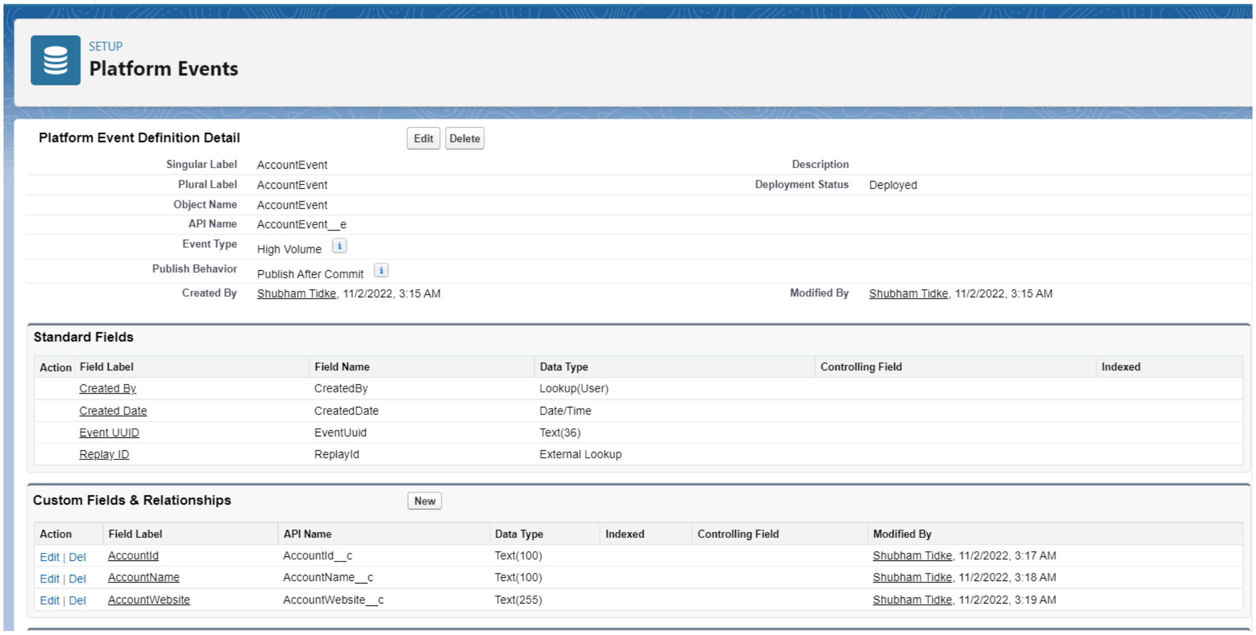
AWS - Salesforce Integration

1. SALESFORCE to AWS [Using Lambda function]

Use case: Whenever we create an account record in salesforce, add the record details in the AWS Dynamodb table.

i. Create a Platform event in your salesforce org

Platform events are based on event-driven architecture and are used to deliver notifications in a secure, scalable way within salesforce or external apps. To create an event when an account gets created, we create a platform event on Account sObject with custom fields that we want to store in the AWS dynamodb table,



Platform Event Definition Detail

Singular Label	AccountEvent	Description	
Plural Label	AccountEvent	Deployment Status	Deployed
Object Name	AccountEvent		
API Name	AccountEvent__e		
Event Type	High Volume		
Publish Behavior	Publish After Commit		
Created By	Shubham Tidke, 11/2/2022, 3:15 AM	Modified By	Shubham Tidke, 11/2/2022, 3:15 AM

Standard Fields

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		
	Created Date	CreatedDate	Date/Time		
	Event UUID	EventUuid	Text(36)		
	Replay ID	ReplayId	External Lookup		

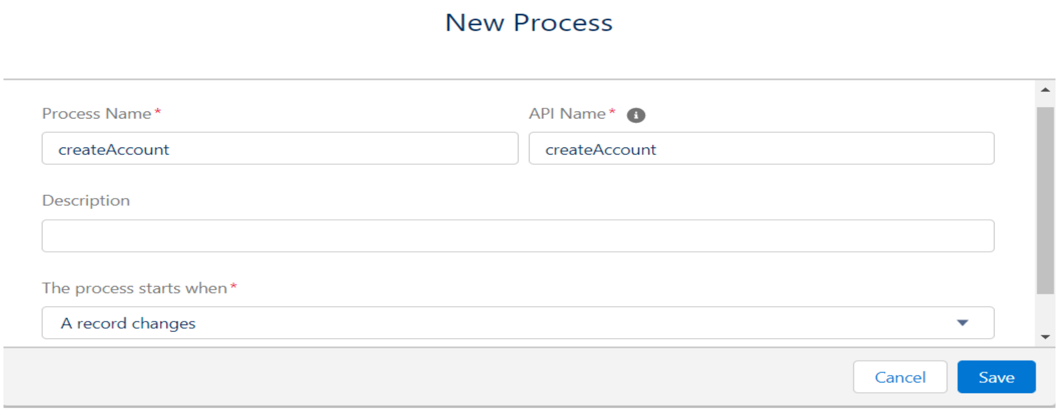
Custom Fields & Relationships

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit | Del	AccountId	AccountId__c	Text(100)			Shubham Tidke, 11/2/2022, 3:17 AM
Edit | Del	AccountName	AccountName__c	Text(100)			Shubham Tidke, 11/2/2022, 3:18 AM
Edit | Del	AccountWebsite	AccountWebsite__c	Text(255)			Shubham Tidke, 11/2/2022, 3:19 AM

ii. Process Builder to publish the event

Once the platform event is defined We need to publish the event once an account is created. Platform events can be published in various ways. Here, we are using a process builder.

