ASSIGNMENT 2

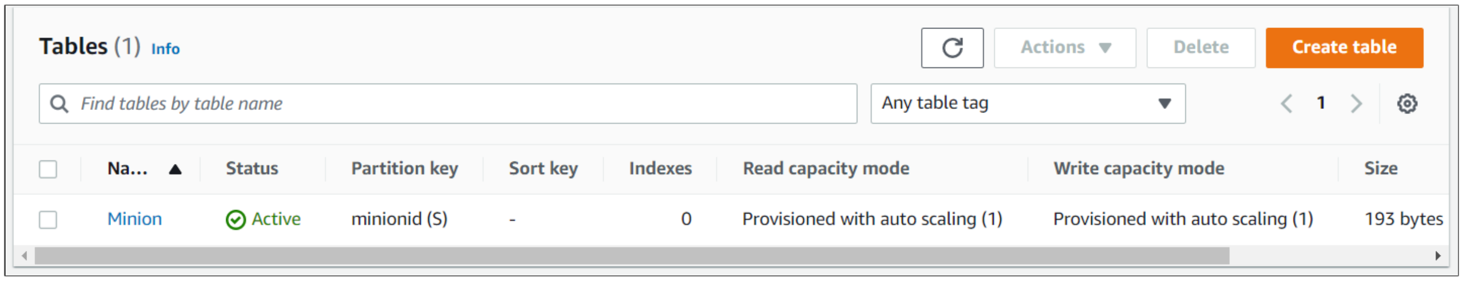
HARSHIKA AKULA (2075727)

This report explains how we build a Minion Summoning API for Gru that is we will be implementing a lambda function which will be invoked whenever Gru requests a minion.

Process:

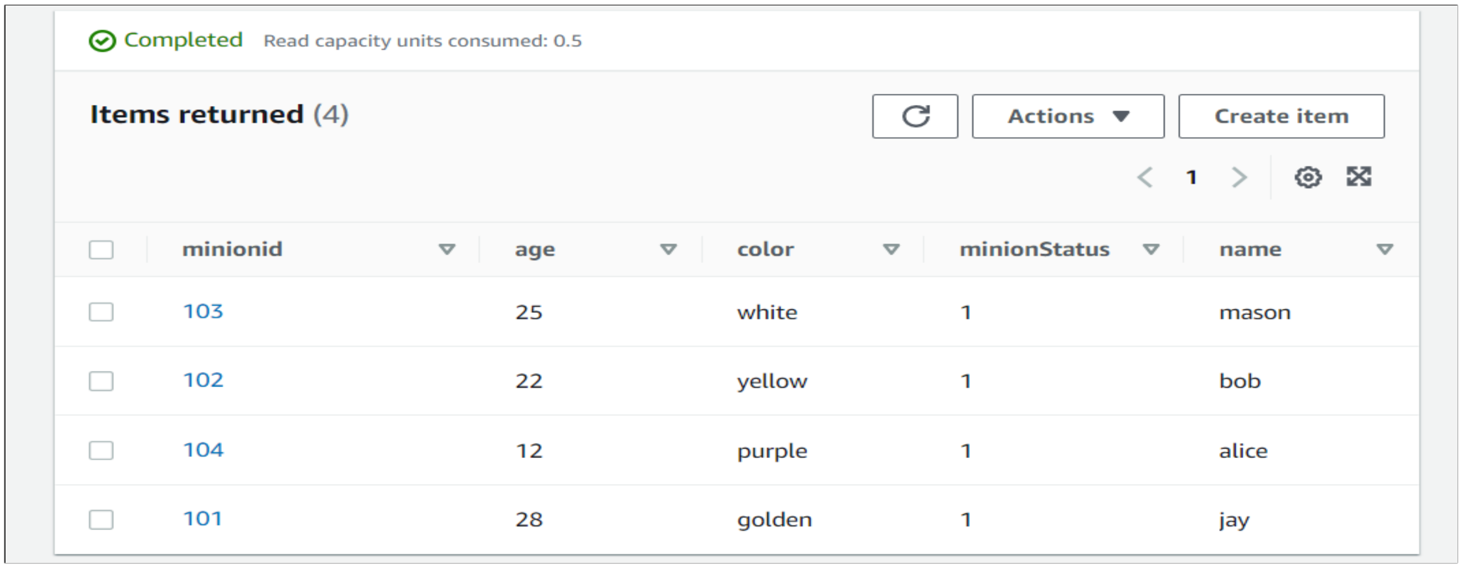
Step1: Create a DynamoDB table named "Minion":

