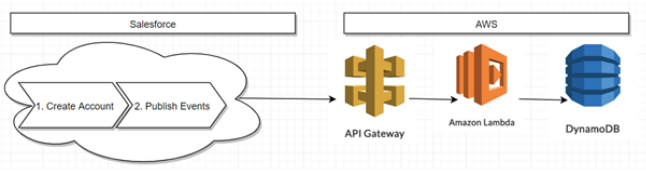
Salesforce AWS Integration using API gateway and Lambda

Published on May 27, 2019

Rajesh K.
Senior Salesforce Development Engineer at Akamai | Ex-Morgan Stanley

5 articles + Follow



S

Prerequisite

1. AWS Free Tier Account ,
2. Salesforce Developer Account
3. Basic to advance knowledge of Salesforce Architecture and Coding Pattern.

4. Basic to Intermediate knowledge AWS lambda, Dynamo DB and server less processing.

Consideration

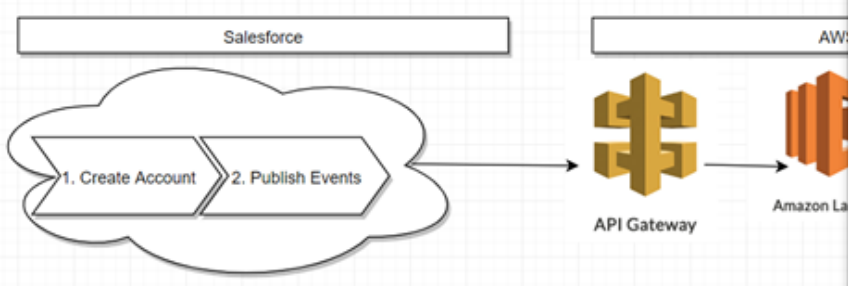
1. All AWS service creation has been kept simple .. i.e most part like AWS lambda has been configured to have admin access role .
2. Connection b/w Salesforce and AWS api gateway has been made through open authentication.
3. I will have separate article for security consideration b/w salesforce and AWS connection

Use Case Flow

1. Account is created in Salesforce

3. Amazon Gateway receives the data and transform and pass to an

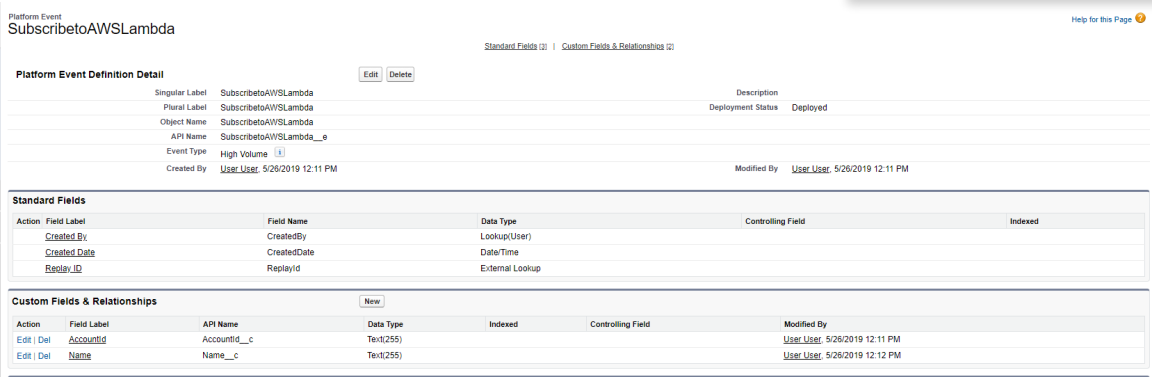
4. Amazon Lambda has logic to pass this data to dynamo DB and i



Salesforce Changes

a) Create Platform Even name **"SubscribetoAWSLambda"** (you ca want)

b) Create two custom field in platform event "AccountId" & Name i (Refer below image for more detail)



Platform Event
SubscribetoAWSLambda

Help for this Page

Standard Fields [3] | Custom Fields & Relationships [2]

