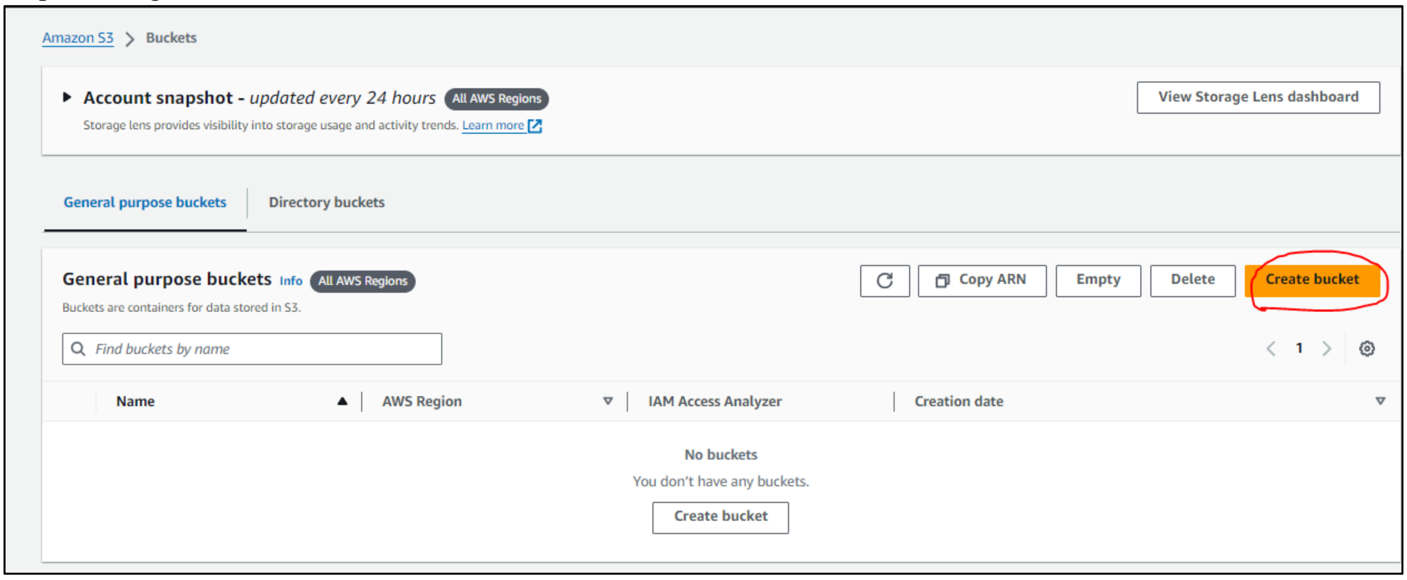
Documentation for Cloud Formation Template (Using Lambda – API – IAM – DynamoDB)

PART – 1

Step 1: Log-in to your AWS console account
Step 2: Navigate to S3 services and create bucket.



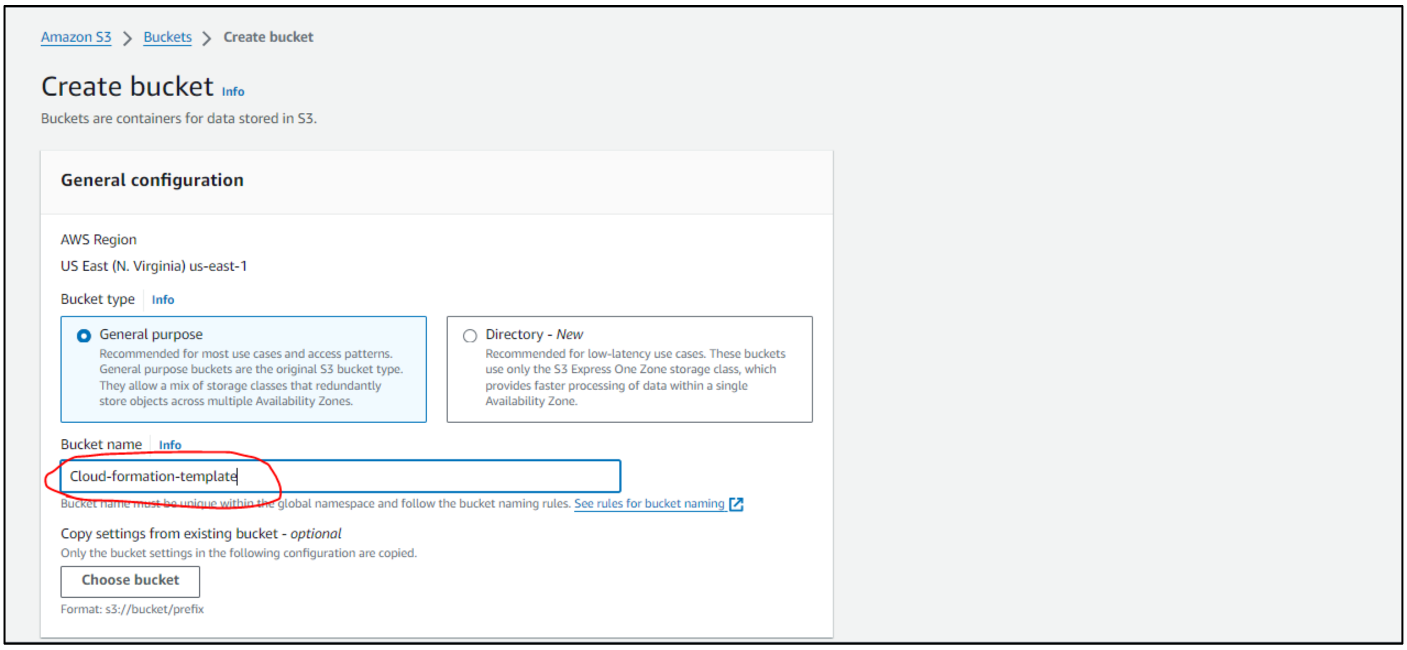
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets Info All AWS Regions

Buckets are containers for data stored in S3.

