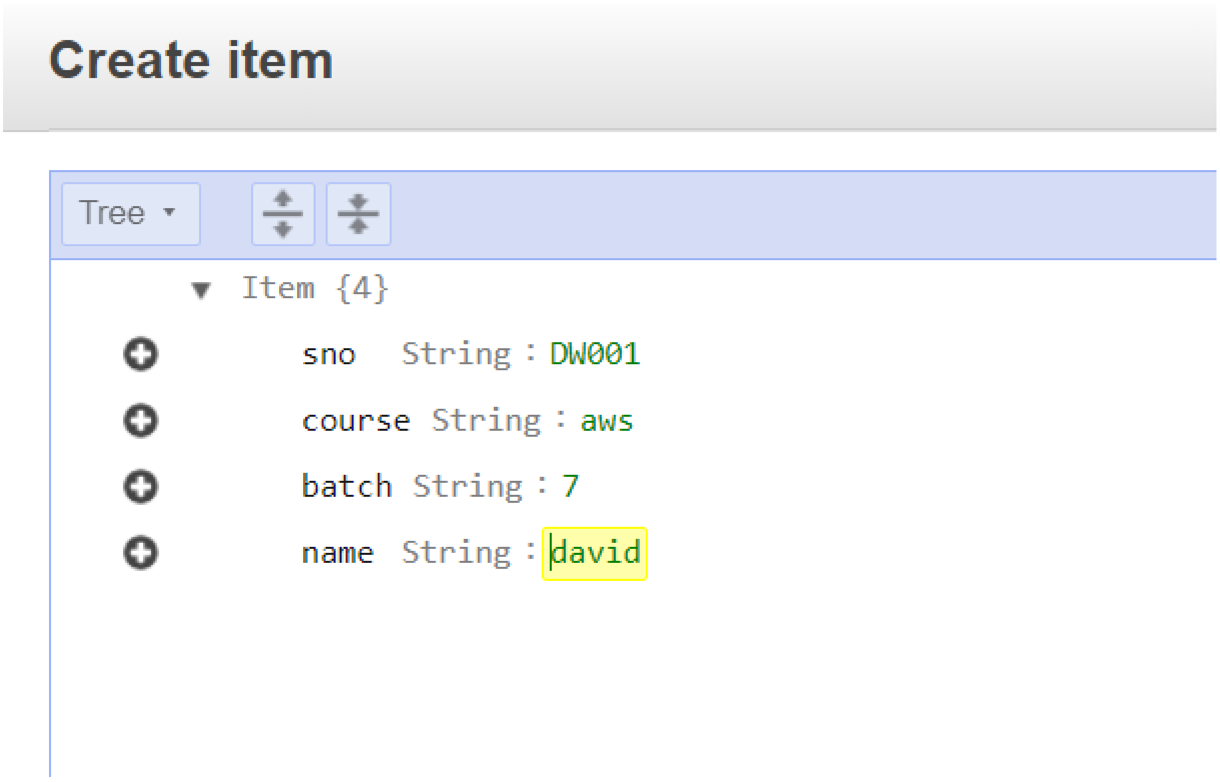
Dynamodb

- Dynamodb is a NoSql database offering by AWS
- It is completely serverless and works on key value based entries
- Since its a NoSql db , it supports dynamic schema creation
- Items in the table represents a record
- Primary key and index can be considered as a parallel to partition in athena
- DynamoDB supports ACID transactions
- Similar to all other aws services , DDB can be accessed and used by console , SDK or CLI

Creating a table and adding entries in DDB

- Navigate to DDB
- Click on create table
- Give an appropriate name and the primary key . This key would act as a partition . Once included in the query , the results would be much faster . Similar to parition in athena
- Keep the rest of the settings as default and click on create table
- Items tab is where all the records would be stored . Let us enter a sample record . Remember , there is no specific schema that needs to be followed other than the primary key we have entered