Platform Event Definition Detail Edit Delete

Singular Label	SubscribetoAWSLambda	Description	
Plural Label	SubscribetoAWSLambda	Deployment Status	Deployed
Object Name	SubscribetoAWSLambda		
API Name	SubscribetoAWSLambda__e		
Event Type	High Volume		
Created By	User User, 5/26/2019 12:11 PM	Modified By	User User, 5/26/2019 12:11 PM

Standard Fields

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

Custom Fields & Relationships New

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit | Del	AccountId	AccountId__c	Text(255)			User User, 5/26/2019 12:11 PM
Edit | Del	Name	Name__c	Text(255)			User User, 5/26/2019 12:12 PM

c) Create process builder to publish data in above vents when account is created ,

d) Create below class (Please note this class can be optimised for bulk operations and AWS can be read from custom Metadata/Custom Setting) . Please note that endpoint will be replaced when we create API gateway in later section.

```
public class AWSCallout

{

@future(callout = true)

public static void callAwsgateway(String recordId,String Name)
```

```
HttpRequest request = new HttpRequest();
//Please set the endpoint of API gateway which we are going to create in
request.setEndpoint('awsapigateway');
request.setMethod('POST');
request.setHeader('Content-Type', 'application/json');
// Set the body as a JSON object
request.setBody('{"id": "'+recordId+'","Name": "'+Name+'"}');
HttpResponse response = http.send(request);
System.debug('response==='+response.getBody());
// Parse the JSON response
if (response.getStatusCode() != 201) {
    System.debug('The status code returned was not expected: ' +
        response.getStatusCode() + ' ' + response.getStatus());
} else {
    System.debug(response.getBody());

}
}

}
```

e) Create Subscriber on platform Event (basically platform event Trigger)

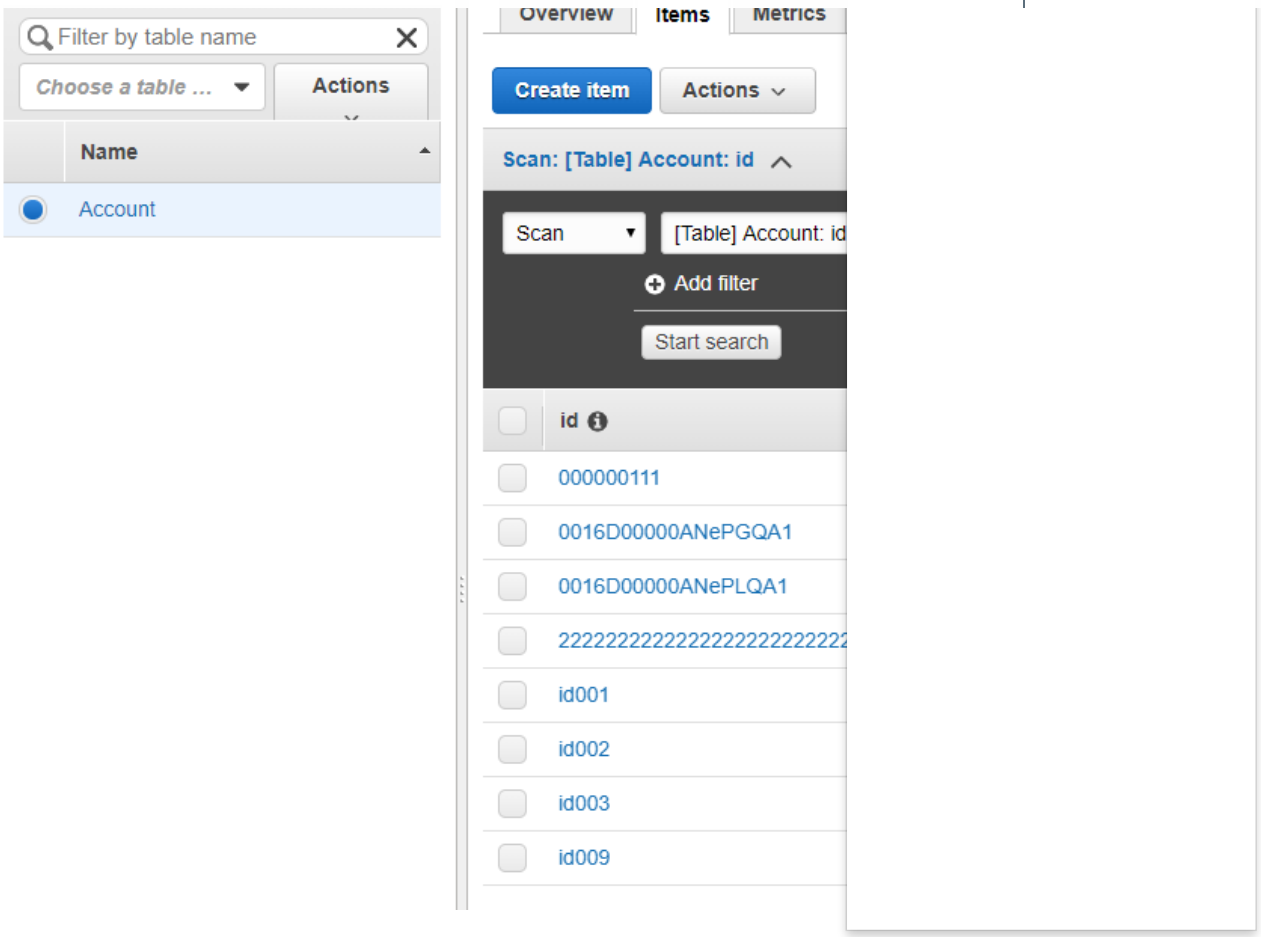
```
trigger callAWSAPIgateway on SubscribetoAWSLambda__e (after Insert) {
for (SubscribetoAWSLambda__e saws : Trigger.new)
{
  AWSCallout.callAwsgateway(saws.AccountId__c,saws.Name__c);
}

}
```