No buckets

You don't have any buckets.



Create bucket

Buckets are containers for data stored in S3.

AWS Region
US East (N. Virginia) us-east-1

Bucket type

(●) General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

( ) Directory - *New*
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name
Cloud-formation-template
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

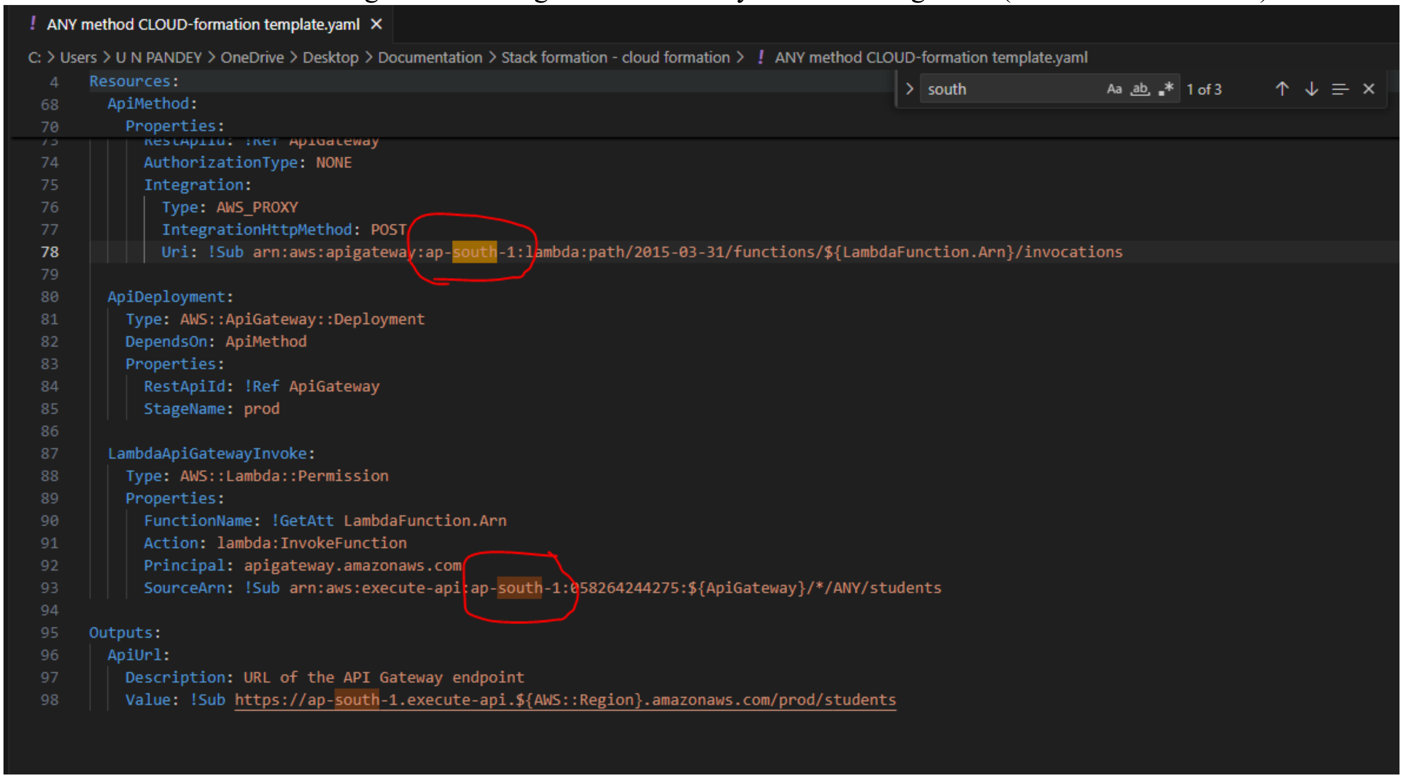
Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
Format: s3://bucket/prefix

Remember the bucket name (eg. Cloud-formation-template) and scroll down and click on create bucket.

Step 3: Create a LAMBDA file using python scripting to execute the CRUD operation on invoking of API URL. Sample python code is shared in the repository named as index.py
Step 4: After creation of index.py, covert it into zip file and upload it in S3 bucket that we have created in Step – 2.
Step 5: Create a YAML file for the cloud formation template where services like Lambda-API-IAM-DynamoDB will be created.
Step 6: You can write a new yaml file or You can use the Sample YAML file which is given in the . repository named as (ANY method CLOUD-formation template).

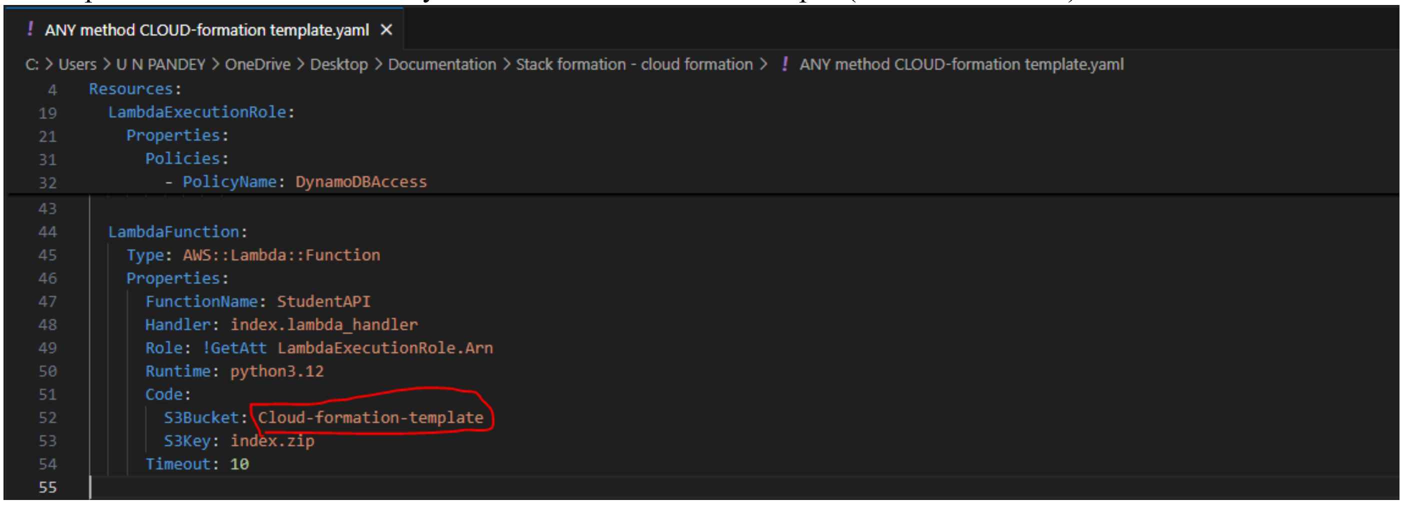
Step 7: While creating the YAML file or using the sample, note the changes you have to make.