In this step we create a table named **Minion** in DynamoDB Console. I have also given Partition key/Primary key with type string as "**minionid**" which will be unique throughout the table. Added three more attributes/items such as **"age", "color" and "name"** and one **status flag** (set to "1"by default) to know if the minion with particular "**id**" has already been summoned or not. I have used the defaults for all the other settings. Below is the final screenshot of creating of "**Minion**" table in Console and respective items in table.



Tables (1) Info

	Na...	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode	Size
☐	Minion	Active	minionid (S)	-	0	Provisioned with auto scaling (1)	Provisioned with auto scaling (1)	193 bytes



Completed Read capacity units consumed: 0.5

Items returned (4)

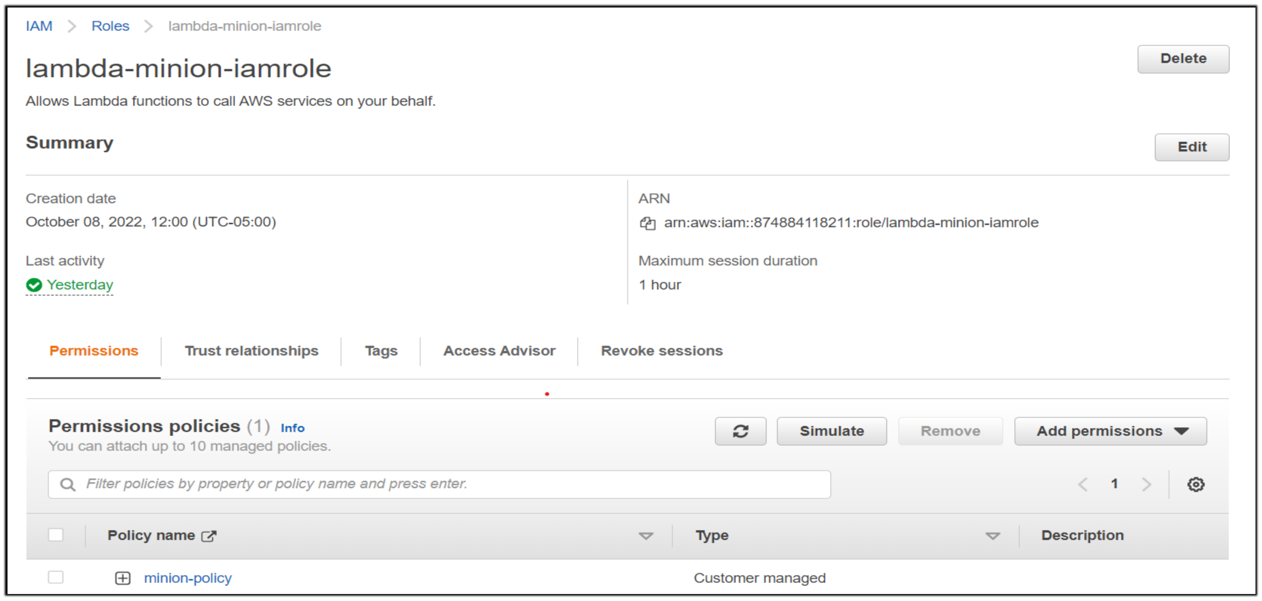
	minionid	age	color	minionStatus	name
☐	103	25	white	1	mason
☐	102	22	yellow	1	bob
☐	104	12	purple	1	alice
☐	101	28	golden	1	jay