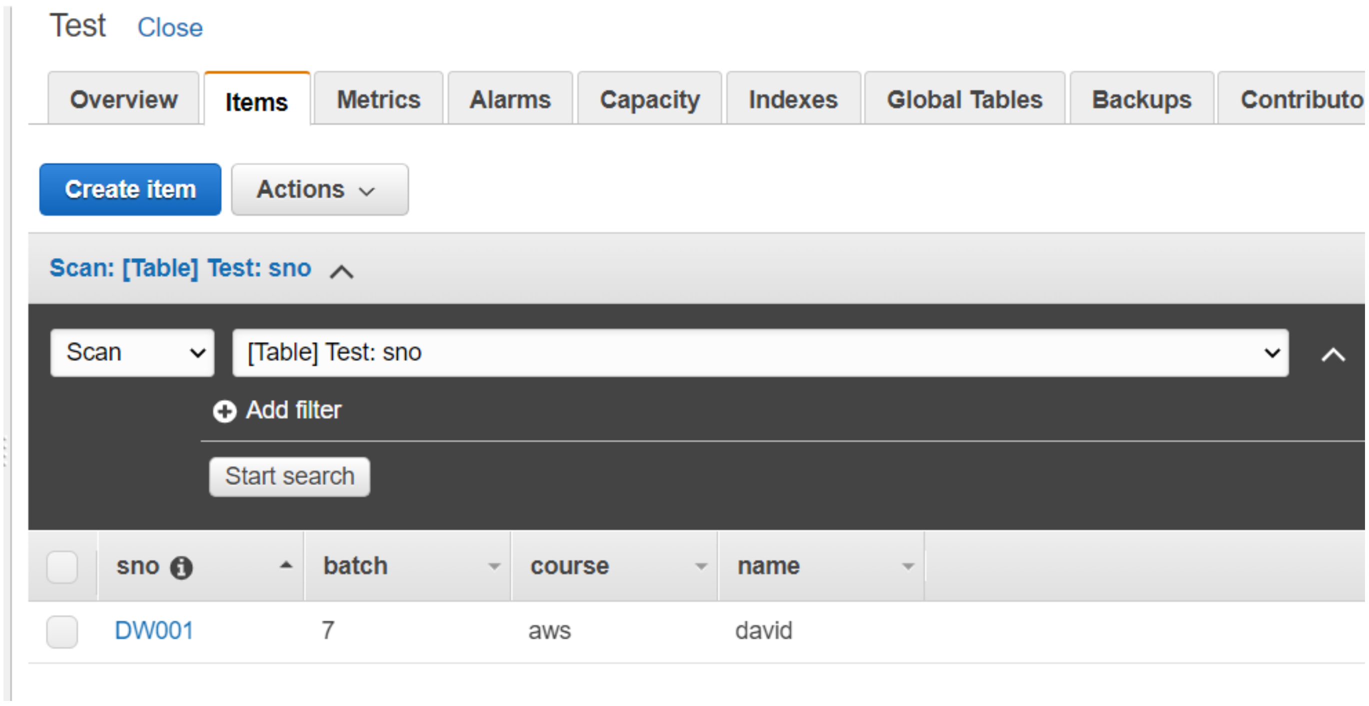
- Click on create item



Create item

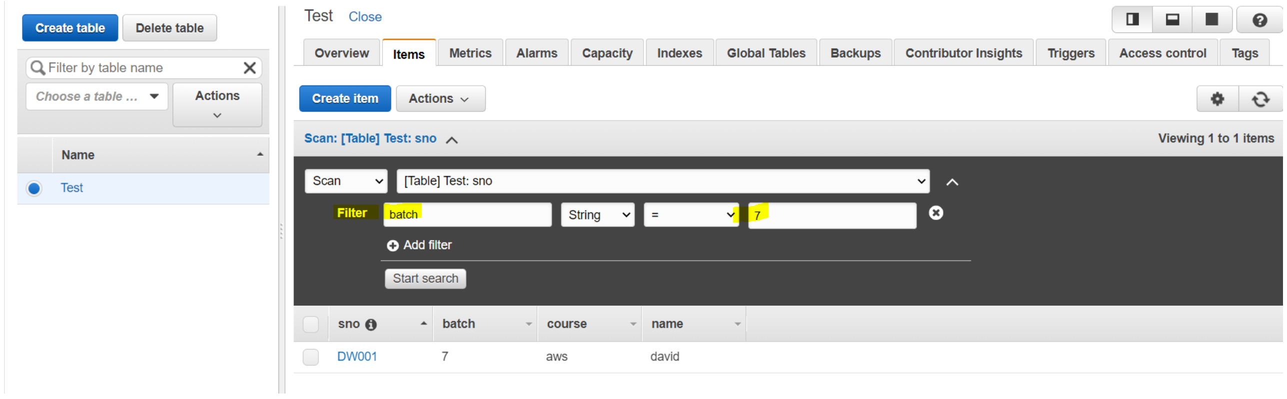
```
▼ Item {4}
    sno     String : DW001
    course  String : aws
    batch   String : 7
    name    String : david
```

- Add items by clicking on "+" and selecting append
- Once you have added the record , click on save



sno	batch	course	name
DW001	7	aws	david

- Same data can be queried by using the filter tab just above items



Scan: [Table] Test: sno — Viewing 1 to 1 items

Scan | [Table] Test: sno

Filter: batch | String | = | 7

sno	batch	course	name
DW001	7	aws	david

Using api gateway and lambda to interact with DDB

- Since DDB is serverless , it can be a good fit where we want entire serverless architecture with a db
- Using lambda with boto , one can easily interact with the DDB tables
- In order to print a specific table from dynamodb , we can use below code

'''bash import json import boto3 dynamodb = boto3.resource('dynamodb') table = dynamodb.Table('Test')