Create a process builder on the platform event



New Process

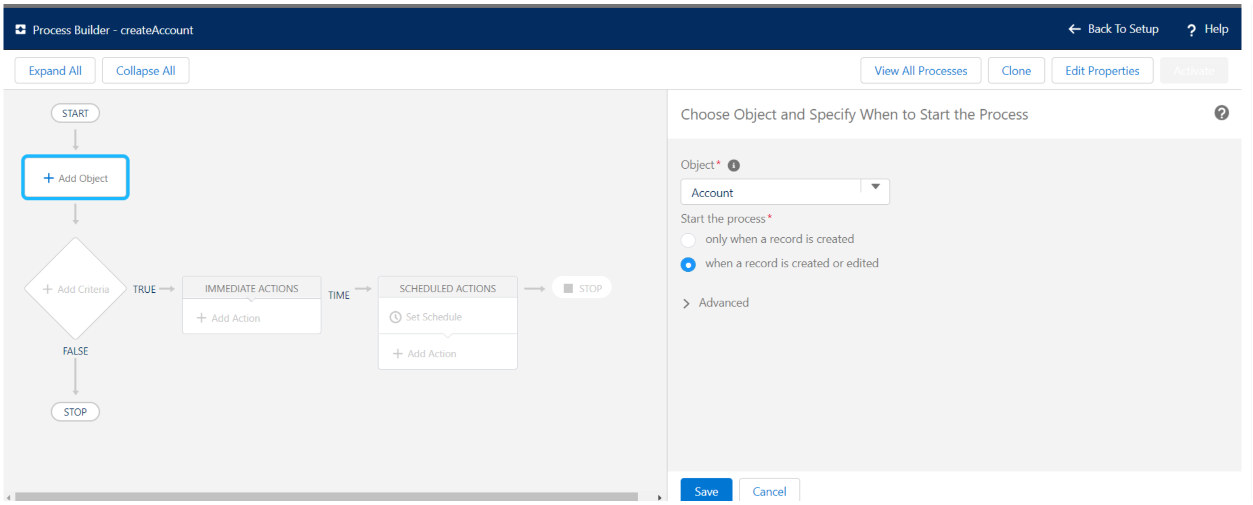
Process Name*: createAccount

API Name*: createAccount

Description:

The process starts when*: A record changes

Select Account as Object,



Process Builder - createAccount

Choose Object and Specify When to Start the Process

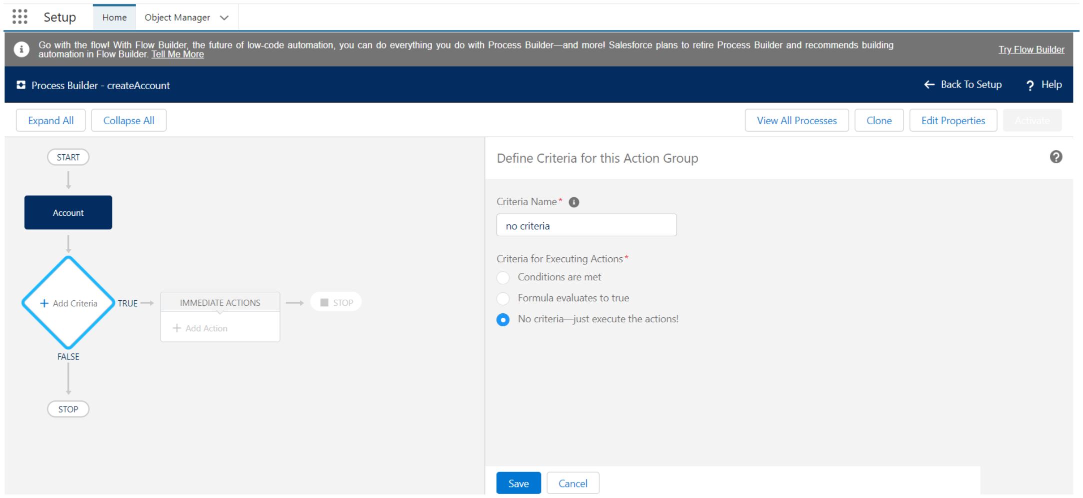
Object*: Account

Start the process*

○ only when a record is created

● when a record is created or edited

Select ‘No criteria’ in criteria section



Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Process Builder—and more! Salesforce plans to retire Process Builder and recommends building automation in Flow Builder. Tell Me More

Process Builder - createAccount

Define Criteria for this Action Group

Criteria Name*: no criteria

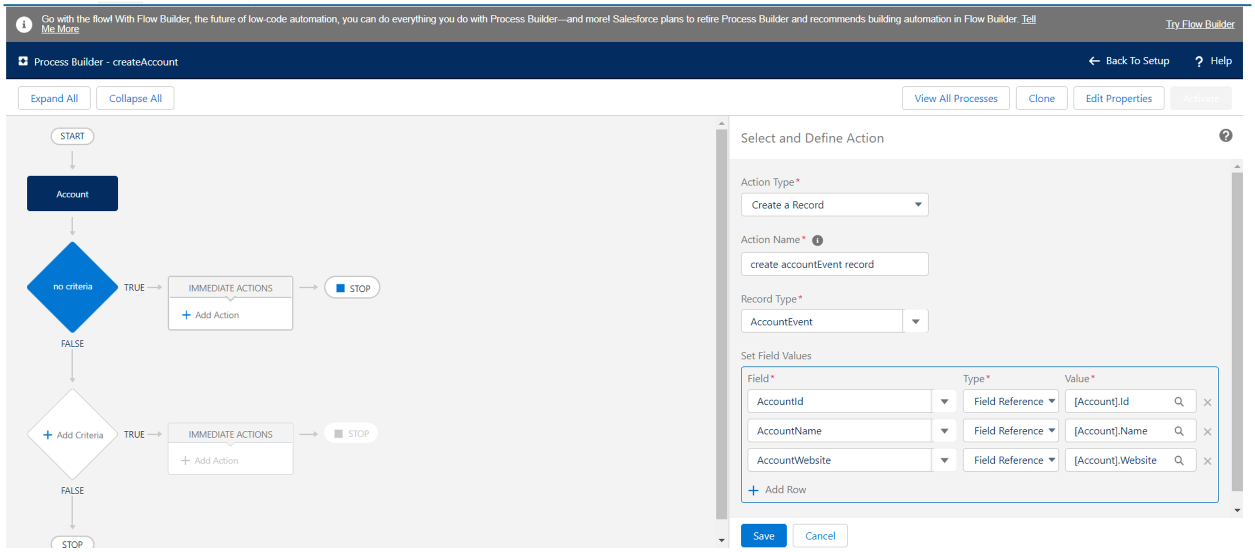
Criteria for Executing Actions*

○ Conditions are met

○ Formula evaluates to true

● No criteria—just execute the actions!

