Amazon Web Services and Microsoft Azure Edge Computing Platform Analysis

Mustafa Barez

Abstract

Cloud computing has been one of the leading topics in recent years as the transition from on-premise to off-premise cloud environments has given businesses far more capabilities. The recent years have seen a growth in cloud services as well as an expansion in the IoT devices operating in the cloud. As the volume of data processed in these devices increases, the study of edge computing has become more prevalent. Edge computing sends a section of computation to the edge of the network, thus restricting bandwidth and reducing latency. The present study looks to compare the edge computing services of the two industry-leading cloud providers, Amazon Web Services or AWS and Microsoft Azure, to see precisely where each platform excels in which criteria. The study presents an in-depth analysis of AWS IoT Greengrass and Azure IoT Edge with experiments conducted from documentation guides for each software. Experimental results show that developers should use AWS IoT Greengrass when set-up, security, and quality of documentation are important, but Azure IoT Edge when ease of use and code deployment are critical.

Keywords: *Cloud Computing, Edge Computing, IoT, AWS, Azure*

Introduction

The shift from on-premise to cloud-based services has led to major changes in the IT industry. Businesses are constantly migrating towards the cloud and the financial benefits provide strong justification. Cloud computing is defined as the delivery of computation, storage, and other IT services in a pay-as-you-go method through a cloud platform [3]. The cloud term specifically refers to the data center hardware and software and provides a number of benefits making it preferred over on-premise services. This includes a wide array of computing resources available on-demand, elimination of upfront expenses in hardware, and payments for services that are used. The services offered by cloud providers include: Software as a Services (SaaS), Infrastructure as a Services (IaaS), and Platform as a Service (PaaS), and can be offered to external businesses making it a public cloud [2], or for internal use, making it a private cloud [3].

One of the first major breakthroughs in cloud computing was the launch of AWS in 2006. Here Amazon reinvented the way businesses purchase computation and began offering services such as S3 and EC2 (Elastic Cloud Compute) which remain two of the most popular offerings to date. In 2010, Microsoft released a cloud platform named Azure, becoming a direct competitor to AWS, and offer a wide variety of IT services on its cloud platform. As both cloud providers began expanding, more services became available and allowed them to incorporate an IoT stack [4].

According to the International Data Corporation or IDC, IoT spending will surpass 1 trillion in spending by 2022. The IDC also estimates that total spending in 2019 was about 726 billion, justifying the acceleration of this technology. IoT is defined as a series of network smart devices that contain microchips, sensors, and wireless capabilities [5] and includes smart refrigerators, watches, health sensors, and vehicles.

With the rise of this technology, a segment of cloud computing known as edge computing has emerged and became prevalent over the years. Edge computing seeks to share computing resources, cloudlets, found at the edge of the network and in the proximity of the user [6]. This provides two advantages: restricting bandwidth to the geographic location of the user, and lowering latency for cloud-resistant resources and end-users [6]. With this, computation can be offloaded to the cloudlet for applications that are sensitive to latency and unavailable to resources found deep within the cloud [1].

Considering the concept of edge computing, the two major public cloud providers, AWS and Azure, have provided software capable of extending the cloud workloads to local devices [1]. For Amazon, the edge software is AWS IoT Greengrass which allows customers to build IoT device applications to collect and analyze data closer to the source of information and react with other devices on local networks [7]. Amazon also provides a Lambda Edge service which runs code closer to the geographical location of users and lowers latency. Microsoft Azure presents Azure IoT Edge, software that moves workloads to the edge of the network where devices spend less time communicating with the cloud, and more time reacting to local events [9].

Previous studies have compared the two cloud providers but were created during the time when IoT services were starting to be introduced. With AWS launching its IoT services in late 2015 and Azure in 2016, there has yet to be a study comparing the two major cloud providers on the edge services and their features. The following paper will look to conduct an in-depth analysis of the major edge computing software used by, Amazon and Microsoft, and provide scenarios where one application is more useful than the other. Here, the contributions of this paper are as follows:

- Comparative study of edge services between the two market-leading cloud companies, AWS and Microsoft Azure
- Analysis of specific features of each software including security, ease of use, quality of documentation, set-up, and code deployment
- Recommendations for which settings one software might be preferred over the other

Experimental Results

The experiments are divided as follows: AWS IoT Greengrass simulation, and Azure IoT Edge model. A Raspberry Pi 4 Model B, 2GB RAM was utilized for the AWS experiment. Subscriptions have been made for AWS and Azure in order to create resource groups to configure components and deploy services. An HP Pavilion Notebook PC, 64-bit Windows operating system, 12 GB RAM was used for cloud deployment purposes.