def lambda_handler(event, context): response = table.scan() print(response) #print(response[item])

```
# TODO implement
return {
    'statusCode': 200,
    'body': json.dumps(response)
}
```

'''

- Here the table name Test has to be replaced with the table name that you have created
-
- We can also insert or pass information to the api call while invoking it .
- This information is stored in the event object and can be later utilized to operate on
- In below example , lets see if we can push data to DDB table using api gateway
- For this we'll be using post method which is generally used to insert/pass data
- Use below lambda code , make sure the function has permissions to interact with the DDB table we created earier

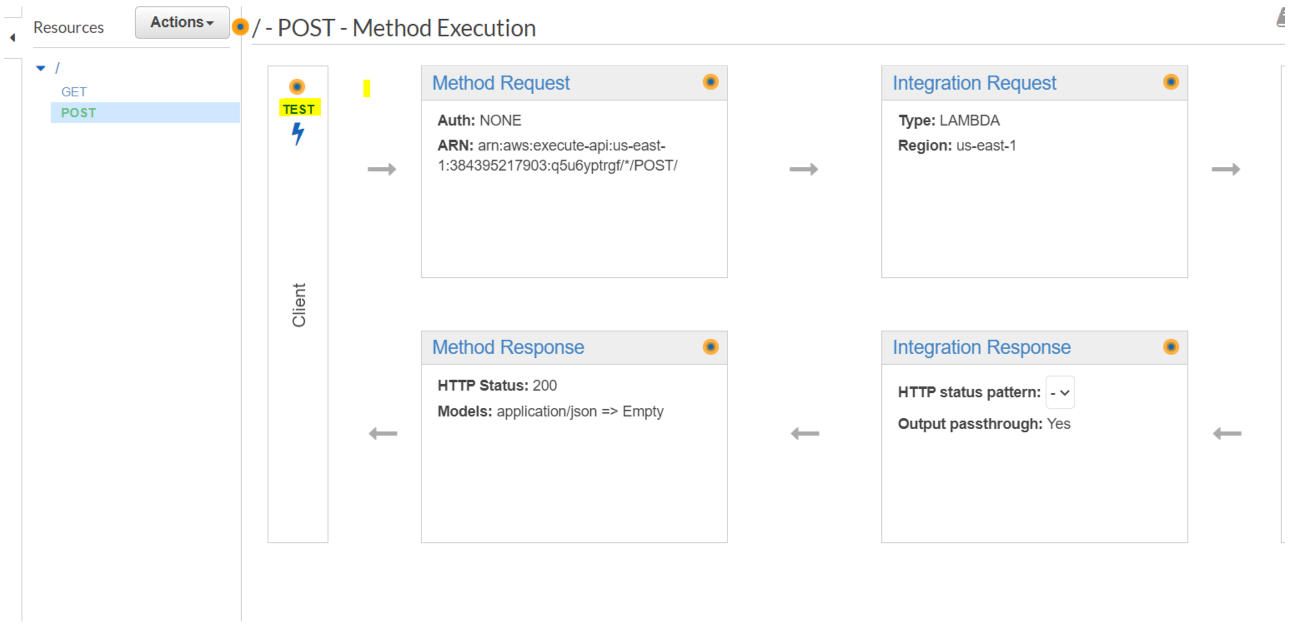
'''bash import json import boto3 dynamodb = boto3.resource('dynamodb') table = dynamodb.Table('Test') def lambda_handler(event, context): # TODO implement table.put_item(Item=event) return { "statusCode": 200, "message": "record inserted successfully" }

