Let Us Create A Lambda Function For Our IoT Device In The AWS Cloud Using C#

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] ViceChancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area/Section: Computer Science.
Type of the Paper: Experimental Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed in: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.7995727
Google Scholar Citation: IJMTS

How to Cite this Paper:

Chakraborty, S., & Aithal, P. S. (2023). Let Us Create A Lambda Function for Our IoT Device In The AWS Cloud Using C#. *International Journal of Management, Technology, and Social Sciences (IJMTS), 8*(2), 145-155. DOI: https://doi.org/10.5281/zenodo.7995727

International Journal of Management, Technology, and Social Sciences (IJMTS)
A Refereed International Journal of Srinivas University, India.

CrossRef DOI: https://doi.org/10.47992/IJMTS.2581.6012.0276

Received on: 15/05/2023
Published on: 03/06/2023

Let Us Create A Lambda Function for Our IoT Device In The AWS Cloud Using C#

Sudip Chakraborty [1] & P. S. Aithal [2]
[1] D.Sc. Researcher, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *This research paper explores the process of creating a Lambda function for an IoT device in the AWS cloud using C#. It provides a comprehensive guide to the steps involved in building a Lambda function, including setting up the AWS environment, creating the Function using C#, and deploying it to the cloud. Additionally, the paper will discuss the benefits of using Lambda functions for IoT devices, such as cost-effectiveness and scalability. It will provide examples of real-world use cases for this technology. By the end of this paper, readers will clearly understand how to create Lambda functions for their IoT devices using C# in the AWS cloud and will be able to apply this knowledge to their projects.*
Design/Methodology/Approach: *A comprehensive literature review was conducted to gather information on the AWS cloud, IoT devices, and Lambda functions. The Lambda function was created using C# in Visual Studio. This involved writing the necessary code to perform the desired functionality, testing the Function locally, and packaging it for deployment. The Lambda function deployed to the AWS cloud using the AWS Management Console.*
Findings/Result: *The AWS environment can be easily configured with IoT devices and Lambda functions. AWS allows for cost-effective and scalable solutions that can be easily managed and monitored. Creating a Lambda function using C# in Visual Studio is straightforward and allows for powerful and flexible coding techniques. The deployment of the Lambda function to the AWS cloud is simple and can be achieved using the AWS Management Console.*
Originality/Value: *It offers a comprehensive guide to creating a Lambda function for an IoT device in the AWS cloud using C#. This guide includes a step-by-step approach to setting up the AWS environment, creating the Function using C#, and deploying it to the cloud. It demonstrates the real-world application of Lambda functions for IoT devices, providing examples of use cases and demonstrating the benefits of this technology.*
Paper Type: *Experimental-based Research.*

Keywords: Lambda function, IoT devices, AWS cloud, Use of C#

1. INTRODUCTION :

The Internet of Things (IoT) has emerged as a significant technology trend in recent years, enabling various devices to communicate with each other and the cloud. One of the critical challenges of IoT is managing the large amounts of data generated by these devices and ensuring their scalability and cost-effectiveness. Lambda functions, a serverless computing technology offered by Amazon Web Services (AWS), have emerged as a powerful and flexible solution for managing IoT devices. Using Lambda functions, developers and engineers can create functions that automatically scale based on demand, reducing costs and improving performance. This research paper aims to provide a comprehensive guide to creating a Lambda function for an IoT device in the AWS cloud using C#. The paper will cover the entire process, from setting up the AWS environment and creating the Function using C# in Visual Studio to deploying the Function to the cloud.
Additionally, it will provide examples of real-world use cases for Lambda functions in IoT applications and demonstrate the benefits of this technology, including cost-effectiveness and scalability. Overall, it

offers a valuable resource for developers and engineers looking to implement Lambda functions in their IoT projects and businesses and organizations looking to develop cost-effective and scalable IoT solutions. The paper is organized as follows: Section 2 provides an overview of related work already done on IoT. Section 3 discusses the objective of the research work. Section 4 highlights the methodology we used for the research work. In section 5, we do the actual experiment. This section describes the procedure with a required screenshot to create an IoT in the AWS cloud. Section 6 provides recommendations for reading to understand the research topic better. Finally, Section 7 concludes the paper and provides future research directions.

2. RELATED WORKS :

In their paper, Das, Patterson, and Wittie present a benchmarking tool for evaluating the performance of edge computing platforms. The tool, called Edgebench, measures the latency, throughput, and resource usage of edge computing [1]. Lee, Satyam, and Fox conducted a study evaluating production serverless computing environments [2]. Baldini et al. discussed the current state of serverless computing and the challenges [3]. Lloyd et al. investigate the factors influencing microservice performance in their paper [4]. Pelle et al. evaluate the performance of AWS for latency-sensitive cloud-native applications by measuring the end-to-end latency of several cloud-native services, including AWS Lambda, AWS Fargate, and AWS Elastic Beanstalk [5]. Yadavalli et al. propose an intelligent IoT system for monitoring and controlling livestock parameters [6]. Baresi, Filgueira Mendonça, and Garriga present a serverless edge computing architecture for low latency [7]. Gillam et al. explores edge computing for connected and autonomous [8]. Bastos provides an overview of the technologies and security features of public cloud IoT solutions. The paper discusses the benefits and challenges of using public cloud IoT solutions, such as scalability, cost-effectiveness, and security risks [9]. Ihejimba and Wenkstern propose a cloud-based traffic signal notification system for blind people. The system uses computer vision and machine learning techniques to detect traffic signals. It sends notifications to users through a mobile app, enabling them to navigate safely and independently in urban environments [10].

3. OBJECTIVES :

The objectives of this research paper on creating a Lambda function for an IoT device in the AWS cloud using C# are:

(1) To provide a comprehensive guide to creating a Lambda function for an IoT device in the AWS cloud using C# in Visual Studio.
(2) To demonstrate the real-world application of Lambda functions for IoT devices, including examples of use cases and the benefits of this technology.
(3) To highlight the technical details of creating a Lambda function using C# and deploying it to the AWS cloud.
(4) To showcase the cost-effectiveness and scalability of Lambda functions for IoT devices and how they can provide a powerful and flexible solution for a wide range of applications.
(5) To test and evaluate the Lambda function to ensure that it meets the specified requirements and can handle a variety of inputs and produce the desired output.

Overall, this research paper aims to provide a valuable resource for developers, engineers, and businesses looking to implement Lambda functions in their own IoT projects and to demonstrate the feasibility and effectiveness of this technology for managing IoT devices.