Add immediate action of type ‘create a record’ on record type ‘AccountEvent’ and map platform event fields with account object fields


Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Process Builder—and more! Salesforce plans to retire Process Builder and recommends building automation in Flow Builder. Tell Me More

Process Builder - createAccount

Select and Define Action

Action Type* Create a Record

Action Name* create accountEvent record

Record Type* AccountEvent

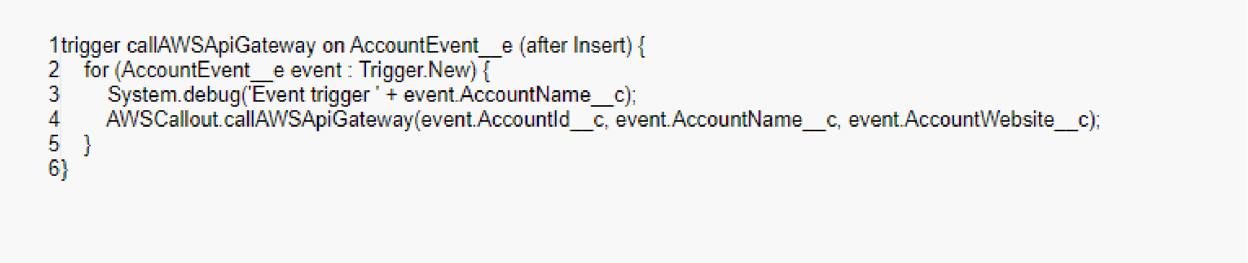
Set Field Values

Field*	Type*	Value*
AccountId	Field Reference	[Account].Id
AccountName	Field Reference	[Account].Name
AccountWebsite	Field Reference	[Account].Website

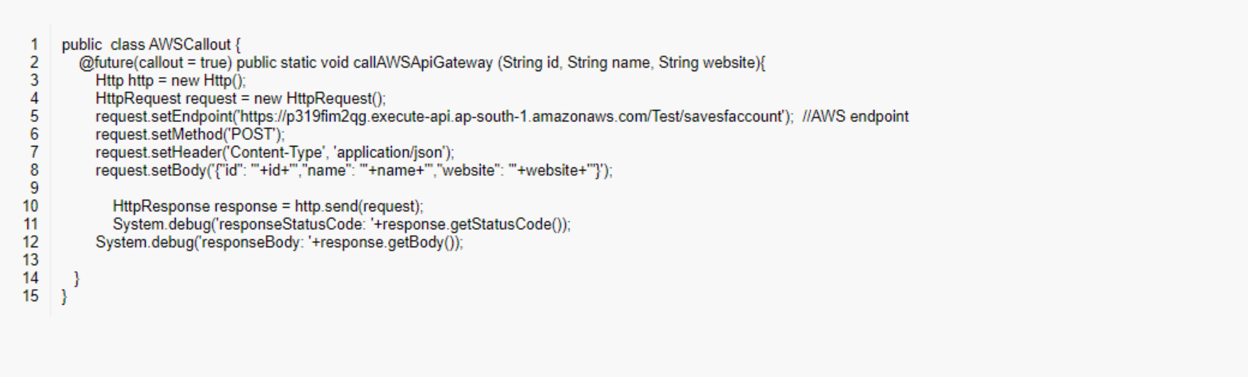
Activate the process builder once done.

iii. Trigger to subscribe to the event

Now as the event is published we need to subscribe to it and here since we will be updating the dynamodb table, we will have an HTTP call on the AWS API gateway endpoint using a trigger.


```
trigger callAWSApiGateway on AccountEvent__e (after Insert) {
  for (AccountEvent__e event : Trigger.New) {
     System.debug('Event trigger ' + event.AccountName__c);
     AWSCallout.callAWSApiGateway(event.AccountId__c, event.AccountName__c, event.AccountWebsite__c);
  }
}
```

Sending a POST request to the AWS endpoint.
[Add AWS endpoint in remote site settings.
AWS endpoint can be found on AWS API Gateway]