Step2: Create a Lambda function "RequestMinion":

Before creating the lambda function, we should also create an IAM role which will grant the permission for the lambda function to read/write items to our DynamoDB table. I have created a **"lambda-minion-iamrole"** and attach the custom inline policy name "minion-policy". Then we will create a Lambda function named "**RequestMinion**" by configuring the IAM role we created previously, and we select python 3.9 as our runtime.

In lambda function, we first import boto3 module interact with DynamoDB. We create an instance "**dynamodb**" to get the access of the resource DynamoDB. We create an object "**table**" to store the table "Minion". We have used method **get_item** and passed the key name as minionid which should match with partition key of Minion Table. So, whenever we give a particular minionid, we will return the status code and particular details of the minion like name, color, and age from Minion table. When the minion with same minionid gets called again, it will return a message saying, "**This minion was already called**!!". In order not to get the duplicate details, we are setting a status flag (either as "0" or "1") as an attribute in data table and set it all default values to 1. When there is no minion with specific minionid in the table, it returns a message saying, "**No Minion found with this id**".

Below are the two screenshots for specific role and specific lambda function created in console.



IAM > Roles > lambda-minion-iamrole

lambda-minion-iamrole

Allows Lambda functions to call AWS services on your behalf.

Summary

Creation date
October 08, 2022, 12:00 (UTC-05:00)

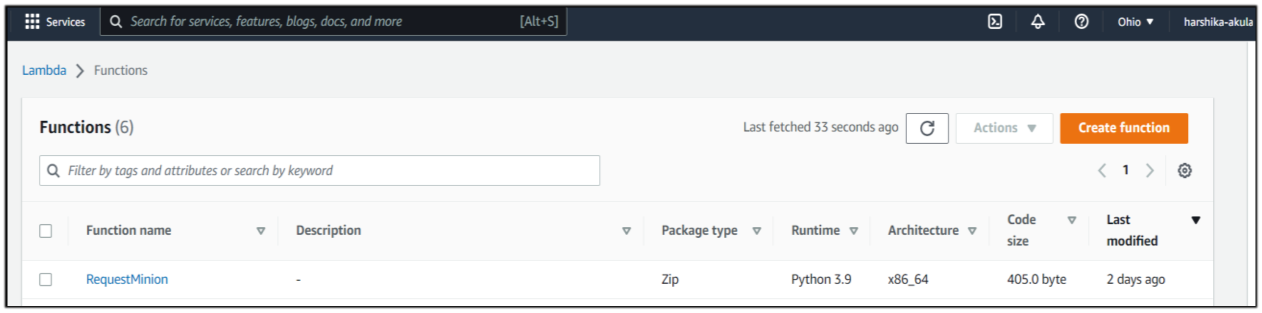
Last activity
Yesterday

ARN
arn:aws:iam::874884118211:role/lambda-minion-iamrole

Maximum session duration
1 hour

Permissions policies (1) Info
You can attach up to 10 managed policies.