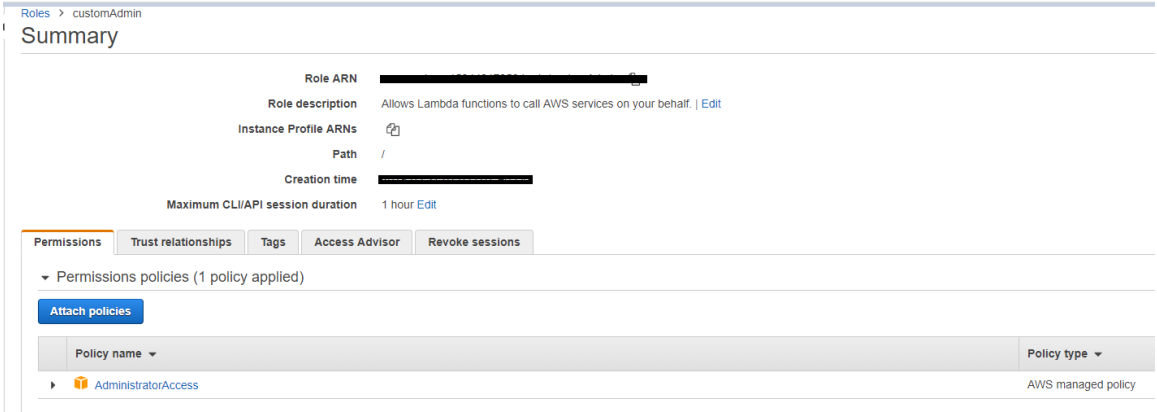
f) Do not forget to add the actual end point mentioned in Step c) to remote site setting.

AWS Changes

> Create below dynamodb table with "id" as Primary partition key



> Login to AWS and go to IAM and create custom role with Administrator access policy .I have used this role for AWS lambda , please note in real time situation we should not give admin access security should be limited here I have used just for demo purpose to avoid any access related issue .