A – Make sure to change/write the region from where you are creating stack (on 78 93 line of code)


! ANY method CLOUD-formation template.yaml ×
C: > Users > U N PANDEY > OneDrive > Desktop > Documentation > Stack formation - cloud formation > ! ANY method CLOUD-formation template.yaml
south
1 of 3

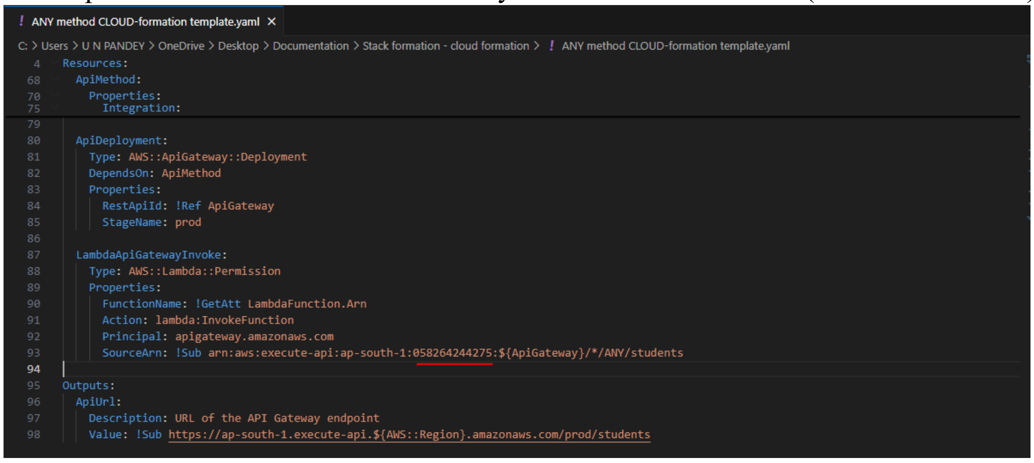
```
4   Resources:
68    ApiMethod:
70      Properties:
73        RestApiId: !Ref ApiGateway
74        AuthorizationType: NONE
75        Integration:
76          Type: AWS_PROXY
77          IntegrationHttpMethod: POST
78          Uri: !Sub arn:aws:apigateway:ap-south-1:lambda:path/2015-03-31/functions/${LambdaFunction.Arn}/invocations
79
80    ApiDeployment:
81      Type: AWS::ApiGateway::Deployment
82      DependsOn: ApiMethod
83      Properties:
84        RestApiId: !Ref ApiGateway
85        StageName: prod
86
87    LambdaApiGatewayInvoke:
88      Type: AWS::Lambda::Permission
89      Properties:
90        FunctionName: !GetAtt LambdaFunction.Arn
91        Action: lambda:InvokeFunction
92        Principal: apigateway.amazonaws.com
93        SourceArn: !Sub arn:aws:execute-api:ap-south-1:058264244275:${ApiGateway}/*/ANY/students
94
95  Outputs:
96    ApiUrl:
97      Description: URL of the API Gateway endpoint
98      Value: !Sub https://ap-south-1.execute-api.${AWS::Region}.amazonaws.com/prod/students
```

B – Replace the S3 bucket name to your created bucket name in Step-2 (on 52 line of code)


! ANY method CLOUD-formation template.yaml ×
C: > Users > U N PANDEY > OneDrive > Desktop > Documentation > Stack formation - cloud formation > ! ANY method CLOUD-formation template.yaml

```
4   Resources:
19    LambdaExecutionRole:
21      Properties:
31        Policies:
32          - PolicyName: DynamoDBAccess
43
44    LambdaFunction:
45      Type: AWS::Lambda::Function
46      Properties:
47        FunctionName: StudentAPI
48        Handler: index.lambda_handler
49        Role: !GetAtt LambdaExecutionRole.Arn
50        Runtime: python3.12
51        Code:
52          S3Bucket: Cloud-formation-template
53          S3Key: index.zip
54        Timeout: 10
55
```

C – Replace the Account ID number with your Account ID number. (on 93 line of code)


! ANY method CLOUD-formation template.yaml ×
C: > Users > U N PANDEY > OneDrive > Desktop > Documentation > Stack formation - cloud formation > ! ANY method CLOUD-formation template.yaml