'''

- Above function is going to get the event from api gateway and use it to put the data to DDB table 'Test'

- Let us now create the invocation for the same in api gateway

- Navigate to the resource we created earlier and click on actions and select post method
- Similar to get method , here give the lambda function name that we created earlier
- Once done save the settings
- Now in order to test it , we'll have to pass a request body or event json



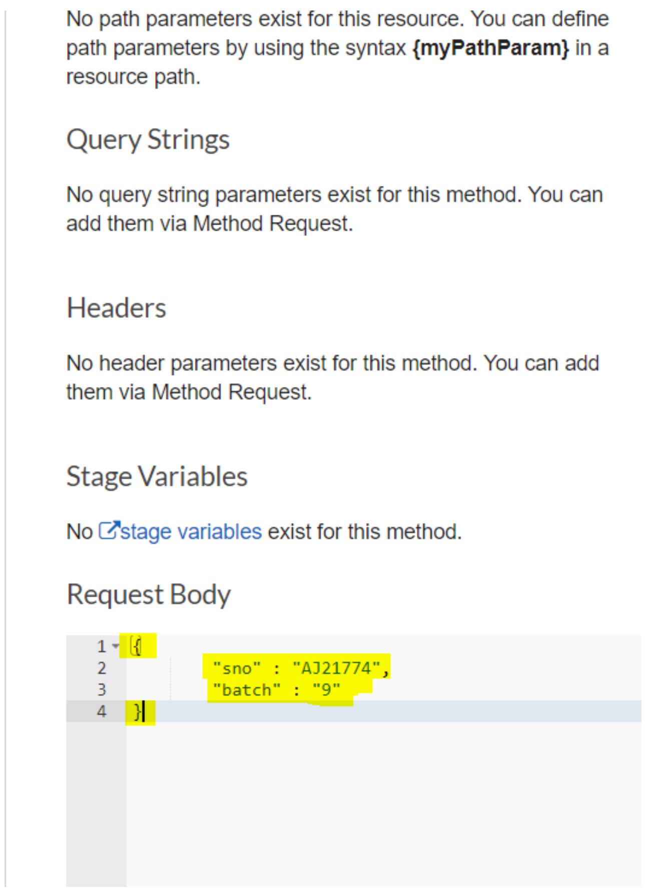
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:384395217903:q5u6yptrgf/*/POST/

Type: LAMBDA
Region: us-east-1

HTTP Status: 200
Models: application/json => Empty

HTTP status pattern: -
Output passthrough: Yes

- Click on test


No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

No query string parameters exist for this method. You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