Policy name	Type	Description
minion-policy	Customer managed	



Lambda > Functions

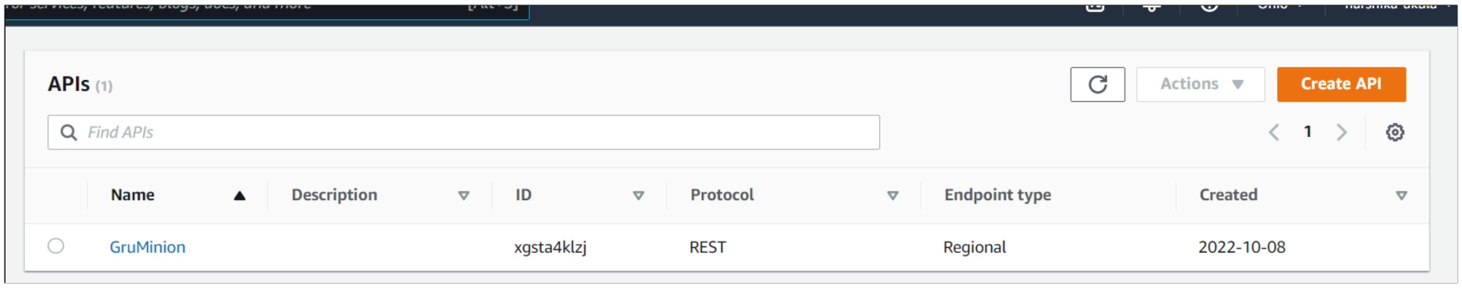
Functions (6)

Last fetched 33 seconds ago

Function name	Description	Package type	Runtime	Architecture	Code size	Last modified
RequestMinion	-	Zip	Python 3.9	x86_64	405.0 byte	2 days ago

Step3: Create an API named “GruMinion”:

In this step, we create a public REST API by clicking on build which will take us to Create New API tab where we give our API name as “**GruMinion**” with Endpoint Type as Regional. Below is the screenshot for API created.

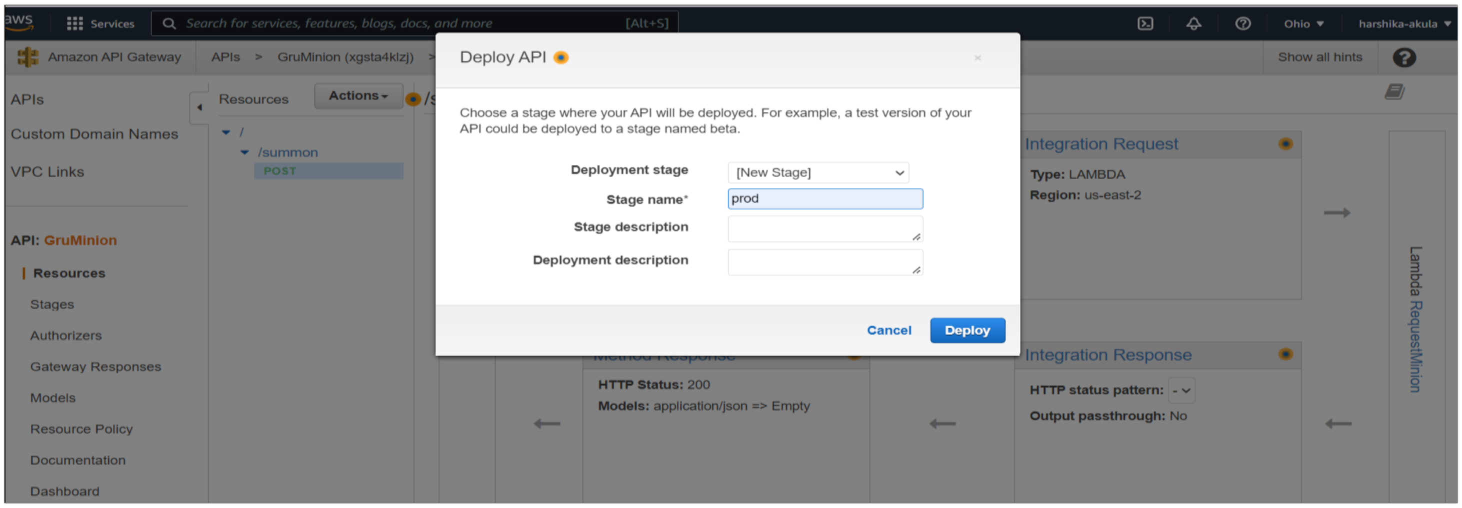


APIs (1)

Name	Description	ID	Protocol	Endpoint type	Created
GruMinion		xgsta4klzj	REST	Regional	2022-10-08

Step4: Create a new resource called “summon” within your API and Deploy API

We will create a resource naming “**summon**” and path variable will be **“/summon”**. Under that resource, we can create method based on what type of http method we want to introduce. We will be using **POST** method with specifying integration type as lambda function and giving lambda function name and region.