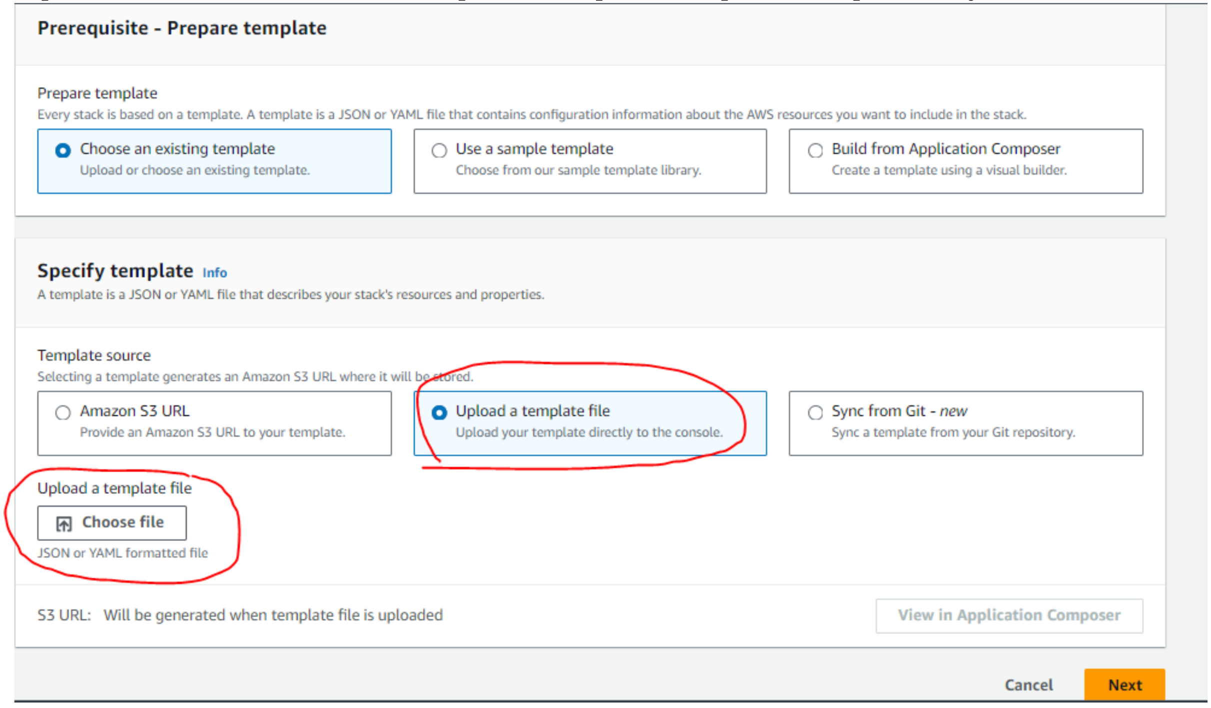
```
4   Resources:
68    ApiMethod:
70      Properties:
75        Integration:
79
80    ApiDeployment:
81      Type: AWS::ApiGateway::Deployment
82      DependsOn: ApiMethod
83      Properties:
84        RestApiId: !Ref ApiGateway
85        StageName: prod
86
87    LambdaApiGatewayInvoke:
88      Type: AWS::Lambda::Permission
89      Properties:
90        FunctionName: !GetAtt LambdaFunction.Arn
91        Action: lambda:InvokeFunction
92        Principal: apigateway.amazonaws.com
93        SourceArn: !Sub arn:aws:execute-api:ap-south-1:058264244275:${ApiGateway}/*/ANY/students
94
95  Outputs:
96    ApiUrl:
97      Description: URL of the API Gateway endpoint
98      Value: !Sub https://ap-south-1.execute-api.${AWS::Region}.amazonaws.com/prod/students
```

Till here we have our Yaml file and Index.py (lambda python file ready). Next, we will go to the creation of cloud-formation using our Yaml file and index.py.

PART – 2

Step 1: Come back to the AWS console login Screen and Navigate to Cloud-formation service.

Step 2: Create a Stack and select the Upload a template file option and upload the yaml file which we created earlier.



Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

[✓ Choose an existing template] Upload or choose an existing template. | Use a sample template — Choose from our sample template library. | Build from Application Composer — Create a template using a visual builder.

Specify template Info

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL — Provide an Amazon S3 URL to your template. | [✓ Upload a template file] Upload your template directly to the console. | Sync from Git - *new* — Sync a template from your Git repository.

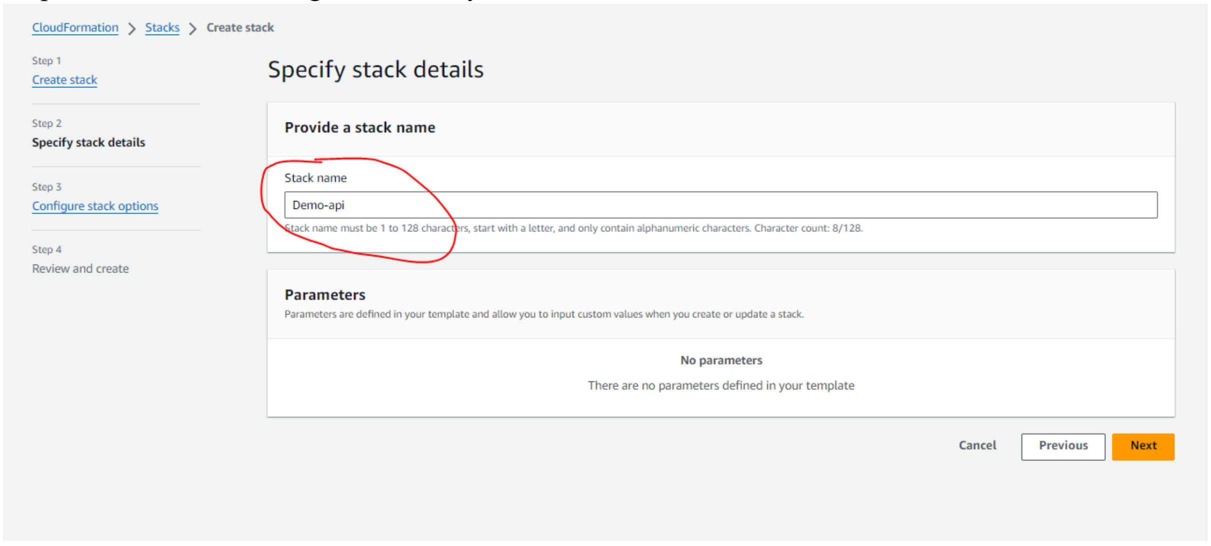
Upload a template file

Choose file

JSON or YAML formatted file

S3 URL: Will be generated when template file is uploaded

Step 3: Click on next and give name to your Stack and click on next.