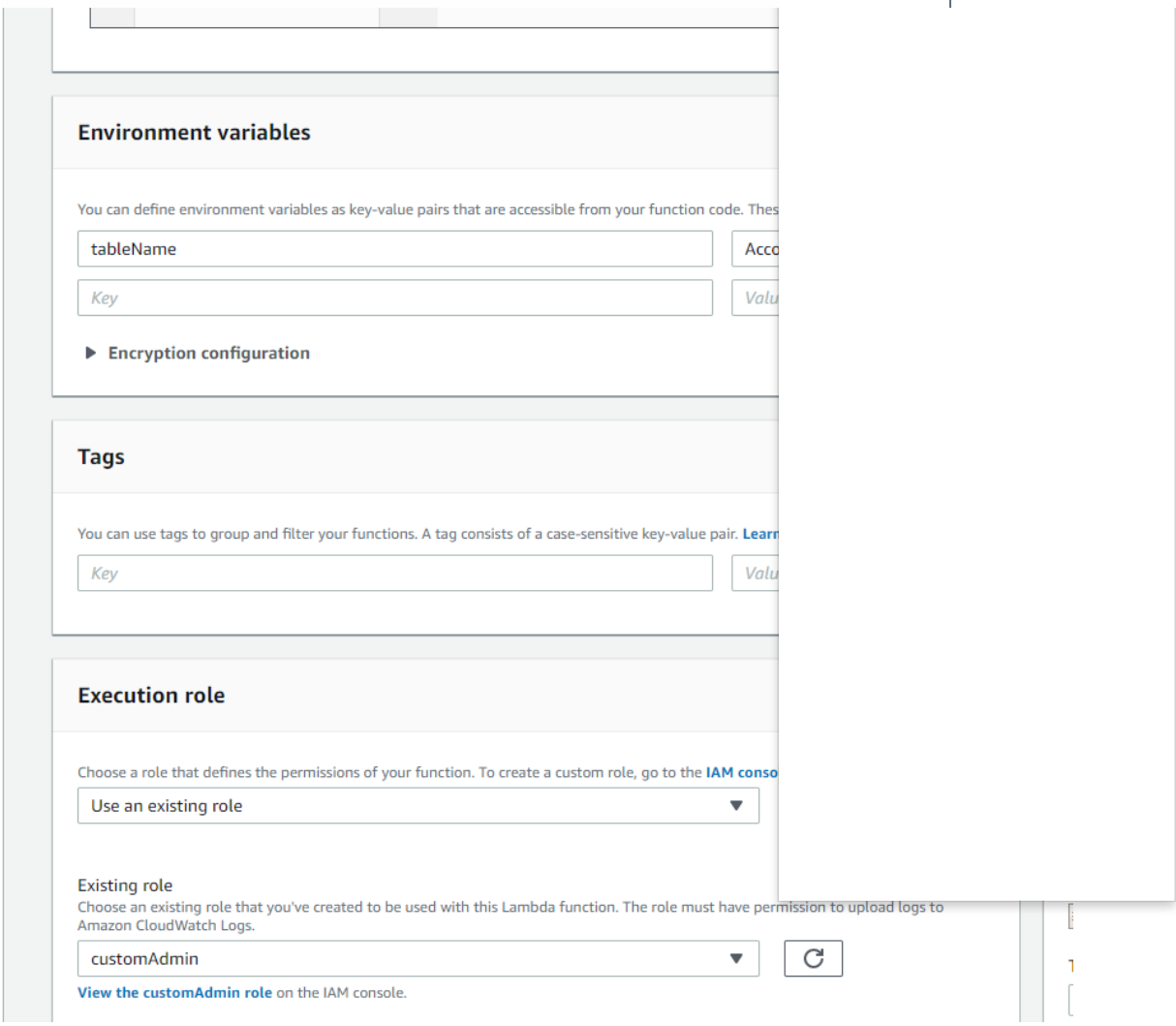


> In services search for lambda function and create below lambda function and in execution role provide role created in above steps .