4. APPROACH AND METHODOLOGY :

In order to create a Lambda function for an IoT device in the AWS cloud using C#, the following methodology can be adopted:

- ❖ **Setting up the AWS environment:** First, the AWS environment needs to be set up, which includes creating an AWS account, creating a Lambda function, and creating an IoT device.
- ❖ **Configuring the AWS Lambda function:** Once the AWS environment is set up, the next step is to configure the AWS Lambda function. This includes specifying the function name, the runtime environment (in this case, C#), and the function handler.
- ❖ **Writing the C# code:** With the AWS Lambda function configured, the next step is to write the C# code that will run on the Lambda function. This code will communicate with the IoT device and perform necessary actions based on the data received.

- ❖ **Testing and debugging the Lambda function:** Once the C# code is written, it must be tested and debugged to ensure it works as expected. This can be done using the AWS Lambda console or the command line interface.
- ❖ **Deploying the Lambda function:** Finally, the Lambda function needs to be deployed to the AWS cloud so that the IoT device can use it. This can be done using the AWS Lambda console or the command line interface.

Following this methodology, the Lambda function for an IoT device in the AWS cloud using C# can be created and deployed successfully.

5. EXPERIMENT :

Create a User in IAM

1) Open https://aws.amazon.com/
2) Click "Sign In to the Console" from the top right corner. If an AWS account is not available, create it.
3) Let us consider already having an AWS account. According to the User, Select **Root** or **IAM**. Provide ID and password.
4) Open the IAM console. From the left side panel, click on "**User.**" From the top right, click on the "**Add users**" button.
5) Enter a username like "**User_Alexa_IoT**"—press the "**Next**" button.
6) Permission options: select "Add user to a group" and click "**Next**."
7) At the bottom right, click on "**Create user**." The User will be created and displayed under the User name list.
8) Now click on Just Created User (user name), and click on the "**Security credentials**" tab (middle of the window).
9) Scroll down> under the "**Access keys**," in the middle right, click on the "**Create access key**" button.
10) Select "**Application running outside AWS**." At the bottom, click on the "**Next**" button.
11) Under the Set Description tag- optional, add like "Alexa request Handle."
12) Click on the "**Create access key**."

Access key: AKIAU[redacted]NPYZT
Secret access key: urpnBzp937XAtcy[redacted]/VTlSzuoX

13) Copy the access key and Secret access key. We can download it as a .csv file by clicking the "Download .csv file" button. Click on the "**Done**" button.
14) In the middle left (first) tab of the windows, click on "**Permissions**." Click "**Add permissions.**" select "**Create inline policy**." Click on "**Choose a service**." Inside the search box, type "Lambda." click on "**Lambda**." under the Manual actions, check on "**All Lambda actions**."
15) Under the resources, click the "**All resources**" radio button. Click on "**Review policy**."
16) Inside the name textbox, type a name like "Policy_Alexa_IoT."
17) At the bottom, click on the "**Create policy" button**.
18) Click on the policy name "Policy_Alexa_IoT." Click on "**+ Add additional permissions**." Click on "**Choose a service**." Inside the search box, type "iot"> click on "**IoT**"> under the Manual actions, check on "**All IoT actions (iot:*)**." Click on "**Review policy**."
19) Click on "**Save changes**."
20) To activate the policies, better to close the browser.

Create Role:

1) From the left side, click on "**Role**."
2) From the right side, click on "**Create Role**."
3) Keep selected "**AWS service**." Click on "**Next**."
4) Under "**Use case**." Select "**Lambda**." Click "**Next**"
5) Search "**Iot**" and check the box on "**AWSIoTFullAccess**." Click on "**Next**"
6) Enter a role name like "role_Lambda_Alexa_IoT."
7) Click on "**Create role**."

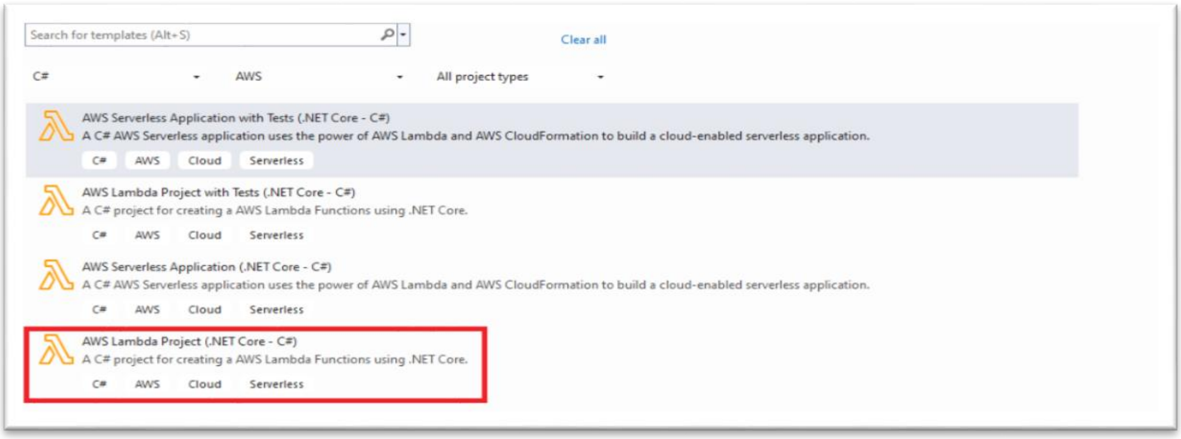
Install AWS Toolkit

1) Download and Install the Visual Studio community edition from https://visualstudio.microsoft.com/vs/community/

2) Open Visual Studio. Click on “Continue without code.” From the top menu bar, click “Extensions” and .click on “Manage Extensions.” Inside the search box, type “**AWS**.” click on the “**Download**” button next to “AWS Toolkit for visual studio 2022”. Click on the “**Close**” button. Close visual studio IDE. After a while, the VSIX installer will start the installation. Click on “**Modify**.” After a few minutes; the installation will be completed. Press the “**Close**” button.