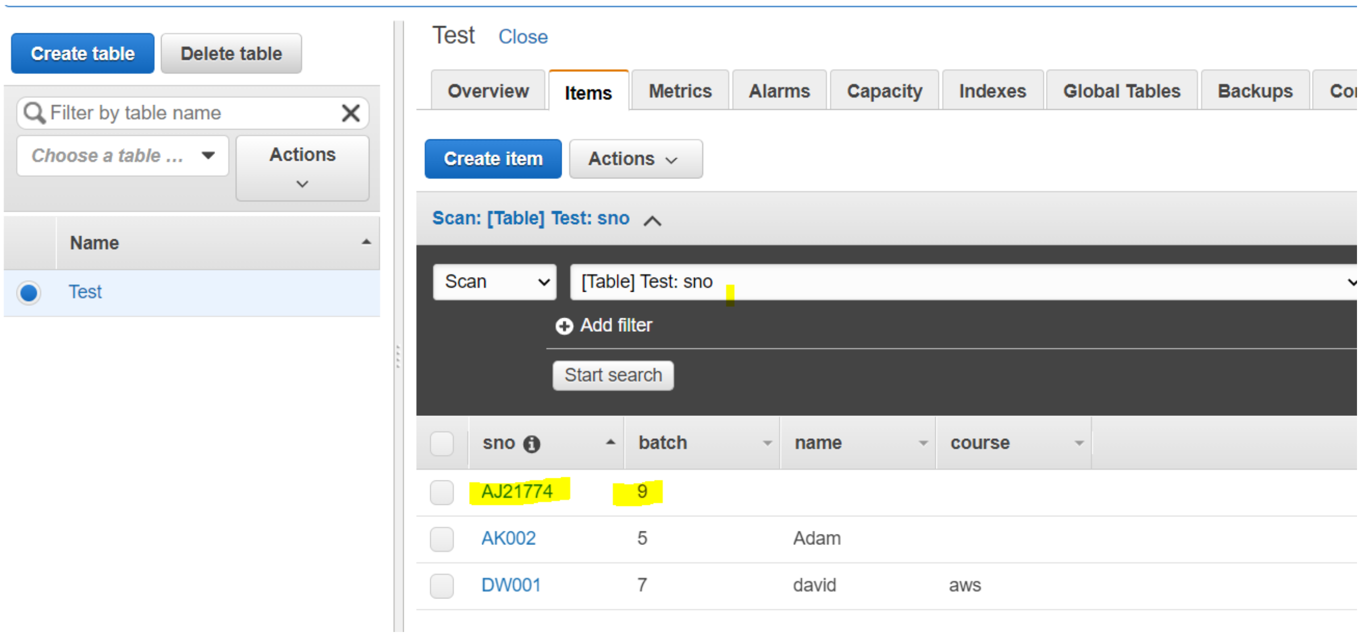
Stage Variables

No stage variables exist for this method.