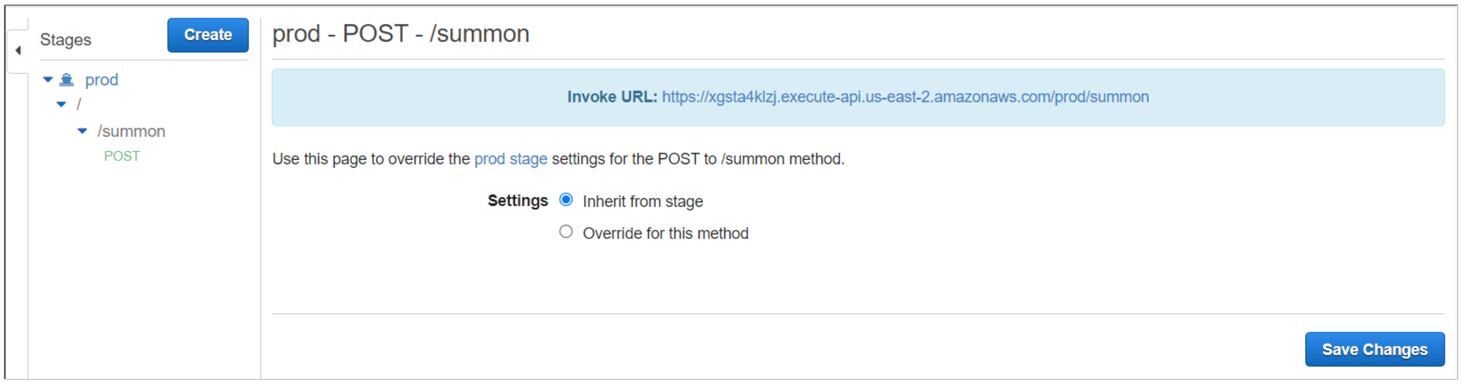
Deploy API

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*: prod
Stage description:
Deployment description:

Type: LAMBDA
Region: us-east-2
HTTP Status: 200
Models: application/json => Empty
HTTP status pattern: -
Output passthrough: No

We click Deploy API under actions and create a **prod** stage. Go to Stages tab on the left and click on the POST method. We can see the API endpoint-based URL that is used to access the API.