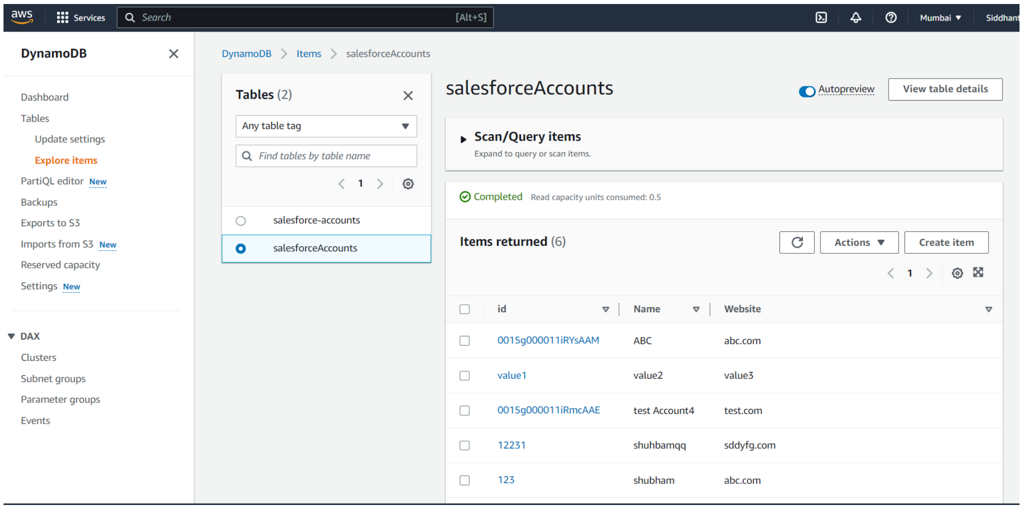


```
public  class AWSCallout {
   @future(callout = true) public static void callAWSApiGateway (String id, String name, String website){
      Http http = new Http();
      HttpRequest request = new HttpRequest();
      request.setEndpoint('https://p319fim2qg.execute-api.ap-south-1.amazonaws.com/Test/savesfaccount');  //AWS endpoint
      request.setMethod('POST');
      request.setHeader('Content-Type', 'application/json');
      request.setBody('{"id": "'+id+'","name": "'+name+'","website": "'+website+'"}');

         HttpResponse response = http.send(request);
         System.debug('responseStatusCode: '+response.getStatusCode());
      System.debug('responseBody: '+response.getBody());

   }
}
```

This conclude the Salesforce side configuration.

In AWS,

iv. Create Dynamodb tables
Search dynamodb in the search bar and select DynamoDB.
Create a table with fields [id, name, website]



salesforceAccounts

✓ Completed Read capacity units consumed: 0.5

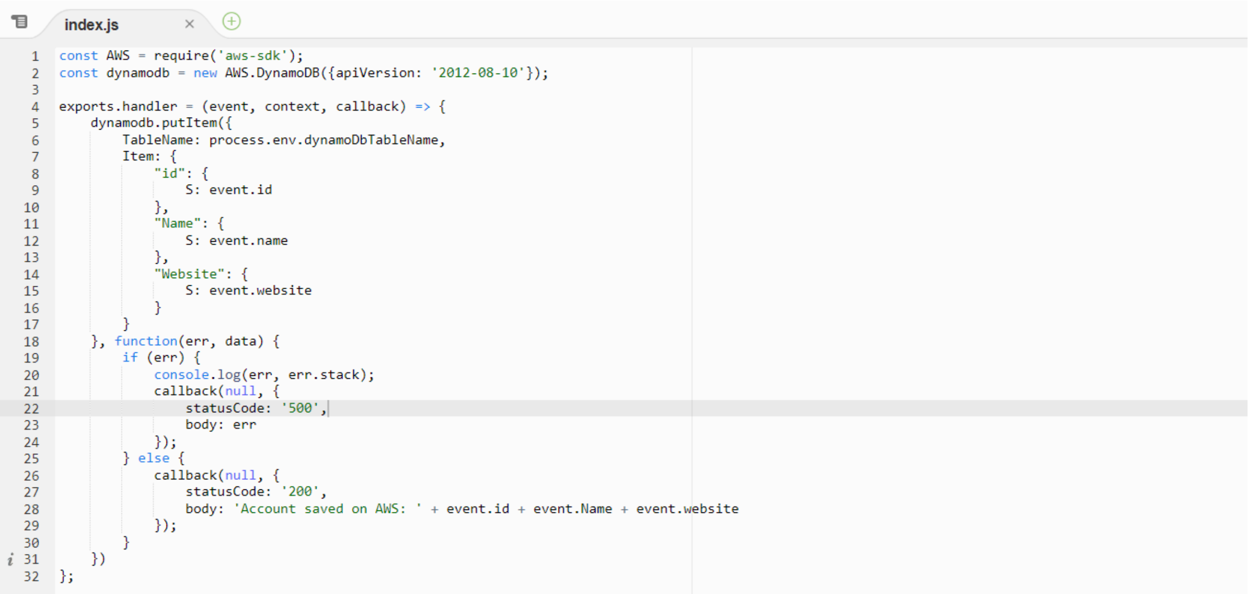
Items returned (6)

□	id	Name	Website
□	0015g000011iRYsAAM	ABC	abc.com
□	value1	value2	value3
□	0015g000011iRmcAAE	test Account4	test.com
□	12231	shuhbamqq	sddyfg.com
□	123	shubham	abc.com

v. Create a Lambda function

Search Lambda in the search bar and click on ‘create function’, Give the name for the lambda function
Under ‘**Change default execution role**’, Select Create a new role from AWS policy templates Choose ‘**Simple microservice Permission**’.