Specify stack details

Provide a stack name

Stack name: Demo-api

Stack name must be 1 to 128 characters, start with a letter, and only contain alphanumeric characters. Character count: 8/128.

Parameters

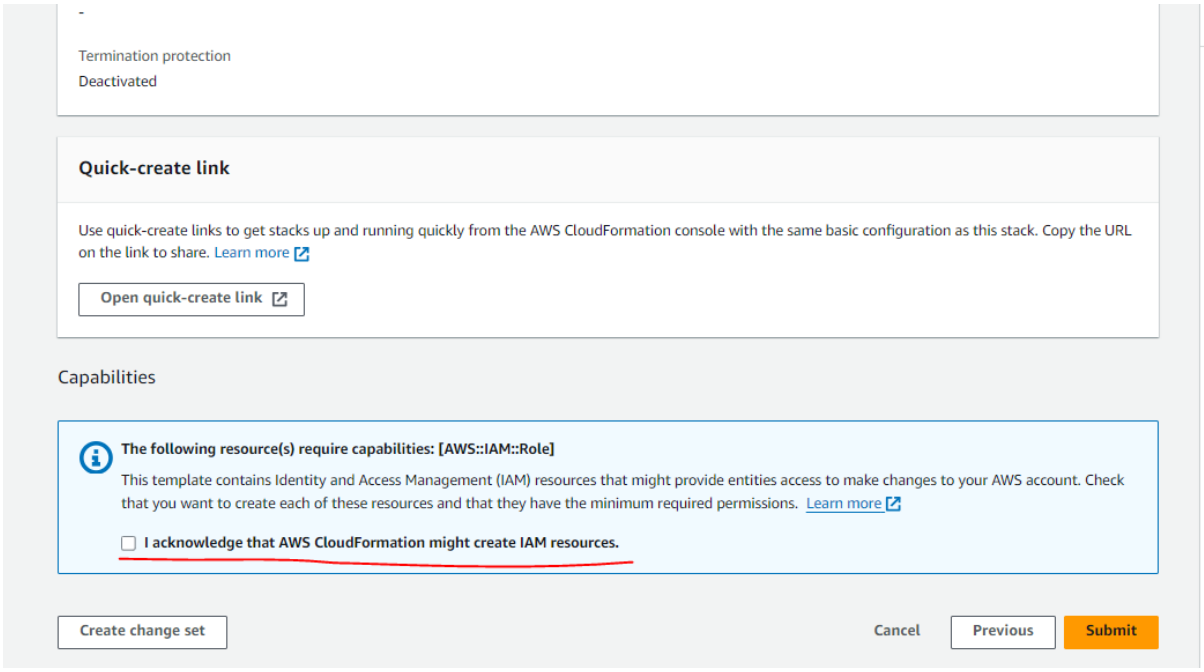
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

No parameters

There are no parameters defined in your template

Step 4: Don't change any configuration on next stage. Just scroll down and click on next.

Step 5: On last review page tick the checkbox (**I acknowledge that AWS CloudFormation might create IAM resources.)**


-

Termination protection
Deactivated

Quick-create link

Use quick-create links to get stacks up and running quickly from the AWS CloudFormation console with the same basic configuration as this stack. Copy the URL on the link to share. Learn more

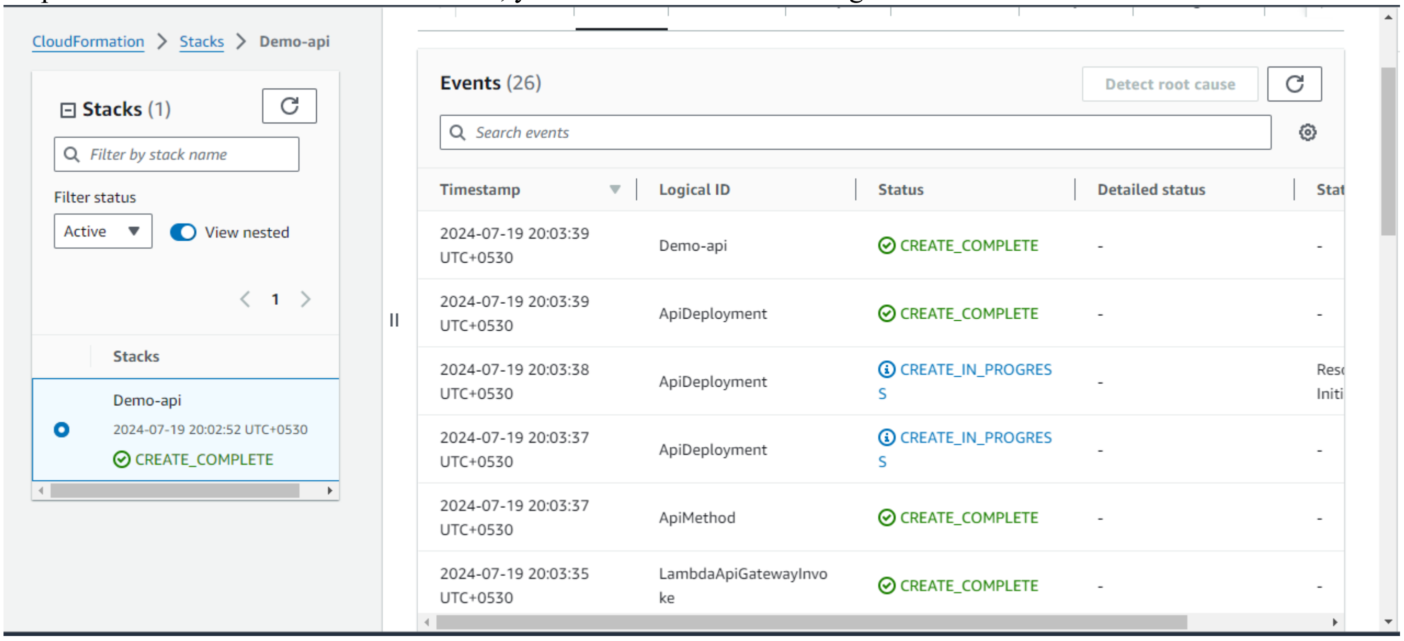
Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. Learn more

☐ **I acknowledge that AWS CloudFormation might create IAM resources.**

Step 6: Click on submit and wait for creation of stack.
Step 7: If any error occurs then go through every step from part-1 and part-2.
Step 8: After successful creation of Stacks, you will come across below given screen.