Create Lambda function:

1. Create a folder on the Desktop.
2. Open Visual Studio.
3. Click on “**Create a new project**.”
4. Select C#, AWS, and All project types. The available project templates will be displayed below. Select “**AWS Lambda Project(.NET Core-C#),**” in figure 1. Click on “**Next**”



Search for templates (Alt+S)

Clear all

C# AWS All project types

AWS Serverless Application with Tests (.NET Core - C#)
A C# AWS Serverless application uses the power of AWS Lambda and AWS CloudFormation to build a cloud-enabled serverless application.
C# AWS Cloud Serverless

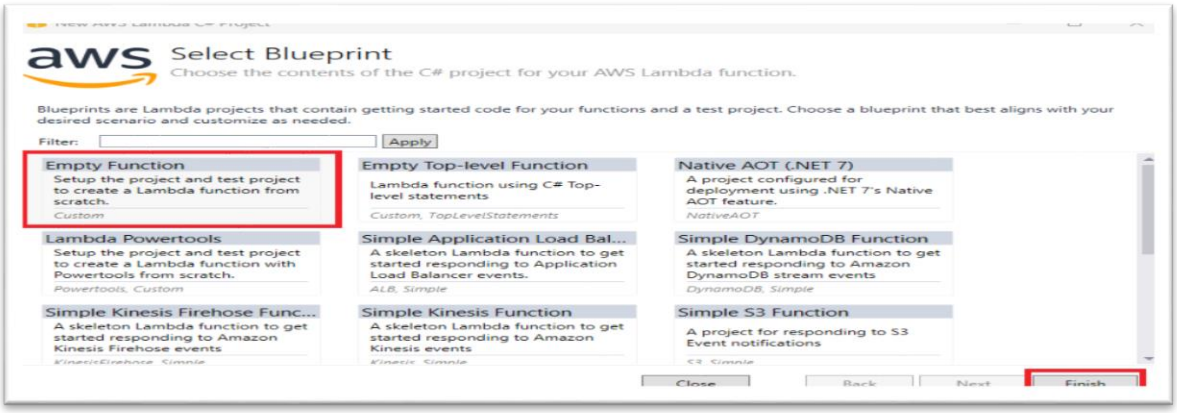
AWS Lambda Project with Tests (.NET Core - C#)
A C# project for creating a AWS Lambda Functions using .NET Core.
C# AWS Cloud Serverless

AWS Serverless Application (.NET Core - C#)
A C# AWS Serverless application uses the power of AWS Lambda and AWS CloudFormation to build a cloud-enabled serverless application.
C# AWS Cloud Serverless

AWS Lambda Project (.NET Core - C#)
A C# project for creating a AWS Lambda Functions using .NET Core.
C# AWS Cloud Serverless

Fig. 1: Project Type selection [source: Microsoft Visual Studio 2022 Community edition]

5. Enter the project name and location. Click on the “**Create**” button. One window appears. Select Empty Function and click on the “**Finish**” button. It is depicted in Figure 2.



aws Select Blueprint
Choose the contents of the C# project for your AWS Lambda function.

Blueprints are Lambda projects that contain getting started code for your functions and a test project. Choose a blueprint that best aligns with your desired scenario and customize as needed.

Filter: Apply

Empty Function
Setup the project and test project to create a Lambda function from scratch.
Custom

Empty Top-level Function
Lambda function using C# Top-level statements
Custom, TopLevelStatements

Native AOT (.NET 7)
A project configured for deployment using .NET 7's Native AOT feature.
NativeAOT

Lambda Powertools
Setup the project and test project to create a Lambda function with Powertools from scratch.
Powertools, Custom

Simple Application Load Bal...
A skeleton Lambda function to get started responding to Application Load Balancer events.
ALB, Simple

Simple DynamoDB Function
A skeleton Lambda function to get started responding to Amazon DynamoDB stream events
DynamoDB, Simple

Simple Kinesis Firehose Func...
A skeleton Lambda function to get started responding to Amazon Kinesis Firehose events
KinesisFirehose, Simple

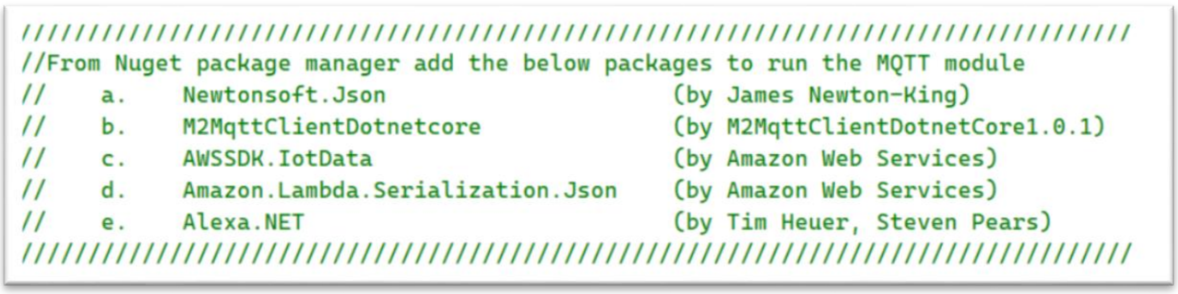
Simple Kinesis Function
A skeleton Lambda function to get started responding to Amazon Kinesis events
Kinesis, Simple

Simple S3 Function
A project for responding to S3 Event notifications
S3, Simple

Close Back Next Finish

Fig. 2: Function selection [source: Microsoft Visual Studio 2022 Community edition]

6. Download the two classes from GitHub https://github.com/sudipchakraborty/AWS-Lambda-Function-for-Our-IoT-Using-C-sharp.git. Inside the project, Paste “Function. cs” and “AWS_MQTT_Client.cs.”. add two classes as existing items.