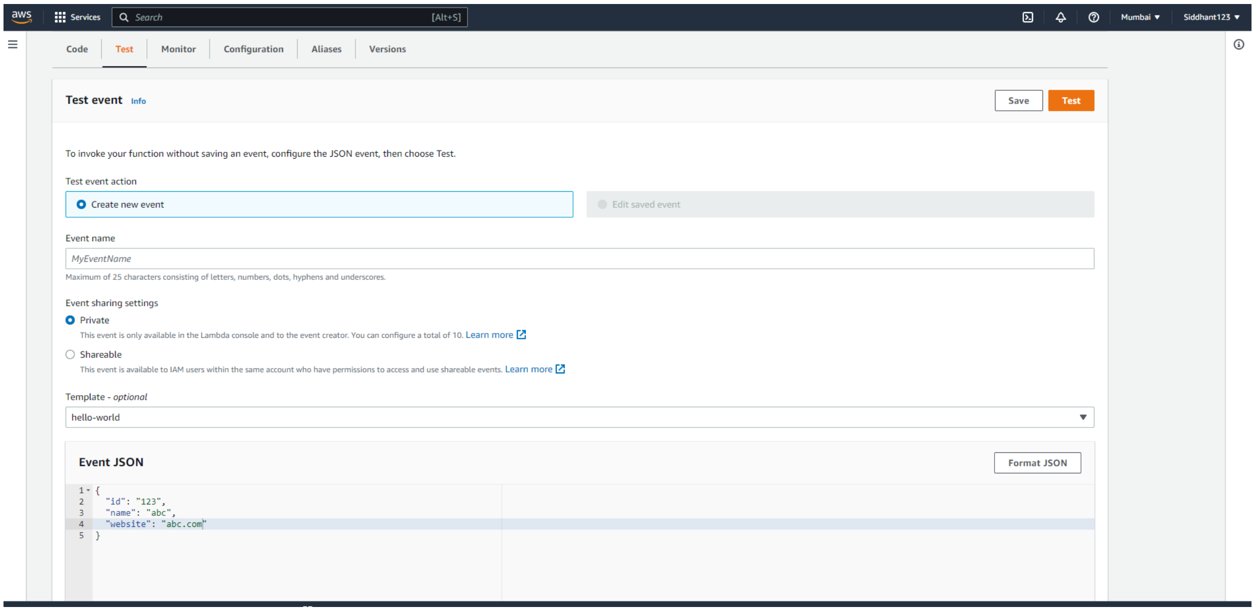
This permission allows the Lambda function to access dynamodb.



```javascript
const AWS = require('aws-sdk');
const dynamodb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

exports.handler = (event, context, callback) => {
    dynamodb.putItem({
        TableName: process.env.dynamoDbTableName,
        Item: {
            "id": {
                S: event.id
            },
            "Name": {
                S: event.name
            },
            "Website": {
                S: event.website
            }
        }
    }, function(err, data) {
        if (err) {
            console.log(err, err.stack);
            callback(null, {
                statusCode: '500',
                body: err
            });
        } else {
            callback(null, {
                statusCode: '200',
                body: 'Account saved on AWS: ' + event.id + event.Name + event.website
            });
        }
    })
};
```

TableName will be the dynamoDB Table Name

Test your Lambda function by creating a test event. Update the event JSON key-value pairs and click test,



Test event

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

● Create new event ○ Edit saved event

Event name

MyEventName

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

● Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

○ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

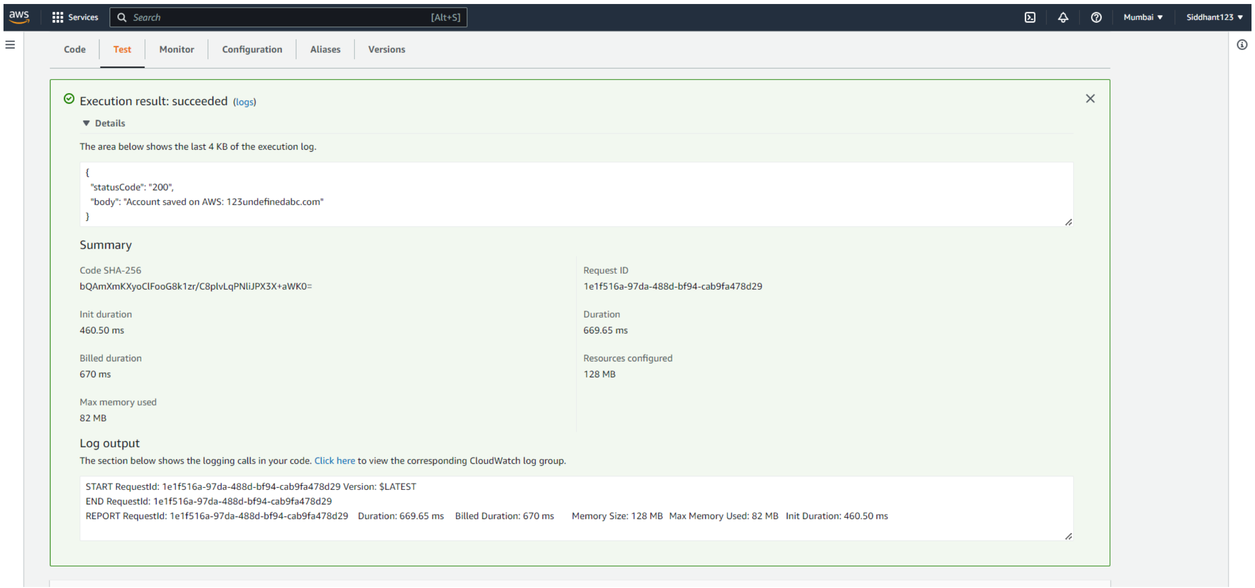
Template - *optional*

hello-world ▾

Event JSON

```json
{
  "id": "123",
  "name": "abc",
  "website": "abc.com"
}
```

On the successful result, The event log would show status code 200,



Execution result: succeeded (logs)

▼ Details

The area below shows the last 4 KB of the execution log.

```
{
  "statusCode": "200",
  "body": "Account saved on AWS: 123undefinedabc.com"
}
```