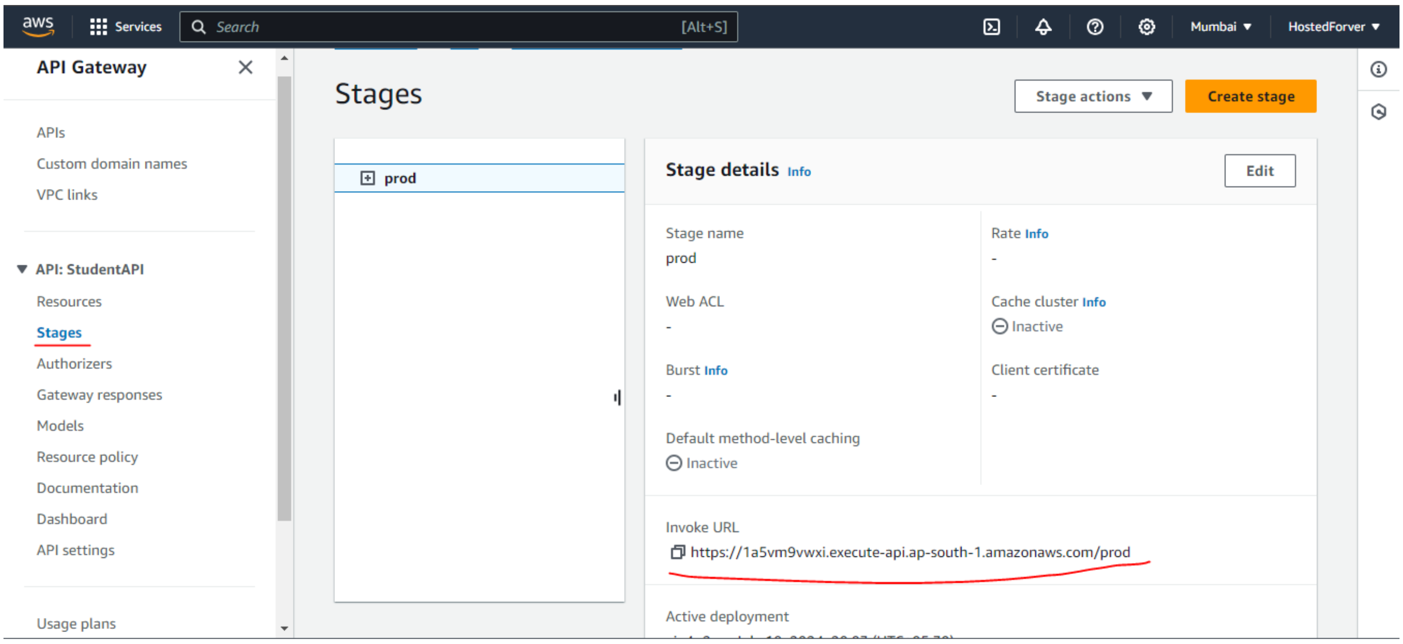
Demo-api
2024-07-19 20:02:52 UTC+0530
CREATE_COMPLETE

Events (26)

Timestamp	Logical ID	Status	Detailed status	Stat
2024-07-19 20:03:39 UTC+0530	Demo-api	CREATE_COMPLETE	-	-
2024-07-19 20:03:39 UTC+0530	ApiDeployment	CREATE_COMPLETE	-	-
2024-07-19 20:03:38 UTC+0530	ApiDeployment	CREATE_IN_PROGRESS	-	Res Initi
2024-07-19 20:03:37 UTC+0530	ApiDeployment	CREATE_IN_PROGRESS	-	-
2024-07-19 20:03:37 UTC+0530	ApiMethod	CREATE_COMPLETE	-	-
2024-07-19 20:03:35 UTC+0530	LambdaApiGatewayInvoke	CREATE_COMPLETE	-	-

Step 9: After creation of stacks navigate to every service (Lambda – API – DynamoDB – IAM) and verify that it is created.
Step 10: After verifying, navigate and go to API Service and you will see StudentAPI is created, click on it.

Step 11: After that you will redirected to next screen from which you have to navigate and go to stages under API:StudentAPI menu in left-side on the screen and after clicking on resource, on the main screen scroll down and copy the Invoke URL as shown below.



Stage name
prod

Rate Info
-

Web ACL
-

Cache cluster Info
Inactive

Burst Info
-

Client certificate
-

Default method-level caching
Inactive

Invoke URL
https://1a5vm9vwxi.execute-api.ap-south-1.amazonaws.com/prod

Active deployment

Step 12: Till here we have successfully created a stack in which we have created services like
API (ANY method)-Lambda-DynamoDB-IAM and also deployed the API.
It’s time to do the CRUD (Create-Read-Update-Delete) operations on DynamoDB table which we have created.

