MONASH University

Monash Tutor Allocation System

Technical Report: Serverless Architecture

FIT3170: Software Engineering Practice

Submitted by Malaysia Team

Stuart Boey Chie Sheng - 29519985
Ahmed Nassar - 28901797
Teo Wei Han - 30242711
Cheong Chee Feng - 30222397
Ong Jeng Yue - 29637996

Submitted on January 21, 2021

Executive Summary

The following technical report is written to investigate the serverless architecture and how it can be used within an organization. The objective of this report is to outline in broad terms the serverless architectures, such as the costs, benefits and posed risks against organization when using serverless architecture and the types of projects that are suitable for serverless approaches. The report is intended to provide information and recommendation to aid the FIT3170 staff on deciding whether to use serverless architecture in future projects.

There are some uncertainties in predicting the cost of running a serverless architecture application as the cost varies among the service provider and also depends on the traffic characteristics of the application. Serverless architecture eliminates the need of provisioning, maintenance and administration of the servers and has a faster development rate and updates. Thus, it also reduces the labour and resource cost. However, the use of serverless architecture might lead to loss of control over the configuration. Organizations will also have no control over the security implementation. Problems will potentially arise during a vendor change and may force organizations to stick to one vendor. Projects that require less resource utilization are more focused in developing new features, have event-based programming , and follow irregular workloads are suitable for serverless architecture.

Based on our research, it is highly recommended to use serverless architecture for FIT3170 as students can be more focused on achieving the learning objectives of this unit instead of spending time on maintaining and creating servers. Besides that, introducing serverless architecture in teaching for other units is also being suggested. However it is better to introduce it to students in an early stage through year 2 units such as FIT2101, so that students will have a foundation knowledge on the concept of serverless architecture.

Table of Contents

1.0 Introduction

Serverless architecture, also known as serverless computing, is an evolutionary technology that has been gaining a lot of attention in the software architecture world recently. The popularity of serverless architecture has grown substantially that even the “Big Three” cloud vendors - Amazon, Google and Microsoft - are heavily invested in it [1].

At first, serverless architectures are application designs that incorporate third-party, cloud-hosted applications and services to manage side logic and state, which was previously described as “Backend as a Service” (BaaS). In serverless architecture, the server side logic for major portions of an application is still written by application developers, but it runs in ephemeral stateless compute containers provided by cloud providers. These stateless containers trigger on events and terminate after execution. This is also known as “Functions as a Service” (FaaS). Such architectures remove the need for a traditional always-on server component and enable the developer to build and run code without any worries about the server or hardware management [1]. Thus, in a serverless application, provisioning, maintenance and administration of the servers by developers is not needed for each of the components that are involved in the architecture.

This report will be documenting the findings and research about the serverless architecture, including some case studies of some successful companies. The investigation will be focusing on understanding the differences of serverless architecture and other alternatives such as container and microservices architecture, historical trend of serverless architecture in the software architecture industry, the cost of running it, its benefits and risks towards an organization. Lastly, based on the researched information, we will provide recommendations of introducing serverless architecture in others units in Monash University.

2.0 Research Limitation

2.1 Lack of previous research studies on the topic

Serverless architecture might be a new topic to be discussed by students who enrolled in FIT3170. Students have to spend more time in doing the research and ensuring those information collected are consulted from appropriate sources. Information included in the report has to be compared and studied among different sources to ensure it can be included in the report.

2.2 Only well-known serverless architecture providers being used

Most of the serverless architecture providers on market are well-known by the public such as AWS Lambda by Amazon Web Services, Azure Functions by Microsoft, Google Cloud Functions and IBM Cloud Functions, which are known as major group providers and are used by a lot of companies. There is not much information about those minor groups providers such as Parse and Back4app.

3.0 General Review

3.1 What is Serverless Architecture

Google Cloud Platform[2] defines serverless as being able to have "no upfront provisioning, no management of servers, and pay-what-you-use economics for building applications". Mike Roberts[1] provides two overlapping descriptions for serverless. The first description for serverless are "applications that significantly or fully incorporate third-party, cloud-hosted applications and services, to manage server-side logic and state" which he also called Backend-as-a-service(BaaS). According to CloudFlare[3], BaaS is a cloud service model where all behind the scenes aspects such as user authentication, database management, remote updating, and push notifications (for mobile apps), as well as cloud storage and hosting were outsourced. However, CloudFlare [3] also mentioned that although there are overlaps between serverless architecture and BaaS, there are significant operational differences between applications built using BaaS and a true serverless architecture. The second definition by Mike Roberts [1] describes serverless as applications whose server-side logic is written by the application developer, but is run in stateless compute containers that are event-triggered, ephemeral, and fully managed by a third party.

Serverless architecture is an architecture that makes use of serveless-ness by adopting its principles. Amazon Web Services (AWS) [4], one of the leading providers of serverless computing, describes serverless architecture as "a way to build and run applications and services without having to manage infrastructure". Azure [5] promotes serverless architecture by saying we can "build apps faster without managing infrastructure" when using it.

Serverless architecture should not be confused with Functions-as-a-Service (FaaS) which is actually a subset of serverless. According to IBM[6], "serverless is focused on any service category, be it compute, storage, database, messaging, api gateways, etc. where configuration, management, and billing of servers are invisible to the end user. FaaS, on the other hand, while perhaps the most central technology in serverless architectures, is focused on the event-driven computing paradigm wherein application code, or containers, only run in response to events or requests."

Although there seems to be various definitions to what serveless and serverless architecture is, one thing all the definitions share in common is that there is no need for users to manage the servers. In other words, serverless architecture does not mean doing away with servers but rather that the management of the servers are conducted by a third-party such as AWS, Google Cloud, Azure or other provisioners of serverless computing.