Request Body

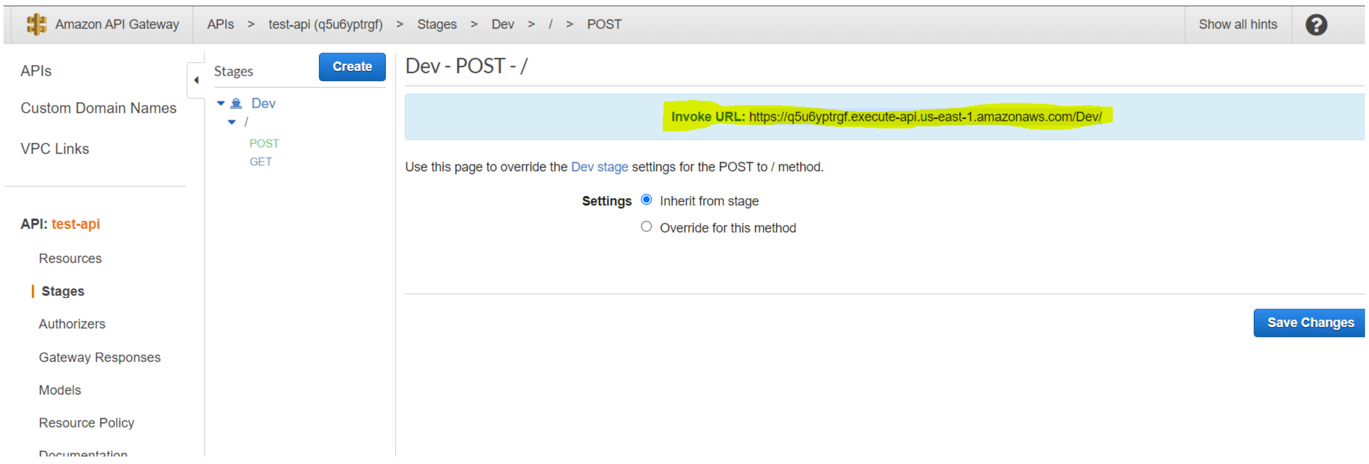
```
{
    "sno" : "AJ21774",
    "batch" : "9"
}
```

- We will pass a json with a sample schema which will be inserted .
- Once you click on test , see if the data is inserted in the table



sno	batch	name	course
AJ21774	9		
AK002	5	Adam	
DW001	7	david	aws

- Now this has been a test scenario where you would be testing it in the api gateway UI
- Now when it comes to atually using it , we deploy the resource to a stage and invoke it using a url

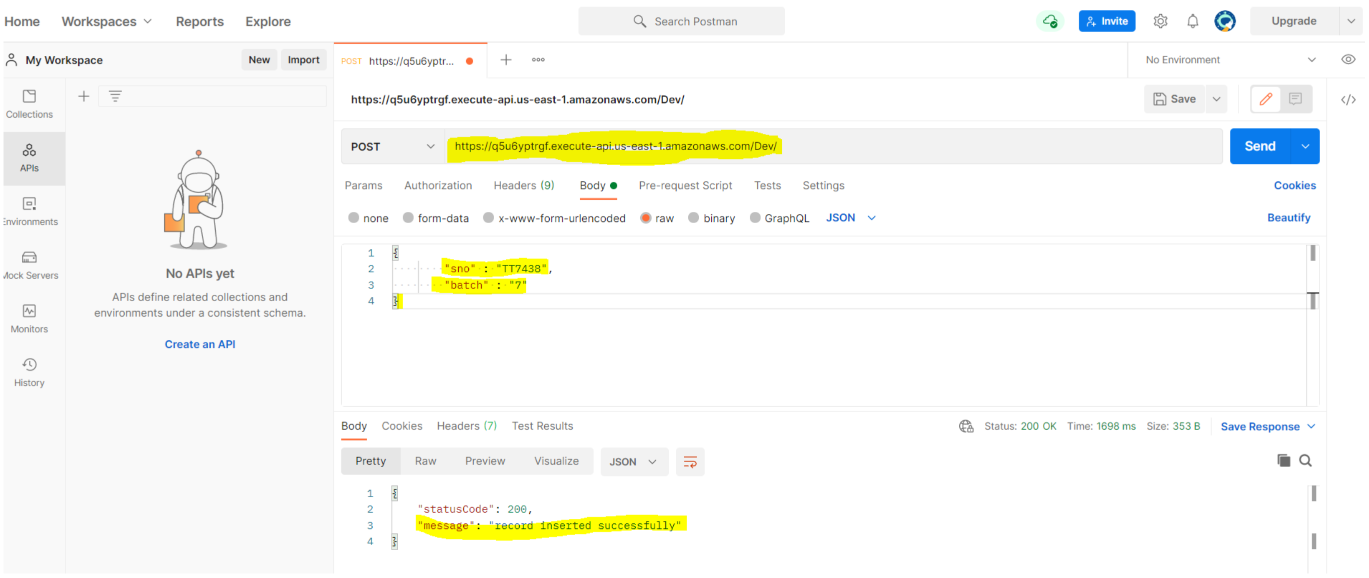


Invoke URL: https://q5u6yptrgf.execute-api.us-east-1.amazonaws.com/Dev/

Use this page to override the Dev stage settings for the POST to / method.