PART – 3

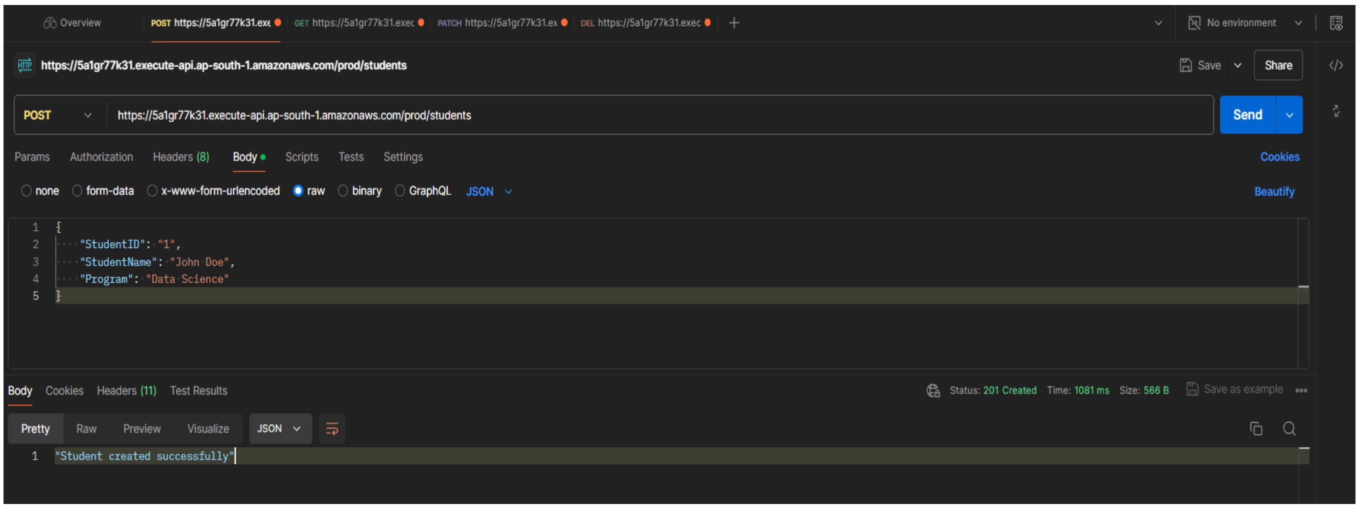
Step 1: After copying the Invoke URL in above Step-11, open POSTMAN console and click on new request.
We will divide the CRUD operation in 4 step (4 methods):
Create (POST), Read (GET), Update (PUT), Delete (DELETE)

Step A : Test POST method (Create a student):

- Set the HTTP method to POST
- Paste the API Invoke URL (ref to Step-11) and add /students at the last of the URL.
- In the "Body" tab, select "raw" and choose "JSON" format
- Enter the following JSON:

```
{
        "StudentID" : "1",
        "StudentName" : "Jeff Bezos",
        "Department" : "Bachelor's"
}
```

(Note : You can add any details in the value column and add any number of student of you want)



https://5a1gr77k31.execute-api.ap-south-1.amazonaws.com/prod/students

POST https://5a1gr77k31.execute-api.ap-south-1.amazonaws.com/prod/students

```
{
    "StudentID": "1",
    "StudentName": "John Doe",
    "Program": "Data Science"
}
```

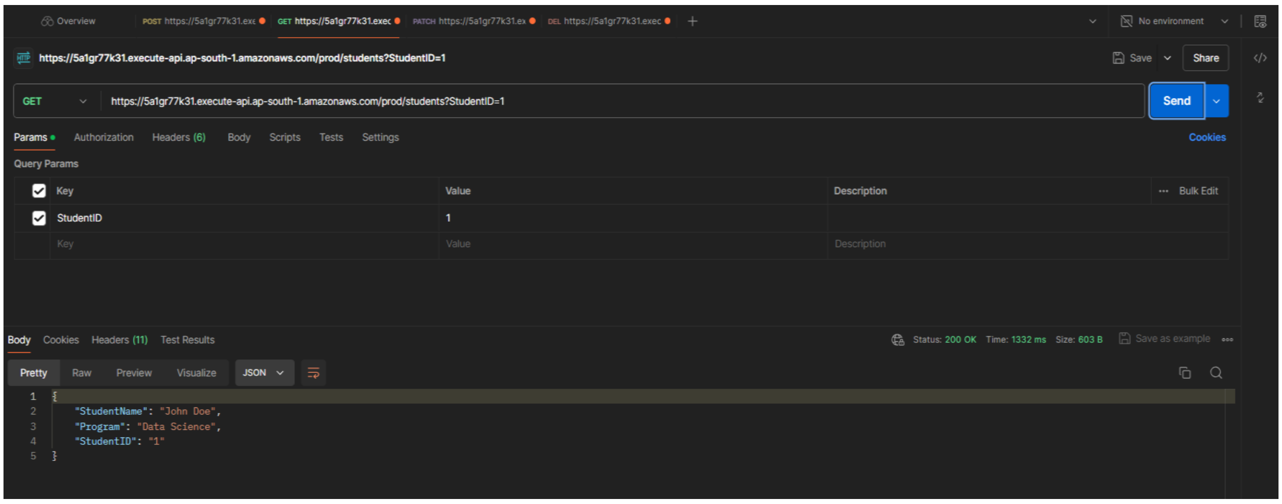
Status: 201 Created Time: 1081 ms Size: 566 B

```
"Student created successfully"
```

Click on send request and you will see the following: "Students created successfully" in the below screen.

Step B : Test GET method (Retrieve a student):

- Create a new request in Postman
- Set the HTTP method to GET
- Paste the API Invoke URL
- In the "Params" tab, add a key "StudentID" with value "1"
- Click on send request and you will the details of student that we created in above Step (POST)



https://5a1gr77k31.execute-api.ap-south-1.amazonaws.com/prod/students?StudentID=1

GET https://5a1gr77k31.execute-api.ap-south-1.amazonaws.com/prod/students?StudentID=1