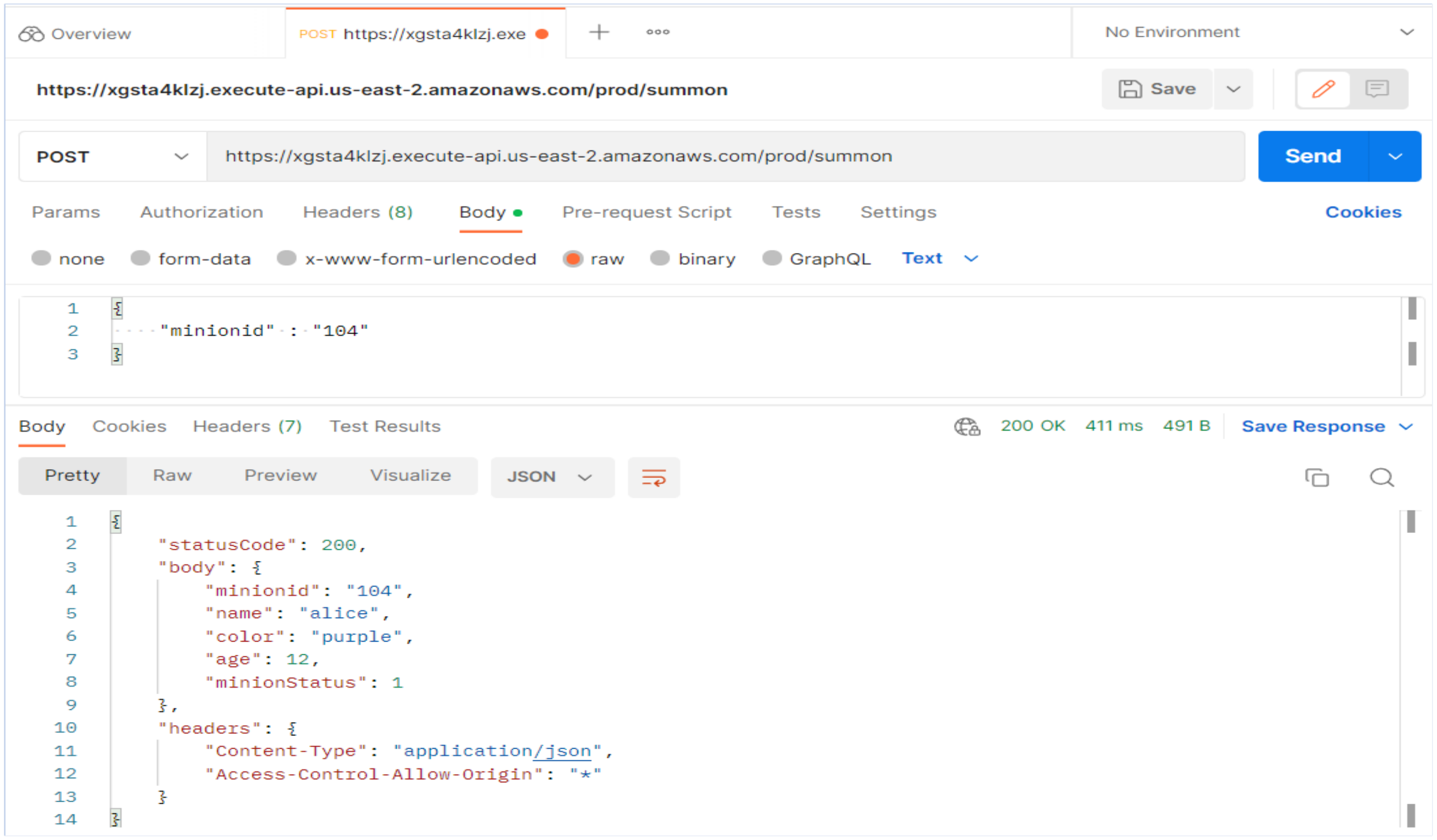
prod - POST - /summon

Invoke URL: https://xgsta4klzj.execute-api.us-east-2.amazonaws.com/prod/summon

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

Settings [✓ Inherit from stage]
Override for this method

We copy this URL and paste it in postman with POST method. We will mention the minionid in the body.
Below is the screenshot and Test Case when minion with specific minionid is called for the first time.