Summary

Code SHA-256
bQAmXmKXyoClFooG8k1zr/C8plvLqPNliJPX3X+aWK0=

Request ID
1e1f516a-97da-488d-bf94-cab9fa478d29

Init duration
460.50 ms

Duration
669.65 ms

Billed duration
670 ms

Resources configured
128 MB

Max memory used
82 MB

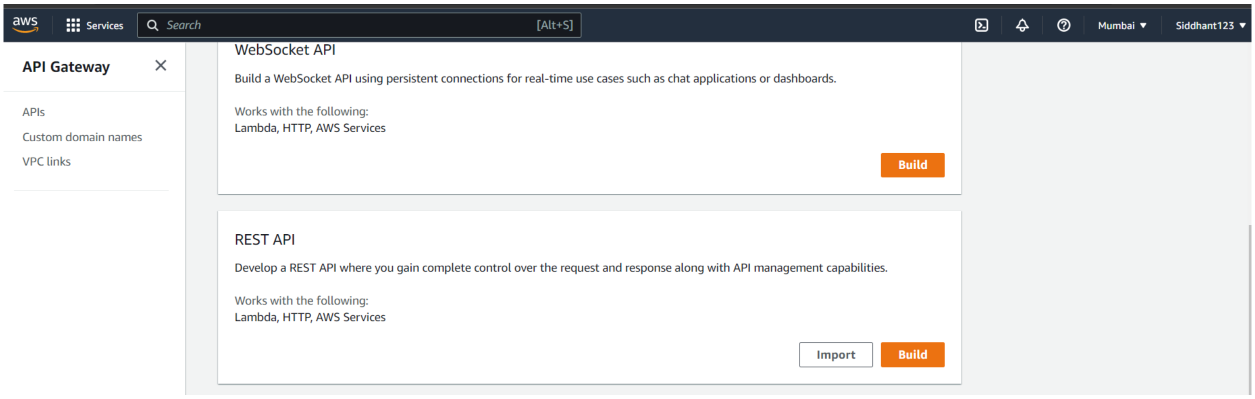
Log output

The section below shows the logging calls in your code. Click here to view the corresponding CloudWatch log group.

```
START RequestId: 1e1f516a-97da-488d-bf94-cab9fa478d29 Version: $LATEST
END RequestId: 1e1f516a-97da-488d-bf94-cab9fa478d29
REPORT RequestId: 1e1f516a-97da-488d-bf94-cab9fa478d29	Duration: 669.65 ms	Billed Duration: 670 ms	Memory Size: 128 MB	Max Memory Used: 82 MB	Init Duration: 460.50 ms
```

vi. Create API Gateway
To receive requests from salesforce and process them in dynamodb

Search API Gateway in the search bar and create API
Use REST API,



WebSocket API

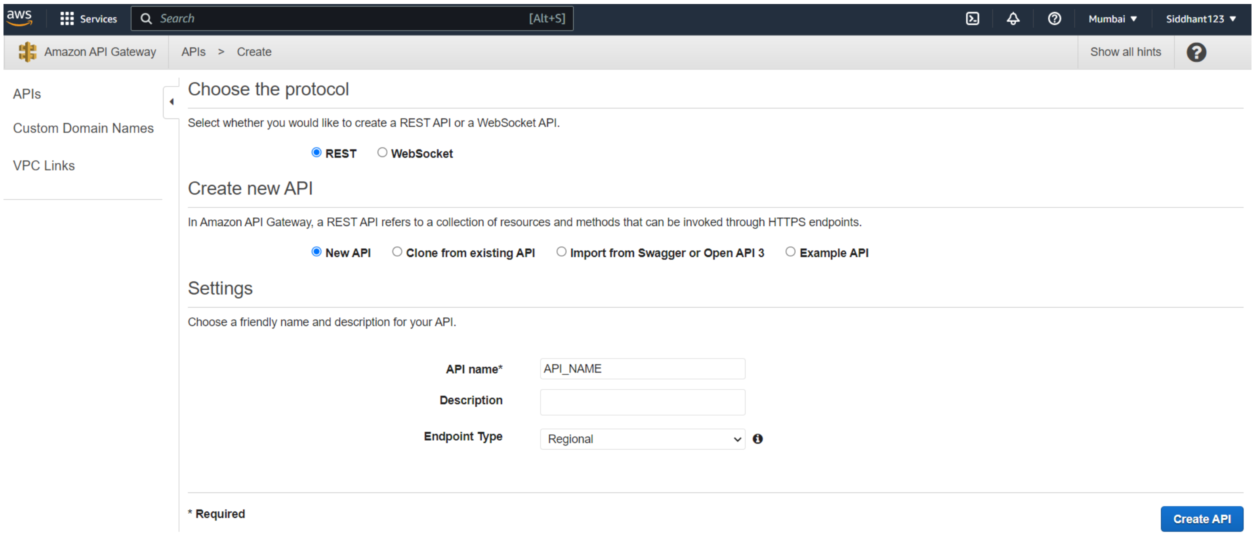
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.

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



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 ○ Clone from existing API ○ Import from Swagger or Open API 3 ○ Example API

Settings

Choose a friendly name and description for your API.