Query Params

Key	Value	Description
StudentID	1	

Status: 200 OK Time: 1332 ms Size: 603 B

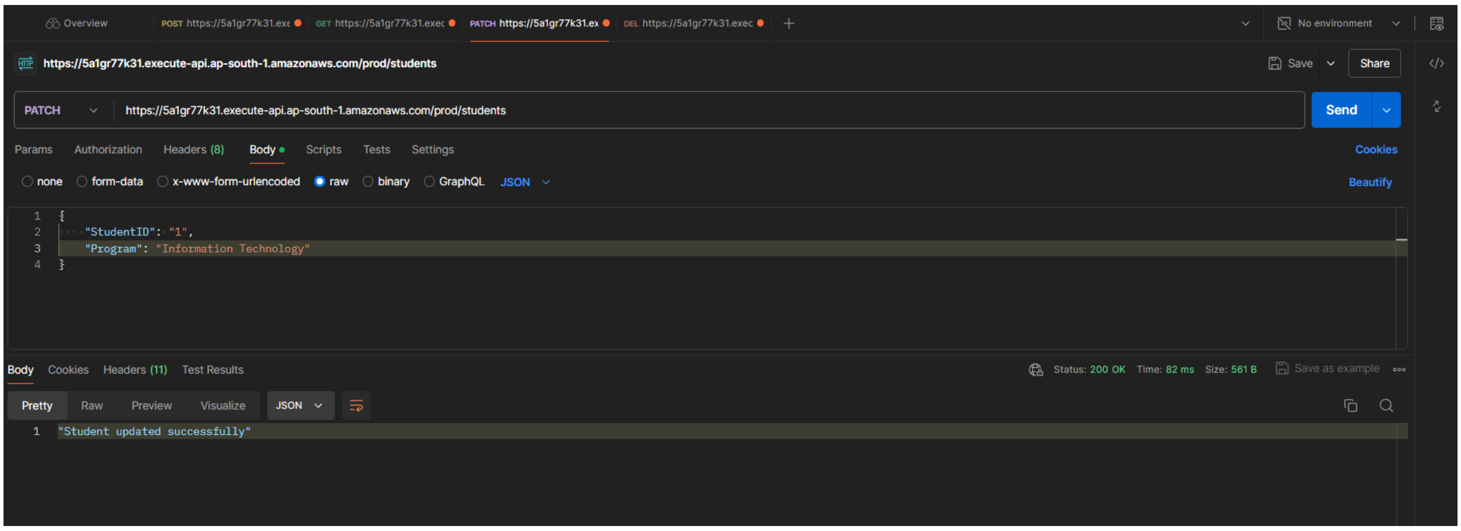
```
{
    "StudentName": "John Doe",
    "Program": "Data Science",
    "StudentID": "1"
}
```

Step C : Test PATCH method (Update a student):

- Create a new request in Postman
- Set the HTTP method to PATCH
- Paste the API Invoke URL
- In the "Body" tab, select "raw" and choose "JSON" format
- Enter the following JSON:

```
{
    “StudentID” : “1”,
    “StudentName” : “Jeff Bezos”,
    “Department” : “Master’s”
}
```

(Note : You can update to any value, for example I have updated the Department value)



PATCH https://5a1gr77k31.execute-api.ap-south-1.amazonaws.com/prod/students

```
{
    "StudentID": "1",
    "Program": "Information Technology"
}
```

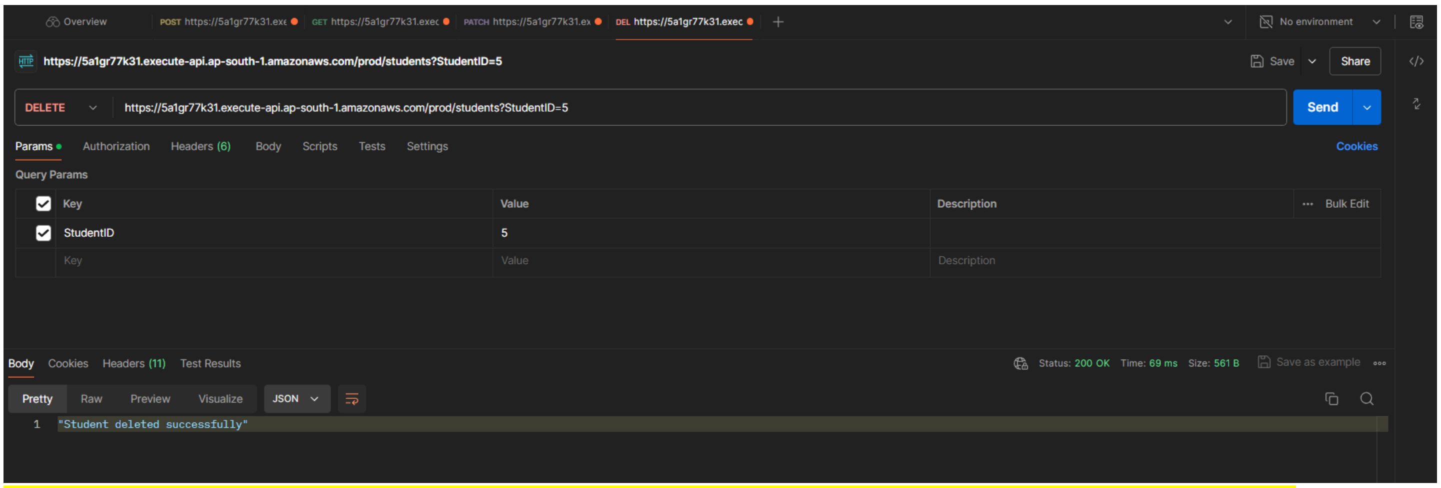
Status: 200 OK Time: 82 ms Size: 561 B

```
"Student updated successfully"
```

Click on send request and you will see the following: “Student updated successfully”

Step D : Test DELETE method (Delete a student):

- Create a new request in Postman
- Set the HTTP method to DELETE
- Paste the API Invoke URL
- In the "Params" tab, add a key "StudentID" with value "1"
- Click on send request and you will see the following: “Student deleted successfully”



DELETE https://5a1gr77k31.execute-api.ap-south-1.amazonaws.com/prod/students?StudentID=5

Query Params

	Key	Value	Description
✓	StudentID	5	

Status: 200 OK Time: 69 ms Size: 561 B

```
"Student deleted successfully"
```

Step E: Repeat the STEP B to verify that delete method is successfully executed, if it is successfully executed than you will see the following output: “Student not found”

By all the steps (PART 1, PART 2, PART 3) followed correctly, we have successfully created the stack using yaml file and performed CRUD operations using POSTMAN.