AWS IoT Greengrass

The process began by configuring the Raspberry Pi to be supported the Greengrass core. The dependency checker script was ran and the Greengrass group was created through the AWS subscription. The Greengrass core software was downloaded onto the Raspberry Pi through scripts found in the documentation page. Hardlink and softlink protection was enabled on the operating system to improve security measures and memory limits were set for lambda functions. This was the initiation process which was setting up the Raspberry Pi to communicate with Greengrass.

Following the set-up of the Raspberry Pi, the Greengrass core software was downloaded onto the HP Notebook. A Greengrass resource group was created through the AWS subscription to allow for components to interact with one another and security resources were downloaded through the command terminal. Raspberry Pi terminal decompressed the Greengrass core software to allow it to be used. The Amazon Trust Services or ATS root CA certificates were chosen in the server authentication section which allowed the Raspberry Pi to communicate with AWS IoT through MQTT messaging protocol. The Greengrass core was initialized on the device and a command line containing the PID number was ran on the terminal to confirm the presence of the software.

The next section consisted of testing Lambda functions on the Greengrass core. The Greengrass core SDK for python was downloaded onto the HP Notebook, unzipped, and the Greengrass SDK folder was copied into the Lambda function code and named Helloworld. The Lambda function deployment package was then compressed to a zip file and uploaded to a function created from the Greengrass core. A published version of the Lambda function was taken and an alias was created to manage code updates. The AWS IoT console was used to add functions to the Greengrass group. Afterwards, the Lambda function was configured to group settings where a subscription was added to allow the Lambda function to publish MQTT messages to AWS IoT and run tests. The tests were conducted to ensure the Lambda function was running on the Raspberry Pi. The following steps were extracted from the AWS IoT Greengrass documentation page found in the Developer Guide [7].

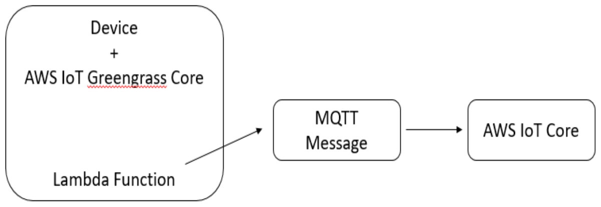

Figure 1. The image shows how a Lambda function communicates with the AWS IoT Core. A Core Device with a Greengrass group containing a Lambda function sends messages to the AWS IoT Core through MQTT protocol. The Lambda function first sends a message to the SDK of the Greengrass core, through MQTT (not shown) and the SDK sends a message to the AWS IoT Core using MQTT. MQTT is a protocol used to exchange data between devices. The following was taken from the Greengrass device setup documentation [11].

Azure IoT Edge

In order to initialize the process, an IoT Hub was created on the Microsoft Azure portal along with a resource group in order to manage all of the activities. This was completed through the Azure Cloud Shell, a CLI that simplifies tasks rather than having them manually completed. The device used to connect with the IoT hub was a linux virtual machine created through the CLI. A connection string was taken from the JSON output and was required for the IoT Edge runtime. Following this was the deployment of the IoT Edge runtime to the virtual machine through manual configuration. A module was then deployed that sent telemetry data to the IoT Hub and was the code being deployed to the virtual machine from the cloud. The following steps were taken from the Azure IoT Edge documentation page [10].

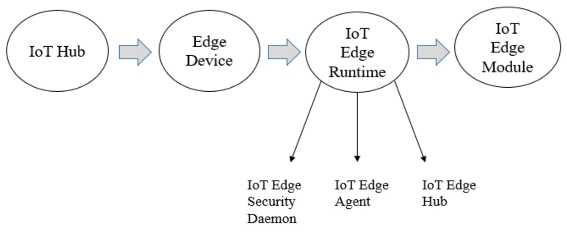

Figure 2. An overview of the Azure IoT Edge steps for deployment. The phase begins with the creation of an IoT Hub. Afterwards an edge device is registered into the hub. Once the device is registered, the IoT Edge runtime is installed into the device and contains three components, the IoT Edge Security daemon, the IoT Edge Hub, and the IoT Edge Agent [10]. The IoT Edge Security daemon starts the agent once the device starts, the IoT Edge Agent deploys and monitors the modules, and the IoT Edge Hub facilitates communication with the device and modules. The process is complete once the module is deployed onto a device.

Analysis of Results

Table 1. – Comparison between features of AWS IoT Greengrass and Azure IoT Edge

Criteria	**AWS IoT Greengrass**	**Azure IoT Edge**
Set-Up	+	-
Ease of Use	-	+
Security	+	-
Quality of Documentation	+	-
Code Deployment	-	+

The following table compares the features between AWS IoT Greengrass and Azure IoT Edge. The '+' symbol represents which software gains the edge for that feature according to the author's opinion and the '-' displays which software displays areas for improvement in comparison to the other.