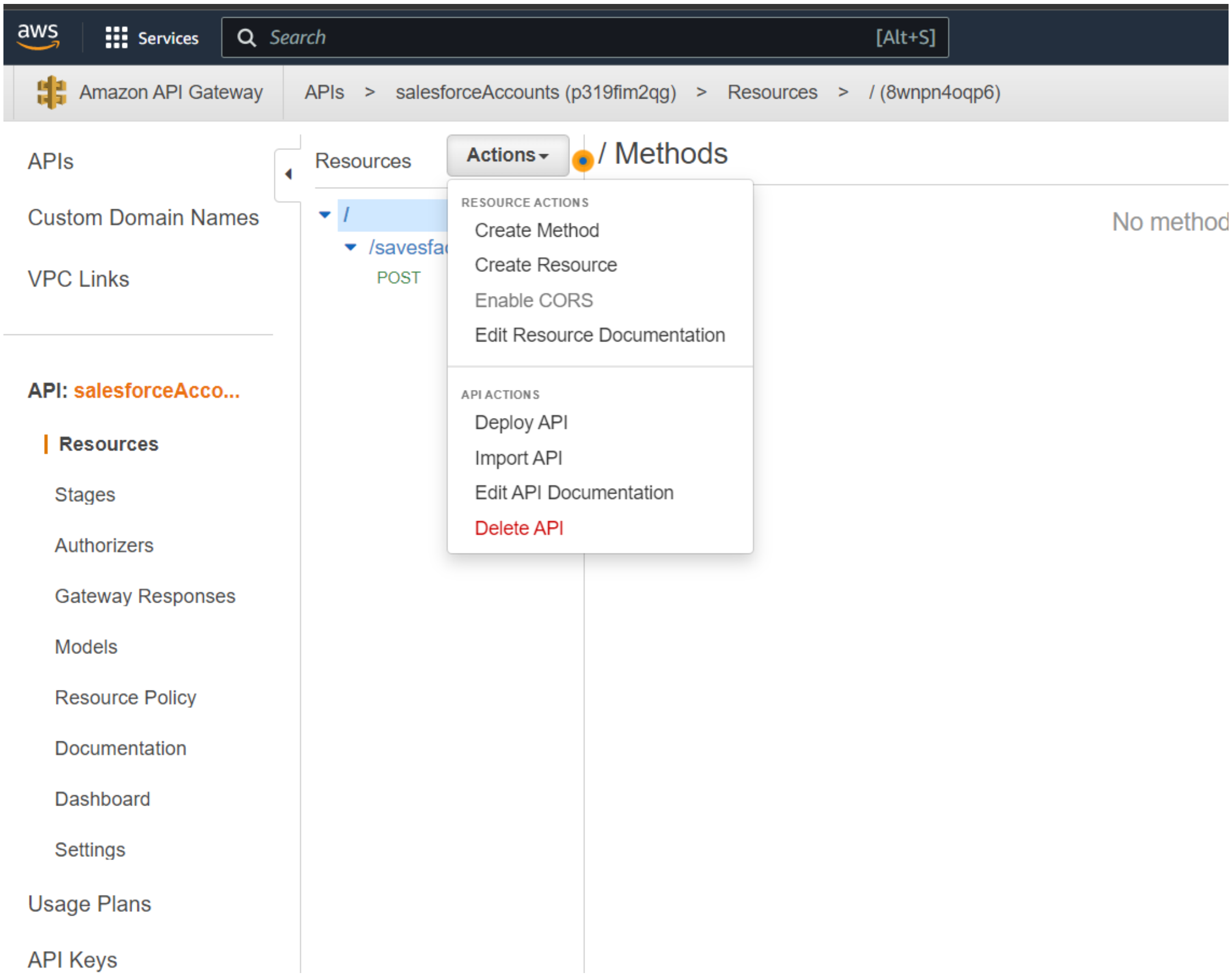
API name* API_NAME

Description

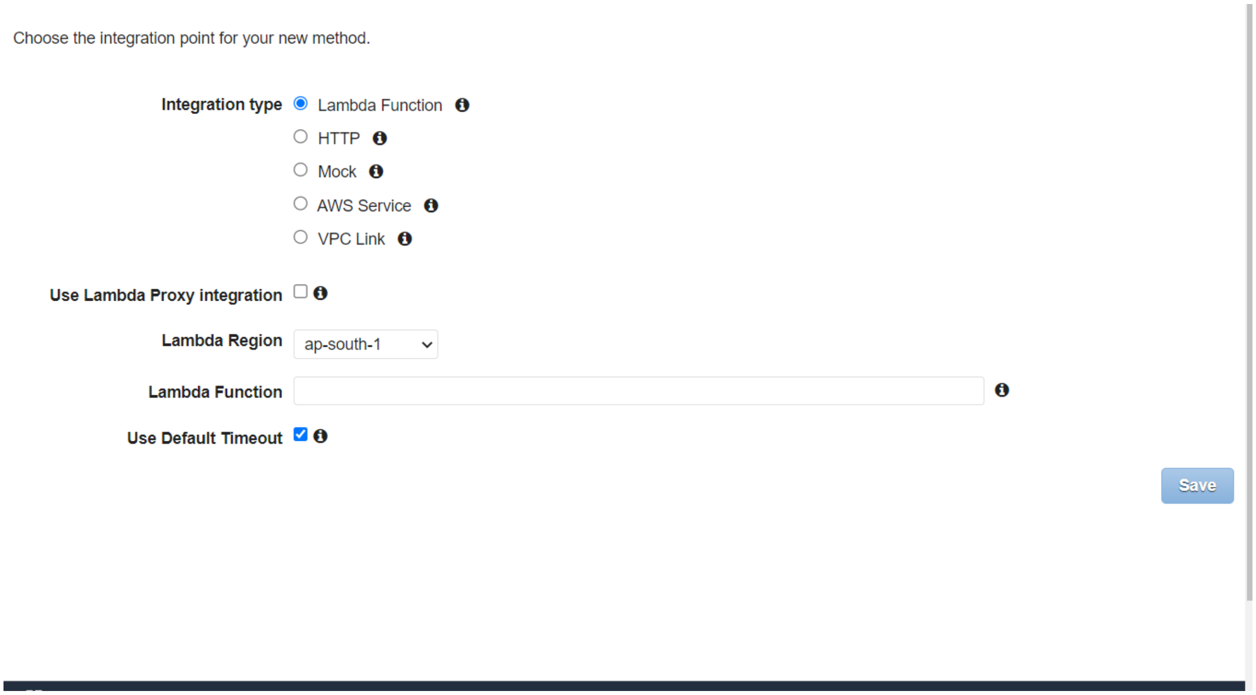
Endpoint Type Regional

* Required

Select Create resource



Create Method 'POST' under the resource,
Select the Lambda function as the integration type and provide the lambda function name which we created earlier



Choose the integration point for your new method.