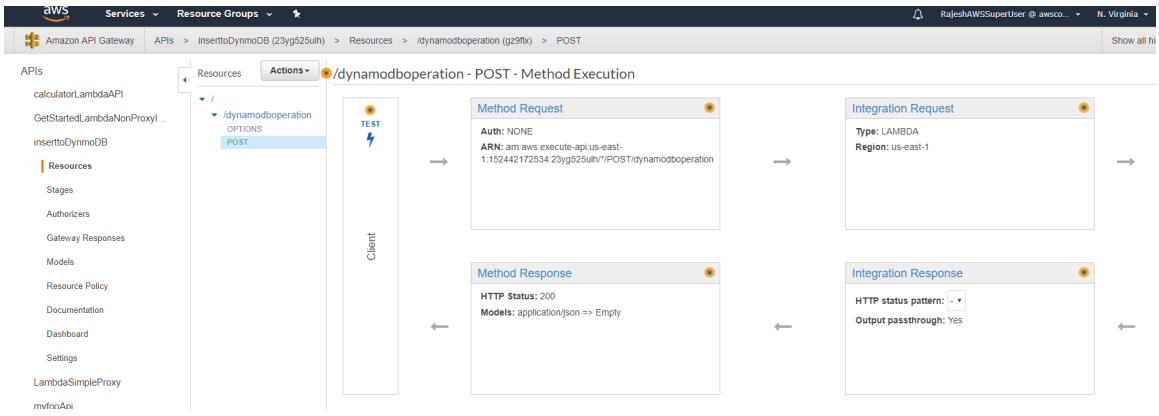
```
const dynamodb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

exports.handler = (event, context, callback) => {
    dynamodb.putItem({
        TableName: process.env.tableName,
        Item: {
            "id": {
                S: event.id
            },
            "Name": {
                S: event.Name
            }
        }
    }, function(err, data) {
        if (err) {
            console.log(err, err.stack);
            callback(null, {
                statusCode: '500',
                body: err
            });
        } else {
            callback(null, {
                statusCode: '200',
                body: 'Hello ' + event.id + event.Name + '!'
            });
        }
    })
};
```

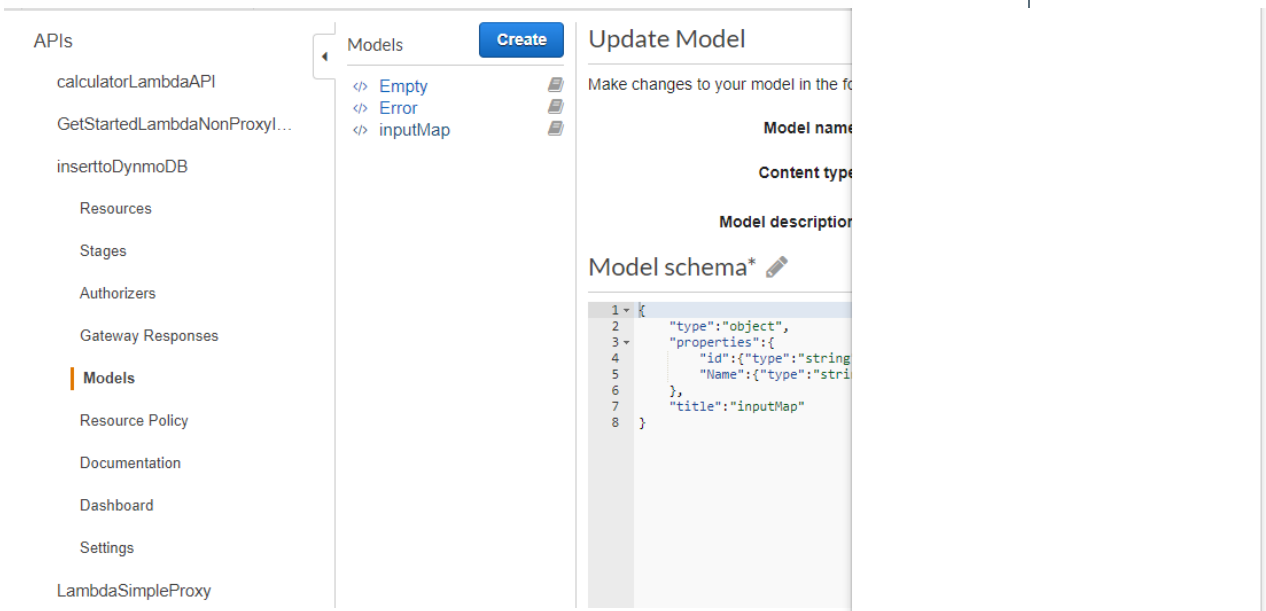
Below image shows the setup for lambda function