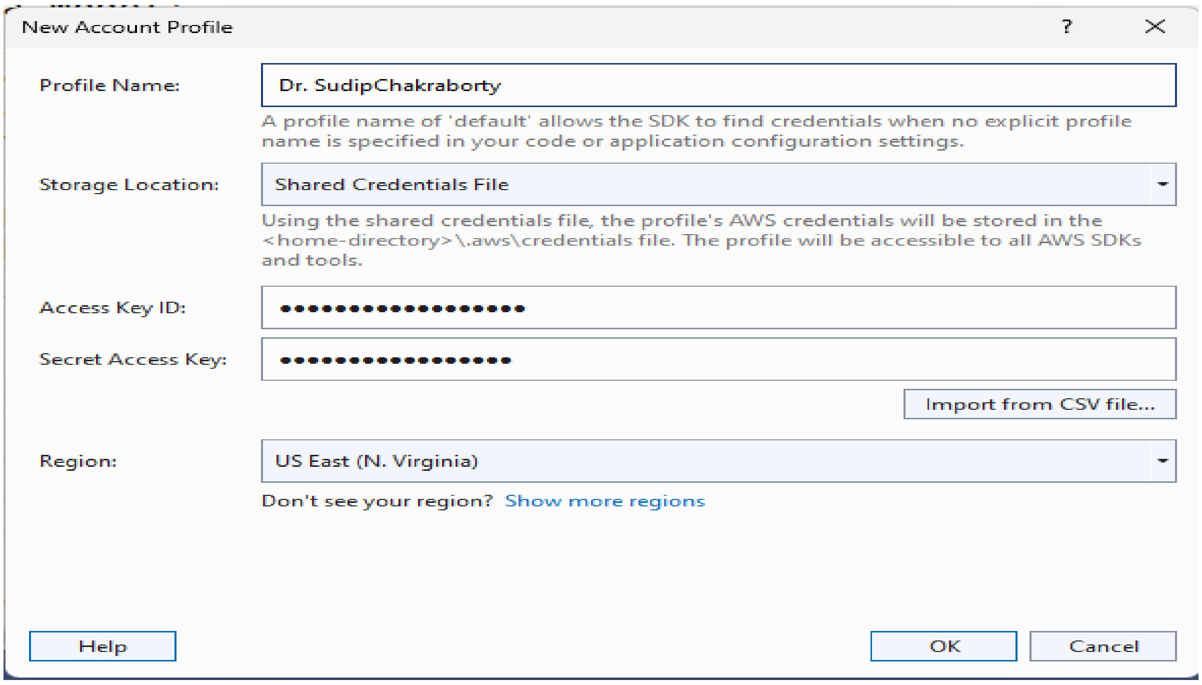
```
/////////////////////////////////////////////////////////////////////////////////
//From Nuget package manager add the below packages to run the MQTT module
//     a.    Newtonsoft.Json                    (by James Newton-King)
//     b.    M2MqttClientDotnetcore             (by M2MqttClientDotnetCore1.0.1)
//     c.    AWSSDK.IotData                     (by Amazon Web Services)
//     d.    Amazon.Lambda.Serialization.Json   (by Amazon Web Services)
//     e.    Alexa.NET                          (by Tim Heuer, Steven Pears)
/////////////////////////////////////////////////////////////////////////////////
```

Fig. 3: required Package to Install [source: Microsoft Visual Studio 2022 Community edition]

7. From the Nuget package manager, install the below packages as depicted in figure 3.
8. Build the project.

<u>Add IAM Credential to the AWS Explorer:</u>

1) The AWS Explorer is available on the left side of the visual studio IDE.
2) On the top, click the “Add AWS Credentials Profile” button. The New Account Profile window appears as depicted in Figure 5.4
3) Select the region by filling in the Profile name, Access Key ID, and Secret Access key. Click on “OK”.



New Account Profile

Profile Name: Dr. SudipChakraborty

A profile name of 'default' allows the SDK to find credentials when no explicit profile name is specified in your code or application configuration settings.

Storage Location: Shared Credentials File

Using the shared credentials file, the profile's AWS credentials will be stored in the <home-directory>\.aws\credentials file. The profile will be accessible to all AWS SDKs and tools.

Access Key ID: ••••••••••••••••••

Secret Access Key: •••••••••••••••••

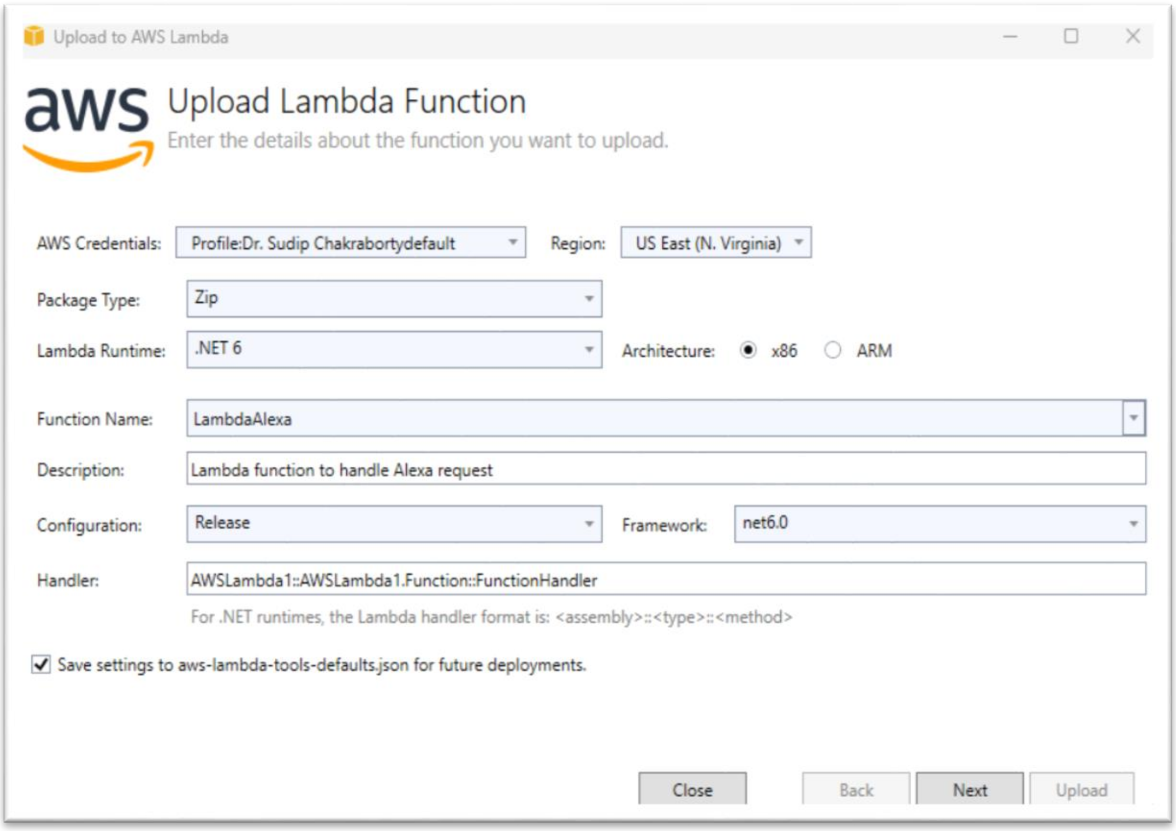
Region: US East (N. Virginia)

Don't see your region? Show more regions

Fig. 4: New Account Profile window [source: Microsoft Visual Studio 2022 Community edition]

<u>Publish Lambda function:</u>

1) From the right side of the IDE, right-click on the project name. Click on “**Publish to AWS Lambda…**”
2) The upload window will appear. Fill function name, Description depicted in Figure 5.