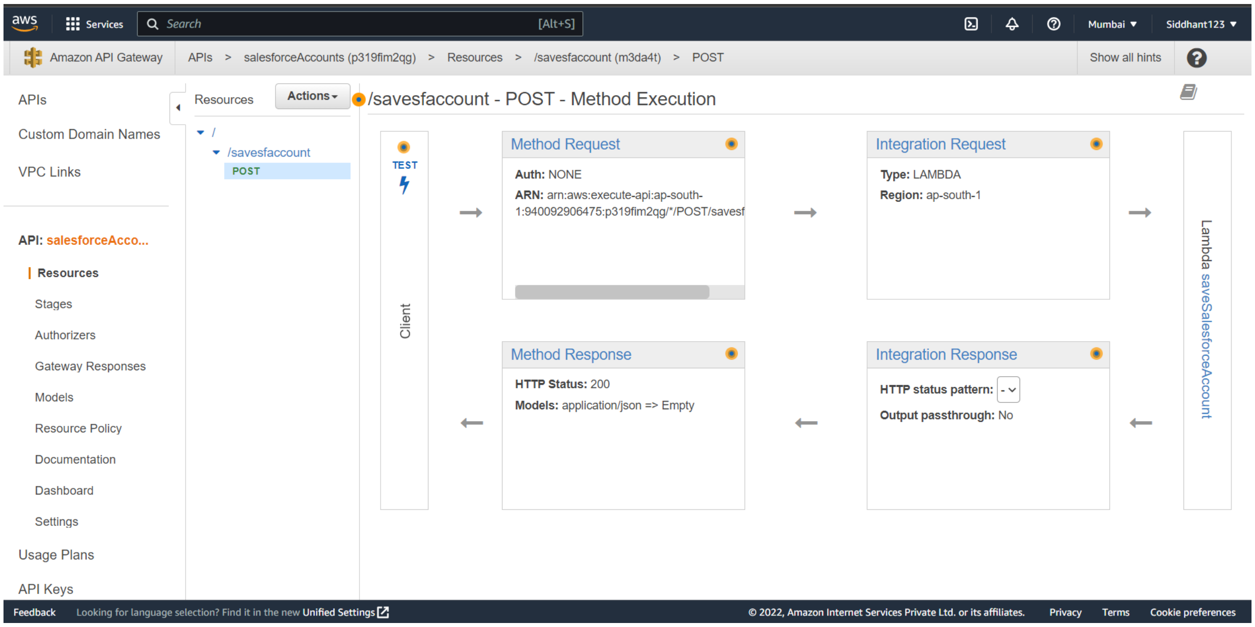
Integration type ● Lambda Function ○ HTTP ○ Mock ○ AWS Service ○ VPC Link

Use Lambda Proxy integration □

Lambda Region ap-south-1

Lambda Function

Use Default Timeout ☑



/savesfaccount - POST - Method Execution

Method Request
Auth: NONE
ARN: arn:aws:execute-api:ap-south-1:940092906475:p319fim2qg/*/POST/savesf

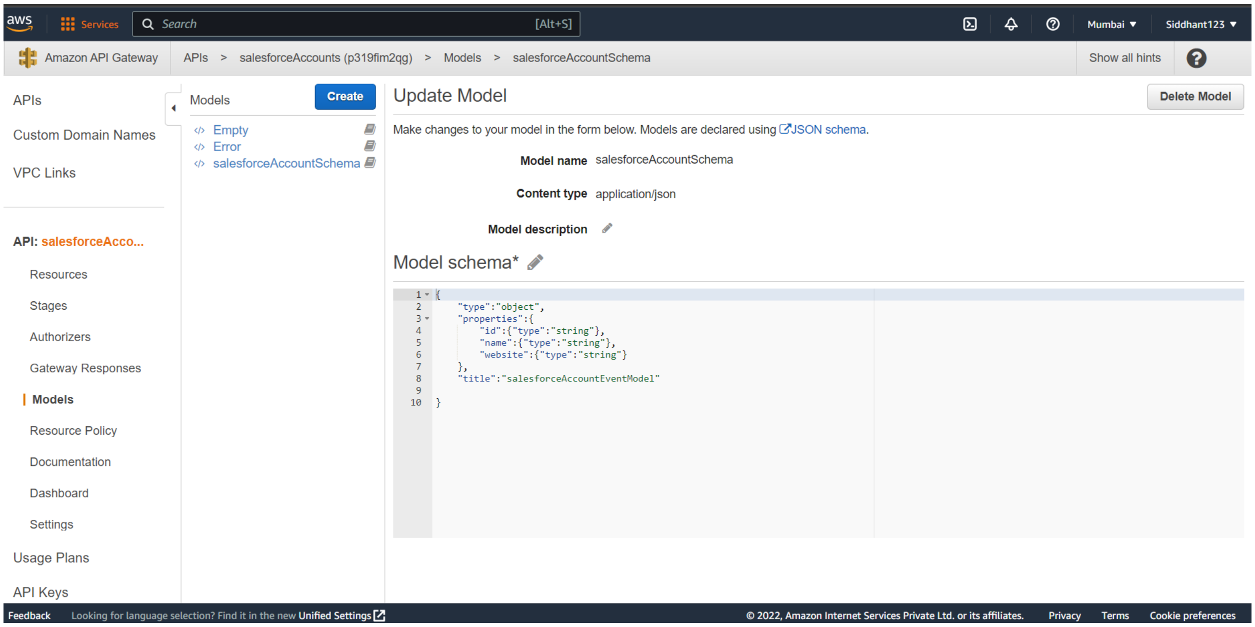
Integration Request
Type: LAMBDA
Region: ap-south-1

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

Integration Response
HTTP status pattern: -
Output passthrough: No

Lambda saveSalesforceAccount

Select Models and click create,



Update Model

Make changes to your model in the form below. Models are declared using JSON schema.

Model name salesforceAccountSchema

Content type application/json

Model description

Model schema*

```
{
    "type":"object",
    "properties":{
        "id":{"type":"string"},
        "name":{"type":"string"},
        "website":{"type":"string"}
    },
    "title":"salesforceAccountEventModel"

}
```

Deploy the API via Actions-> deploy API

To test end-to-end connection,
Create an account on salesforce org, it will create an item in aws dynamodb table.