The comparison in Table 1 between the software of both cloud providers displays AWS IoT as preferential in cases where the set-up, security, and quality of documentation are critical factors. In terms of set-up, AWS IoT Greengrass documentation reveals organized and thorough steps as to which devices are preferred and the process behind their initiation in edge computing experiments. With the details in the set-up process being compared to Azure IoT Edge where there are also tutorials; Greengrass is thought to be more explicit. Due to this, Greengrass would gain the edge in the set-up standards. Azure offers a simple way for deployment compared to AWS which is why its services may be preferred for simple experiments. However, in terms of security, AWS provides comprehensive certificates that enhance security measures. Compared to Azure, AWS provides more sophisticated security enhancements and is the reason behind its preference for security features.

The documentation guides for each software has AWS preferred over Azure based on the supplementary CLI commands, the detail in the command descriptions, and the troubleshooting steps. Azure documentation provides users with a basis for initializing edge based projects but lack of detail in areas such as the IoT Edge modules make it difficult for users to understand the full capabilities of the system. In terms of the code deployment, Azure provides a simplistic overview of how code is run compared to AWS and the Lambda system specifically which is why it is preferred in this area. The Lambda system provides a number of certificate and security measures that make it more tedious for developers to go through and is why an initial project concerning code deployment on the edge should start with Azure.

In terms of which edge computing platform would be preferred over the other, the final decision is based on the needs of the user. If a user is looking for a quick and simple application to deploy on the edge, Azure would be ideal due to its simplicity in both the set-up and code deployment for an intended device. If the user is to build edge applications that require a strong need for security and provide a detailed set-up process, then AWS should be the cloud platform of choice.

Future Works

The study leads to a number of future works that may be sought after. Due to the constant upgrades in the AWS and Azure services, one of the first areas is the expansion of regions covering IoT services. Currently, the AWS IoT regions are limited to sections in the US and due to this, the edge computing deployments are limited. By limiting the regions for deployment, users are only able to conduct experiments in given locations which would limit critical features measured such as latency. Another area of focus would be the amount of individuals involved in choosing which features they prefer for either software. This paper presents an individual's opinion on the software preferred for each criteria, however, future studies may involve numerous developers and IT professionals to gain a better understanding for areas in need of improvement. Along with this, studies should involve areas that include accessing other cloud applications through the edge and includes categories such as MySQL databases. Expanding on different cloud services would also ensure developers gain deep insights into which IoT edge services the cloud platforms excel at.

Conclusion

The two leading cloud platforms based on market share, Microsoft Azure and AWS propose many opportunities for expansion in edge computing. In this paper, experiments were performed to test out the edge computing software for both major cloud providers and the author noted which areas one would be ideal based on select criteria. The AWS IoT Greengrass core was set-up for the Raspberry Pi and highlighted the set-up, security, and documentation quality as the strengths compared to Azure IoT Edge where ease of use and code deployment are favoured. Future studies suggest testing features such as latency which gives developers a better sense of which edge computing software is the benchmark.

References

[1] Rob High. 2019. What is edge computing? *What Edge Computing Means for Infrastructure and Operation Leaders* 1 (October 2019).
[2] Keith Foote . 2017. A Brief History of Cloud Computing. (June 2017). Retrieved December 28, 2019 from https://www.dataversity.net/brief-history-cloud-computing/
[3] AWS. 2019. Types of Cloud Computing. (2019). Retrieved December 28, 2019 from https://aws.amazon.com/types-of-cloud-computing/

[4] Peter Newman. 2017. Amazon introduces host of new IoT services. (December 2017). Retrieved December 28, 2019 from https://www.businessinsider.com/amazon-aws-iot-services-2017-12
[5] Thierer, A., & Castillo, A. (2015). Projecting the growth and economic impact of the internet of things. *George Mason University, Mercatus Center, June*, *15*.
[6] Hadžić, I., Abe, Y., & Woithe, H. C. (2017, October). Edge computing in the epc: a reality check. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (p. 13). ACM.
[7] Diana Diaz. 2019.(2019). Retrieved December 28, 2019 from https://docs.aws.amazon.com/Greengrass/latest/developerguide/what-is-gg.html#gg-platforms
[8] Eanne Byrne. 2019. Lambda@Edge. (2019). Retrieved December 28, 2019 from https://aws.amazon.com/lambda/edge/
[9] Varun Puranik. 2019. Azure IoT Edge. (2019). Retrieved December 28, 2019 from https://azure.microsoft.com/en-ca/services/iot-edge/
[10] Kelly Gremban. 2019. Quickstart: Deploy your first IoT Edge module to a virtual Linux device. (2019). Retrieved December 28, 2019 from https://docs.microsoft.com/en-us/azure/iot-edge/quickstart-linux
[11] Diane Diaz. 2019. Quick Start: Greengrass Device Setup. (2019). Retrieved December 28, 2019 from https://docs.aws.amazon.com/Greengrass/latest/developerguide/quick-start.html