Upload to AWS Lambda

Upload Lambda Function

Enter the details about the function you want to upload.

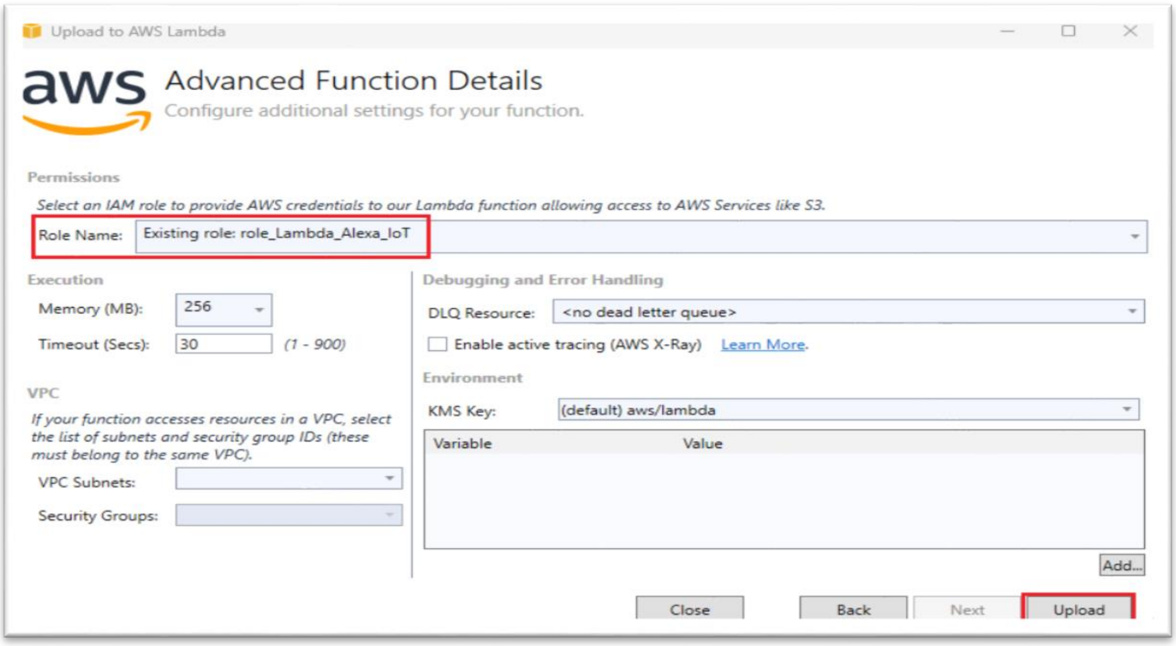
AWS Credentials: Profile:Dr. Sudip Chakrabortydefault
Region: US East (N. Virginia)
Package Type: Zip
Lambda Runtime: .NET 6
Architecture: [✓ x86] ARM
Function Name: LambdaAlexa
Description: Lambda function to handle Alexa request
Configuration: Release
Framework: net6.0
Handler: AWSLambda1::AWSLambda1.Function::FunctionHandler

For .NET runtimes, the Lambda handler format is: <assembly>::<type>::<method>

☑ Save settings to aws-lambda-tools-defaults.json for future deployments.

Fig. 5: Upload Interface [source: Microsoft Visual Studio 2022 Community edition]

3) Click on "**Next**." The "**Advance function details**" window will appear. Under the role Name dropdown list, Select an existing role. If the role is not available, under the "**New role based on AWS managed policy,**" select "AWSLambda_FullAccess(Grants full access to AWS Lambda service, AWS Lambda console features, and other related AWS services)." Click on the "**Upload**" button—the interface depicted in Figure 5.6.



Upload to AWS Lambda

Advanced Function Details

Configure additional settings for your function.

Permissions

Select an IAM role to provide AWS credentials to our Lambda function allowing access to AWS Services like S3.

Role Name: Existing role: role_Lambda_Alexa_IoT

Execution
Memory (MB): 256
Timeout (Secs): 30 *(1 - 900)*

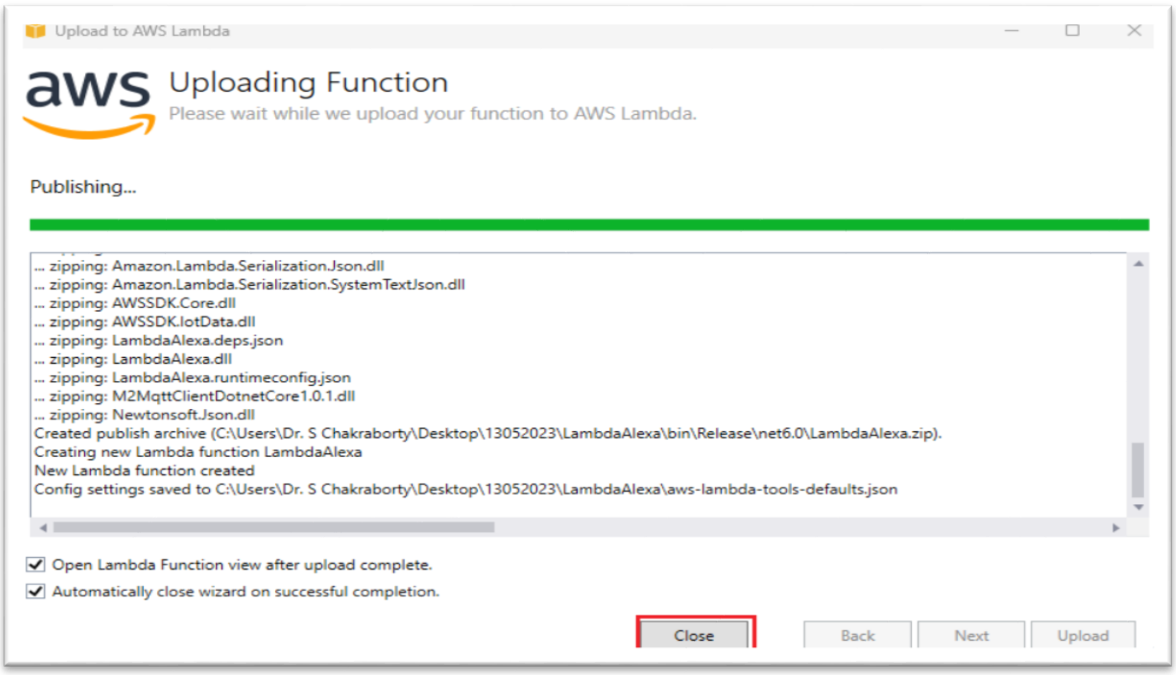
VPC
If your function accesses resources in a VPC, select the list of subnets and security group IDs (these must belong to the same VPC).
VPC Subnets:
Security Groups:

Debugging and Error Handling
DLQ Resource: <no dead letter queue>
☐ Enable active tracing (AWS X-Ray) Learn More.