Settings [✓ Inherit from stage] Override for this method

- However in scenarios where we want to pass a request body with the api call , tools like postman comes in handy
- These tools help us make api calls and pass a request body along with it
- You can install postman on the machine which can be accessed via a browser
- Once installed select api , select post method and paste the invcation url



https://q5u6yptrgf.execute-api.us-east-1.amazonaws.com/Dev/

POST https://q5u6yptrgf.execute-api.us-east-1.amazonaws.com/Dev/

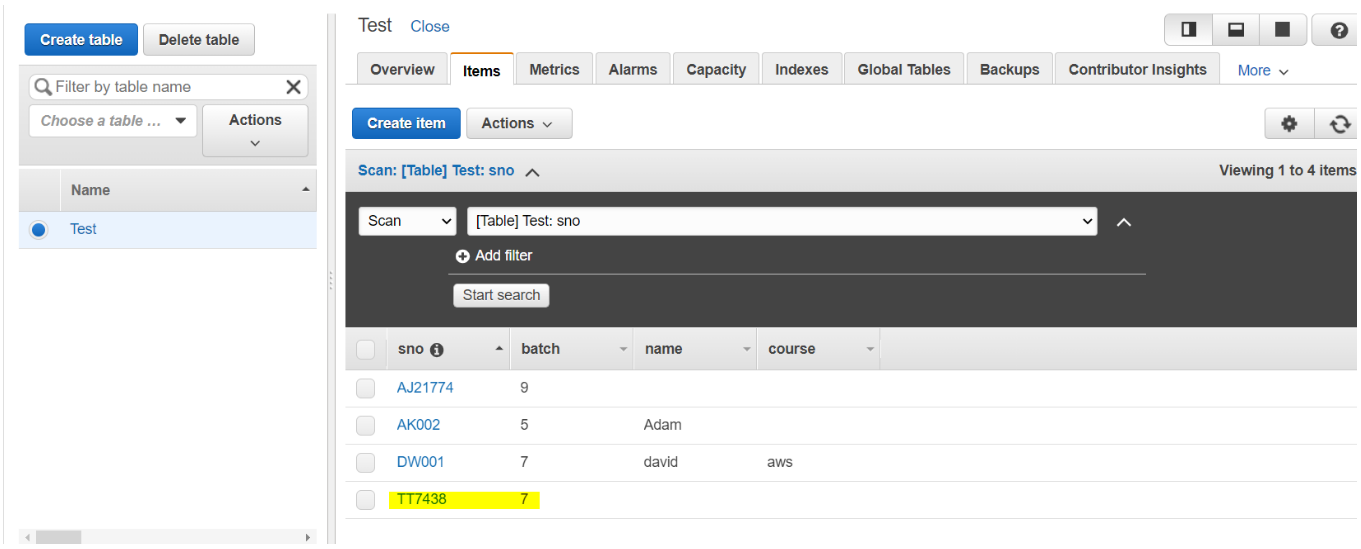
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "sno" : "TT7438",
    "batch" : "7"
}
```

Body — Status: 200 OK Time: 1698 ms Size: 353 B — JSON

```
{
    "statusCode": 200,
    "message": "record inserted successfully"
}
```

- under body select raw and from drop down select json and paste a sample json
- click on send



Scan: [Table] Test: sno — Viewing 1 to 4 items

Scan — [Table] Test: sno

sno	batch	name	course
AJ21774	9		
AK002	5	Adam	
DW001	7	david	aws
TT7438	7		