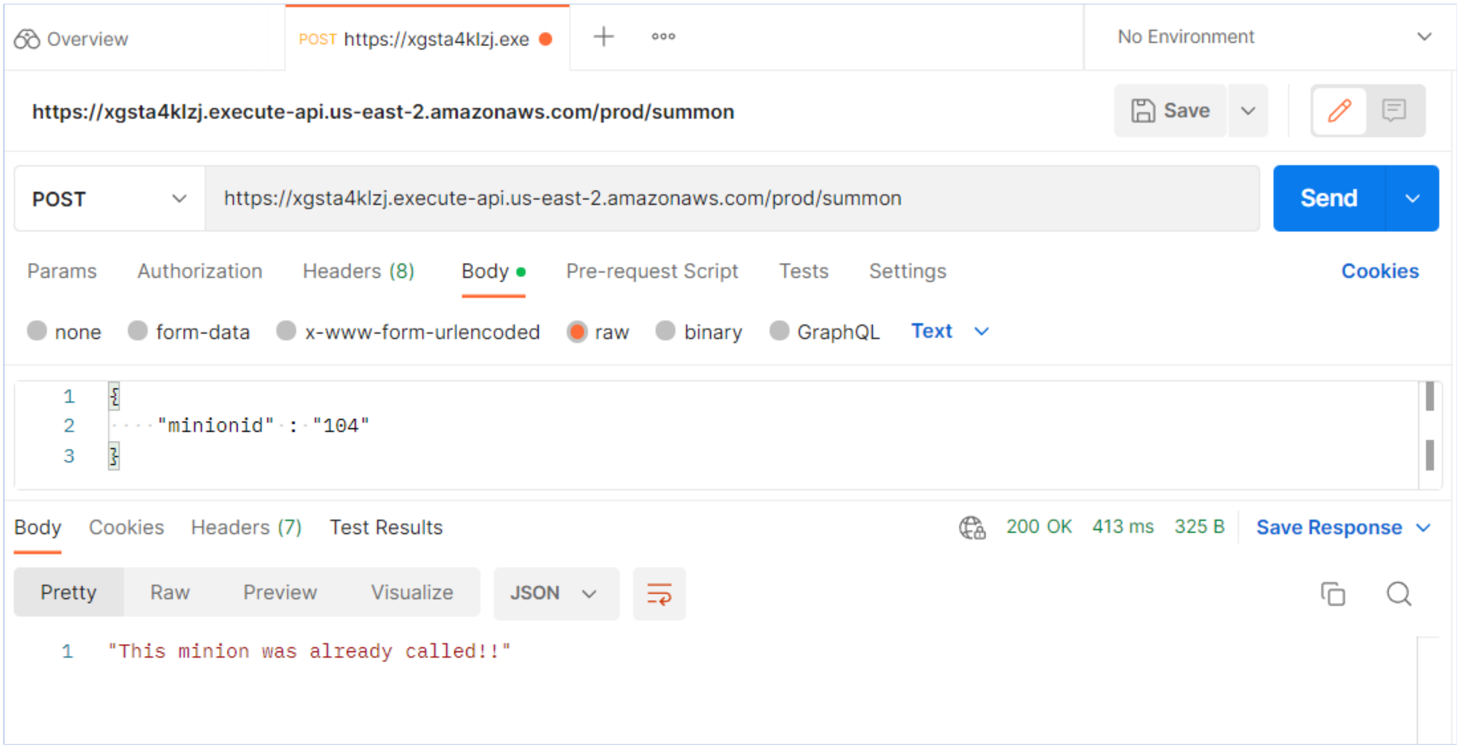
POST https://xgsta4klzj.execute-api.us-east-2.amazonaws.com/prod/summon

```
{
    "minionid" : "104"
}
```

200 OK 411 ms 491 B

```
{
    "statusCode": 200,
    "body": {
        "minionid": "104",
        "name": "alice",
        "color": "purple",
        "age": 12,
        "minionStatus": 1
    },
    "headers": {
        "Content-Type": "application/json",
        "Access-Control-Allow-Origin": "*"
    }
}
```

Below is the screenshot for which the minion with specific minionid is called again and it throws message saying “This minion was already called”.