Environment
KMS Key: (default) aws/lambda

Variable	Value

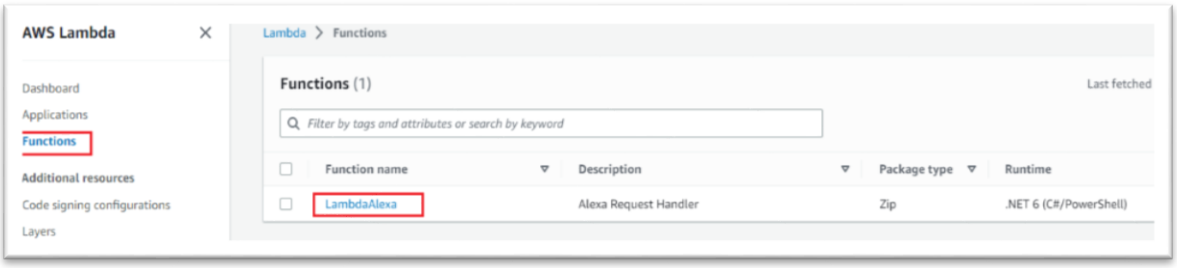
Fig. 6: upload Interface [source: Microsoft Visual Studio 2022 Community edition]


Upload to AWS Lambda
aws
Uploading Function
Please wait while we upload your function to AWS Lambda.
Publishing...
Open Lambda Function view after upload complete.
Automatically close wizard on successful completion.
Close
Back
Next
Upload

```
... zipping: Amazon.Lambda.Serialization.Json.dll
... zipping: Amazon.Lambda.Serialization.SystemTextJson.dll
... zipping: AWSSDK.Core.dll
... zipping: AWSSDK.IotData.dll
... zipping: LambdaAlexa.deps.json
... zipping: LambdaAlexa.dll
... zipping: LambdaAlexa.runtimeconfig.json
... zipping: M2MqttClientDotnetCore1.0.1.dll
... zipping: Newtonsoft.Json.dll
Created publish archive (C:\Users\Dr. S Chakraborty\Desktop\13052023\LambdaAlexa\bin\Release\net6.0\LambdaAlexa.zip).
Creating new Lambda function LambdaAlexa
New Lambda function created
Config settings saved to C:\Users\Dr. S Chakraborty\Desktop\13052023\LambdaAlexa\aws-lambda-tools-defaults.json
```

Fig. 7: upload completed [source: Microsoft Visual Studio 2022 Community edition]

4. After a couple of minutes, the upload will finish. Click the "**Close**" button. As depicted in Figure 7.



Function name	Description	Package type	Runtime
LambdaAlexa	Alexa Request Handler	Zip	.NET 6 (C#/PowerShell)

Fig. 8 : List of lambda function [source: https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions

The link between Lambda and Alexa service:

1. Open Lambda console. From the left side, click on "**Functions**." The available Function will be displayed on the right side, depicted in Figure 8. Click on the function name " LambdaAlexa."
2. Click on the "**+ Add trigger**" button. Depicted in Figure 9.

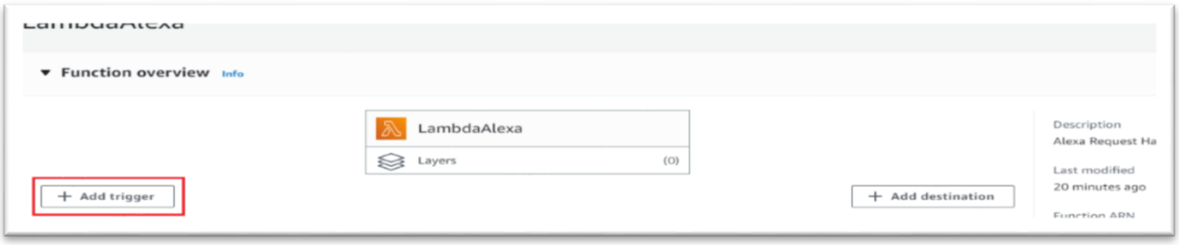
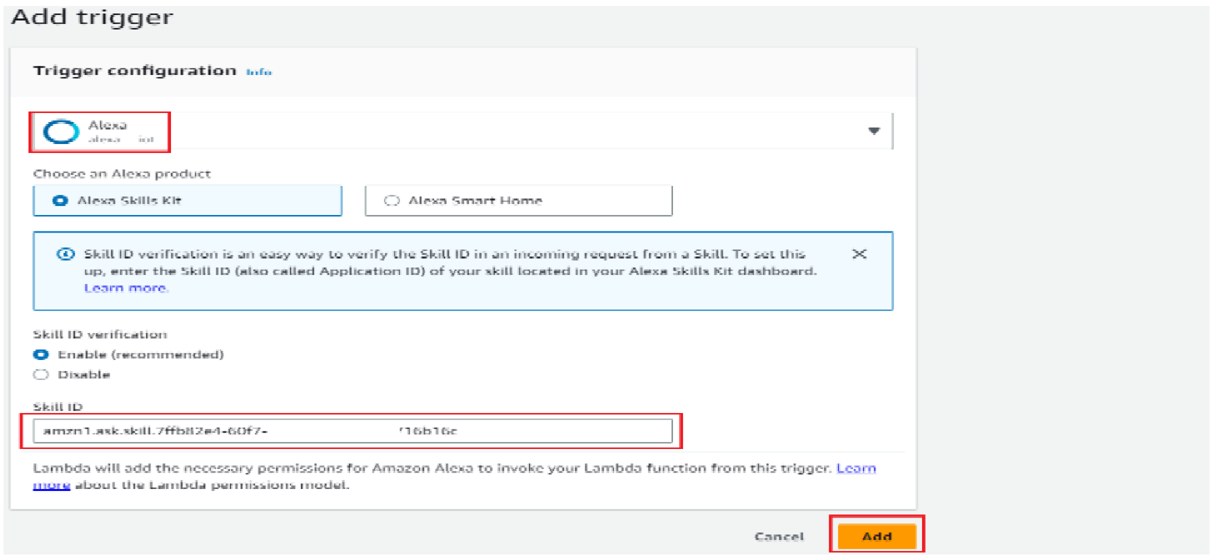


Fig. 9: upload completed [source: https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions]

3. Open Alexa Console, Copy the Skill ID, and paste it inside the Skill ID box. As depicted in Figure 10. Click the "**Add**" Button.