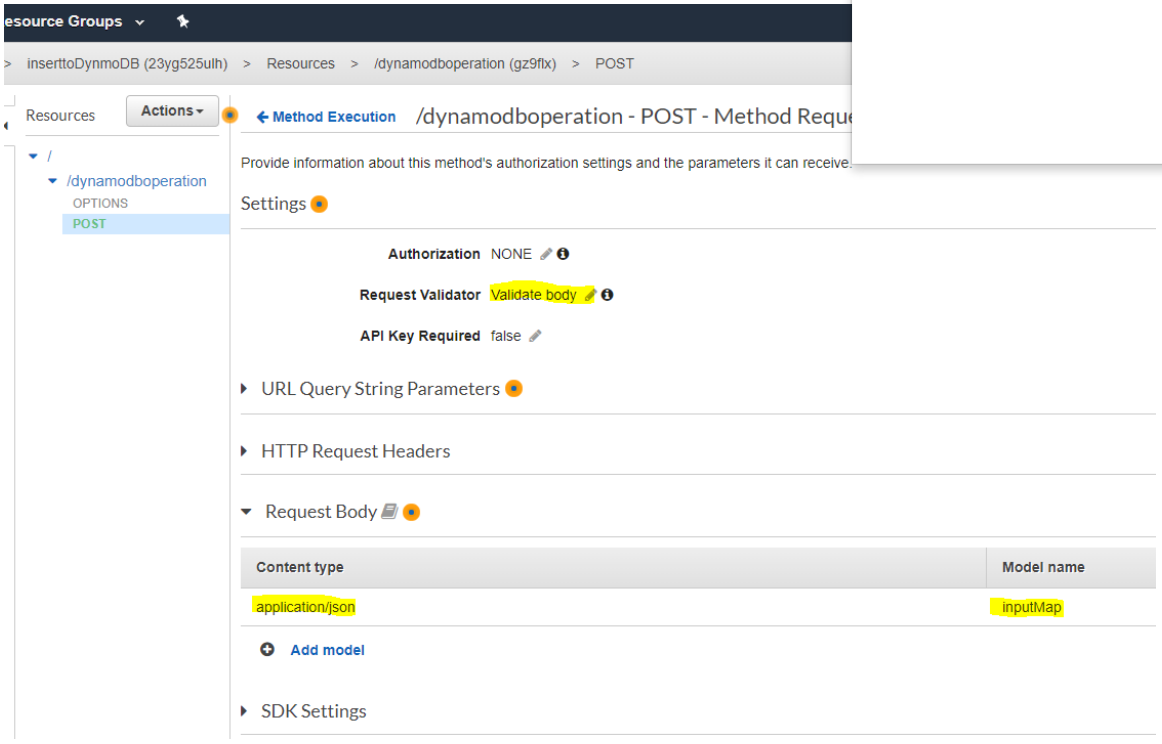
> Create below API gateway



> Create below model inside the gateway



> Configure post method as below .



> Deploy API gateway to stage (you can stage name whatever you want) I have named test.

> Go to "test" Stage and click on POST method and note down the Invoke URL and replace into your code in Salesforce Changes Step c) .

> Test the functionality by creating account in Salesforce which will flow to dynamo db table created earlier .

purpose for production code need to handle bulk scenario with

Published by

Rajesh K.
Senior Salesforce Development Engineer at Akamai | Ex-Morgan Stanley
Published • 1y

Salesforce-AWS Dynamo DB Integration with AWS lambda and API getway .

Like Comment Share

Reactions

7 Comments

Most Relevant

Add a comment...

Anirban Goswami • 3rd+
Senior Data Engineer - Apple
1y ...

truly fine stuff

· 1 Reply

Rajesh K. • 2nd
Senior Salesforce Development Engineer at Akamai | Ex-Morgan Stanley
1y ...

Thanks **Anirban** !!!

Siddharth Raj • 2nd
Senior Data Scientist at Accenture
1y ...

great job Rajesh..... really helpful

· 1 Reply

Rajesh K. • 2nd
Senior Salesforce Development Engineer at Akamai | Ex-Morgan Stanley
1y ...

Thank you Bhaiya !!

Rajesh K.

Senior Salesforce Development Engineer at Akamai | Ex-Morgan S

+ Follow

n Rajesh K.

Salesforce using JAVA

n LinkedIn

'Javascript Promises' in Lightning Web Component.

Rajesh K. on LinkedIn

Sharing Javascript code in Web Component(LWC)

Rajesh K. on LinkedIn

ghtr
orce)

icles