https://xgsta4klzj.execute-api.us-east-2.amazonaws.com/prod/summon

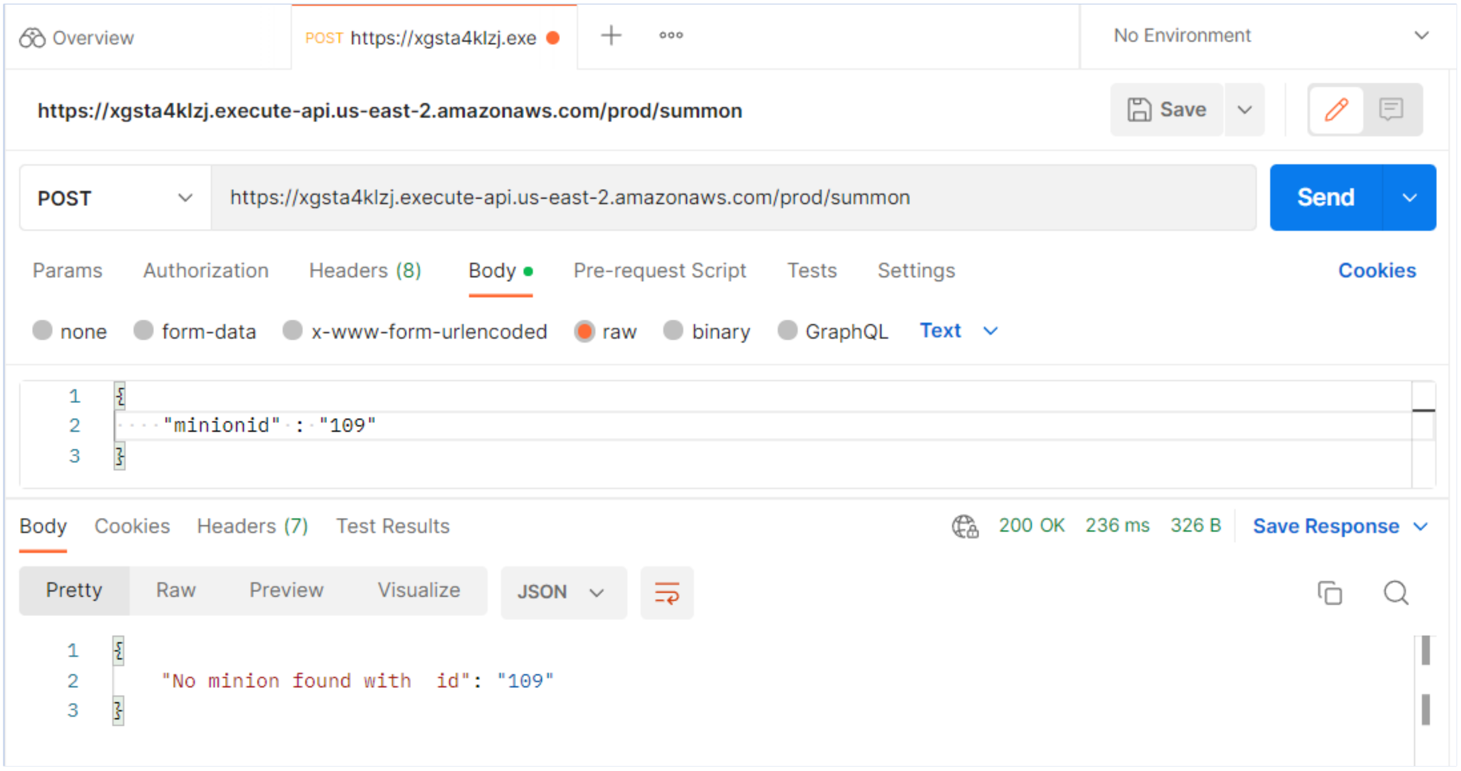
POST https://xgsta4klzj.execute-api.us-east-2.amazonaws.com/prod/summon

```
{
    "minionid" : "104"
}
```

200 OK 413 ms 325 B

```
"This minion was already called!!"
```

Below is the screenshot for which the minion with specific minionid is called that is not in DynamoDB table “Minion”.



https://xgsta4klzj.execute-api.us-east-2.amazonaws.com/prod/summon

POST https://xgsta4klzj.execute-api.us-east-2.amazonaws.com/prod/summon

```
{
    "minionid" : "109"
}
```

200 OK 236 ms 326 B

```
{
    "No minion found with  id": "109"
}
```