Add trigger

Trigger configuration Info

Alexa

Choose an Alexa product

Alexa Skills Kit [✓]   Alexa Smart Home

Skill ID verification is an easy way to verify the Skill ID in an incoming request from a Skill. To set this up, enter the Skill ID (also called Application ID) of your skill located in your Alexa Skills Kit dashboard. Learn more.

Skill ID verification

[✓ Enable (recommended)] Disable

Skill ID

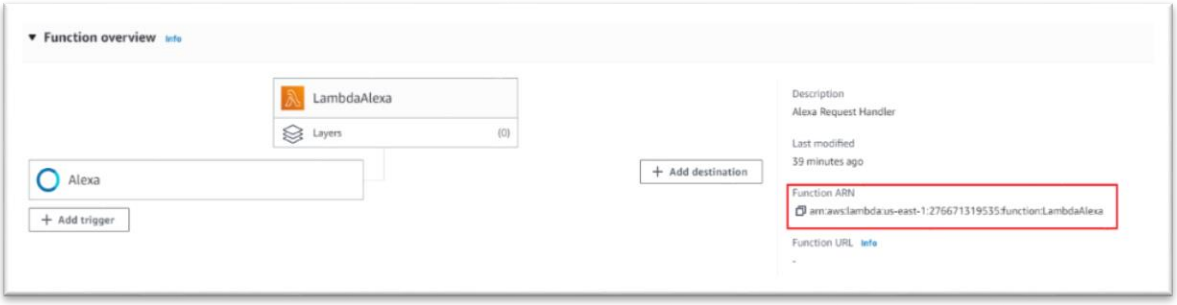
amzn1.ask.skill.7ffb82e4-60f7- '16b16c

Lambda will add the necessary permissions for Amazon Alexa to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Cancel   Add

Fig. 10: upload completed [source: Microsoft Visual Studio 2022 Community edition]

4. Copy the Lambda function ARN and paste it inside the Alexa default End Point. Save the endpoint Depicted in Figure 11.



Description
Alexa Request Handler

Last modified
39 minutes ago

Function ARN
arn:aws:lambda:us-east-1:276671319535:function:LambdaAlexa

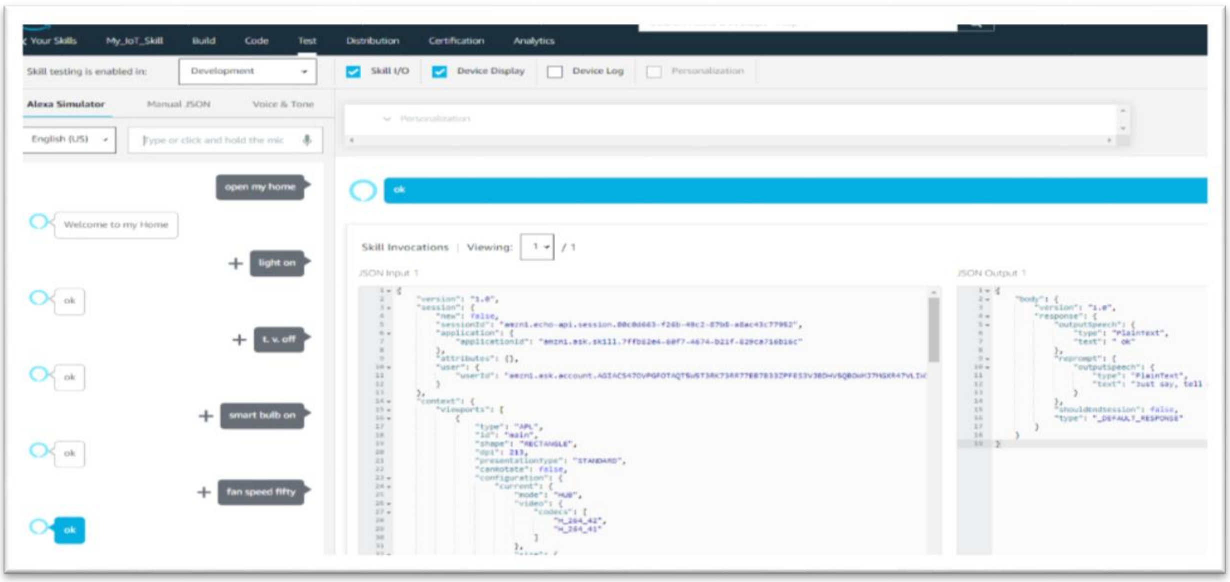
Function URL Info
-

Fig. 11: Lambda function ARN [source: https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions]

5. The link is now established. Close the browser.

Test Lambda function:

1) Create an Alexa skill. Lots of youtube videos are available. One video link is https://www.youtube.com/watch?v=COz51cENHO0.
2) Let us assume we have already developed skills. Now, Open the Alexa developer console, and click the “Test” button. We can send a command, either by voice or typing. The test starts with “Open” and then the invocation name. Here invocation name is “my home.” So we can send typing “open my home.” Then we can send our command according to the Alexa command handler. Some test commands are executed, depicted in Figure 5. 12.
User: open my home
Alexa: Welcome to my home
User: light on
Alexa: ok.



```
{
    "version": "1.0",
    "session": {
        "new": false,
        "sessionId": "amzn1.echo-api.session.00c0d663-f26b-40c2-87b6-a8ac43c77952",
        "application": {
            "applicationId": "amzn1.ask.skill.7ffb12e4-60f7-4674-b21f-829ca716b16c"
        },
        "attributes": {},
        "user": {
            "userId": "amzn1.ask.account.AGIACS47OVPGFOTAQT6WST3RK73RR77EB7833ZPFES3V3BDHVSQBOWH37HGXR47VLIW
        }
    },
    "context": {
        "viewports": [
            {
                "type": "APL",
                "id": "main",
                "shape": "RECTANGLE",
                "dpi": 213,
                "presentationType": "STANDARD",
                "canRotate": false,
                "configuration": {
                    "current": {
                        "mode": "HUB",
                        "video": {
                            "codecs": [
                                "H_264_42",
                                "H_264_41"
                            ]
                        },
```

```
{
    "body": {
        "version": "1.0",
        "response": {
            "outputSpeech": {
                "type": "PlainText",
                "text": " ok"
            },
            "reprompt": {
                "outputSpeech": {
                    "type": "PlainText",
                    "text": "Just say, tell
                }
            },
            "shouldEndSession": false,
            "type": "_DEFAULT_RESPONSE"
        }
    }
}
```

Fig. 12: alexa test interface [source: https://developer.amazon.com/]

6. RECOMMENDATIONS :

- This paper provides reference information to create Alexa-enabled IoT devices. The below-enlisted papers can be treated as a reference for IoT-based research work. All papers are practical-oriented.
- A good youtube link on Lambda in C# [11-16]: https://www.youtube.com/watch?v=BPWmaYJIjCk&t=632s
- The researcher interested in IoT using Sinric Pro can Navigate: https://www.srinivaspublication.com/journal/index.php/ijcsbe/article/view/1980.
- One good reference on Virtual IoT device Creation using Sinric Pro: https://www.srinivaspublication.com/journal/index.php/ijaeml/article/view/2093/817
- How to create IoT inside the AWS cloud: https://www.srinivaspublication.com/journal/index.php/ijcsbe/article/view/2283/875
- Physical IoT device creation using AWS and ESP module: https://www.srinivaspublication.com/journal/index.php/ijmts/article/view/2321/881
- Create Multiple IoT Device Controller Using AWS, ESP32, and C#: https://www.srinivaspublication.com/journal/index.php/ijaeml/article/view/2359/896

7. CONCLUSION :

The research paper on creating a Lambda function for an IoT device in the AWS cloud using C# has provided a comprehensive guide to creating a powerful and flexible solution for managing IoT devices. The paper has demonstrated the real-world application of Lambda functions for IoT devices. It has also highlighted the technical details of creating a Lambda function using C# and deploying it to the AWS cloud. The Function was able to handle a variety of inputs and produce the desired output, demonstrating its effectiveness in managing IoT devices. The research paper has provided a valuable resource for developers, engineers, and businesses looking to implement Lambda functions in their IoT projects. The paper has demonstrated the feasibility and effectiveness of this technology for managing IoT devices. It has highlighted the benefits of using Lambda functions, including cost-effectiveness, scalability, and improved performance.

REFERENCES :

[1] Das, A., Patterson, S., & Wittie, M. (2018, December). Edgebench: Benchmarking edge computing platforms. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)* (pp. 175-180). IEEE. Google Scholar↗

[2] Lee, H., Satyam, K., & Fox, G. (2018, July). Evaluation of production serverless computing environments. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (pp. 442-450). IEEE. Google Scholar↗

[3] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., ... & Suter, P. (2017). Serverless computing: Current trends and open problems. *Research advances in cloud computing*, 1-20. Google Scholar↗

[4] Lloyd, W., Ramesh, S., Chinthalapati, S., Ly, L., & Pallickara, S. (2018, April). Serverless computing: An investigation of factors influencing microservice performance. In *2018 IEEE international conference on cloud engineering (IC2E)* (pp. 159-169). IEEE. Google Scholar↗

[5] Pelle, I., Czentye, J., Dóka, J., & Sonkoly, B. (2019, July). Towards latency-sensitive cloud-native applications: A performance study on AWS. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)* (pp. 272-280). IEEE. Google Scholar↗

[6] Yadavalli, P. K., Mutyala, A. K., Palla, V. K., Pappu, A., & Prathipati, N. (2020). Smart IOT system for monitoring and controlling livestock parameters. In *International Conference of Advance Research & Innovation (ICARI)*. Google Scholar↗

[7] Baresi, L., Filgueira Mendonça, D., & Garriga, M. (2017). Empowering low-latency applications through a serverless edge computing architecture. In *Service-Oriented and Cloud Computing: 6th IFIP WG 2.14 European Conference, ESOCC 2017, Oslo, Norway, September 27-29, 2017, Proceedings 6* (pp. 196-210). Springer International Publishing. Google Scholar↗

[8] Gillam, L., Katsaros, K., Dianati, M., & Mouzakitis, A. (2018, April). Exploring edges for connected and autonomous driving. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 148-153). IEEE. Google Scholar↗

[9] Bastos, D. (2019, May). Cloud for IoT—A survey of technologies and security features of public cloud IoT solutions. In *Living in the Internet of Things (IoT 2019)* (pp. 1-6). IET. Google Scholar↗

[10] Ihejimba, C., & Wenkstern, R. Z. (2020, September). DetectSignal: A cloud-based traffic signal notification system for the blind and visually impaired. In *2020 IEEE International Smart Cities Conference (ISC2)* (pp. 1-6). IEEE. Google Scholar↗

[11] Chakraborty, S. & Aithal, P. S. (2022). How to make IoT in C# using Sinric Pro. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *6*(2), 523-530. Google Scholar↗

[12] Chakraborty, S. & Aithal, P. S. (2022). Virtual IoT Device in C# WPF Using Sinric Pro. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *6*(2), 307-313. Google Scholar↗

[13] Chakraborty, S. & Aithal, P. S. (2023). Let Us Create an IoT Inside the AWS Cloud. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *7*(1), 211-219. Google Scholar↗

[14] Chakraborty, S. & Aithal, P. S. (2023). Let Us Create aPhysical IoT Device Using AWS and ESP Module. *International Journal of Management, Technology, and Social Sciences (IJMTS)*, *8*(1), 224-233. Google Scholar↗

[15] Chakraborty, S. & Aithal, P. S. (2023). Let Us Create Multiple IoT Device Controller Using AWS, ESP32 And C#. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *7*(2), 27-34. Google Scholar↗

[16] Chakraborty, S. & Aithal, P. S. (2023). Let Us Create an Alexa Skill for Our IoT Device Inside the AWS Cloud. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *7*(2), 214-225. Google Scholar↗

****