3.2 Differences and Similarities of Serverless Architecture to Alternatives

There are various other alternatives to a serverless architecture. One of them is microservice architecture. Microsoft's documentation [7] defines it as consisting of "a collection of small, autonomous services where each service is self-contained and should implement a single business capability". The similarities between serverless and microservice architecture is that both were developed to help break up a monolith application into more manageable and isolated pieces [8].This made it easier to work on, detect bugs, update and maintain when an application is too big.

Both forms of services provide quicker updates since the isolated microservice or serverless function can be deployed instead of the whole application [8].

According to ByteAnt [9] there are a few differences between a microservice architecture and a serverless architecture. One of them is, for microservice architecture, all relevant libraries hosted on cloud servers are accessible by programmers. This in turn allows the deployment of various functional modules, protocols and API such as JSON, RESTful, AMQP, SOAP and others. Therefore, providing more flexibility to the programmer. On the other hand, serverless architecture provides the execution of cloud-based working environments on demand rather than full access. This also means that microservice architecture could be more expensive as even when the microservice is sitting idle, the cost of the hosting the microservices in servers still has to be paid for.
Serverless architecture provides the means to only run a serverless function when required, folding up promptly, once the task is completed. This translates to less uptime which is in turn cheaper when the service is not required all the time.
Another difference is that in a serverless architecture, the scaling is done automatically by the cloud provider whereas this is not applicable for a microservice architecture. Besides that, serverless architecture uses functions which perform a service and returns the result to the requester [10]. Conversely, microservices can consist of more than one function. Thus it can perform various tasks. In other words, one microservice can equate to multiple serverless functions.

Another alternative to serverless architecture is containers. Docker [11], an industry leader for container services, defines containers as “a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another”. In comparison to applications hosted on servers or virtual machines, both the serverless architecture and containers allows developers to create applications with more flexibility and much less overhead to deal with [12].

However serverless architecture and containers have far more differences than similarities. Containers are applied on a particular machine and use its operating system whereas the serverless architecture runs on servers with no fixed machine assigned for a function or application[10]. Besides that, the number of containers deployed needs to be determined in advance. This is in contrast to serverless architecture where the backend scales automatically by the cloud providers [12]. Another difference is that containers need to be hosted on a server which is constantly running even when the container is not in use. This translates to higher cost. This situation does not arise in serverless architecture as the application code is only running in the server when called upon [12]. In terms of testing, serverless architecture has a problem due to the difficulty in replicating in a local environment. Containers do not have this issue as they run the same no matter what environment they are placed in [12]. Containers are also more flexible than serverless architecture as a programmer could set policies, manage resources, and security to their liking. These options are either much more limited or managed entirely by the cloud providers in a serverless architecture.

3.3 Relation of serverless architectures to historical trends in software architecture

Software architecture is the fundamental structure of a software system and how it is created. Much like how architects need plans and blueprints to build buildings, applications in this day and age have grown to a size where making them without a plan is infeasible and doomed to fail. We make use of software architectures to serve as blueprints for how we envision the application and its system, laying out all the required tasks and issues. Based on research about the history of software architecture by Eoin Woods [13], it has been 25 years since software architecture was recognised as a discipline. Over the years, software architecture has changed as we know it.

In the 1980s, most software systems were big and monolithic, predominantly running on single computers. Back then, architecture was mostly handled by vendors who inherited them from the platform the programs were on. The focus for programmers was on these “programs”. Moving into the 1990s, distributed systems came into trend, with batch processing being the norm. Servers in enterprises became much bigger, creating much more architectural decisions than before. However, the architectural decision was still mostly decided by the vendors who provided the tools for development.

Then, as the Internet started becoming widespread, the demand for Internet-connected devices rose exponentially, drastically changing software architecture to be able to support unpredictable and challenging non-functional qualities. The need for network load-balancers for scalability and firewalls for security became apparent. And with the Internet, came many architectural patterns that we know of and still use to this day such as Model-View-Controller(MVC), Peer-to-Peer(P2P), Representational state transfer (REST), Event-driven architecture and many more.

By the 2010s, the Internet had become a basic service that we use in our everyday life. This once again created a change where applications need to always be available whenever and wherever. Application became Internet-native systems rather than just being connected to it. This further complicated application development with the use of open-source components, remote Internet connected services and custom. The system also had to have flexible architectural styles which gave rise to microservice architectures and the use of containers. However, not long after, cloud computing was introduced and with it came the serverless architecture which followed the need of allowing for flexible architectural styles. Platform-as-a-service(PaaS) started becoming a major concern for vendors to help abstract away the complexity of architectural decisions for programmers. The first usage of the word “serverless” was back in 2012 but it only became popular in 2015 after AWS Lambda was launched[1].

4.0 The Use of Serverless Architecture

4.1 Cost of Serverless Architecture

The way that operational cost of running a serverless architecture application is calculated is vastly different from how we calculate the cost of a more traditional application hosting method such as using virtual machines.

Virtual machines such as AWS EC2 on-demand instances are priced for the compute capacity used by the hour or second [14]. Although it is priced for compute capacity used, it is billed based on allocation, not the actual use. While for serverless computing services such as AWS Lambda, it charges based on the duration that the user's code is executing, the resources assigned to the code during execution and the number of times the code is executed [15].

To understand the cost of using serverless architecture, we can investigate the cost of using Function-as-a-Service(FaaS) from the 4 big players in the market, namely AWS Lambda, Microsoft Azure Functions, Google Cloud Functions and IBM Cloud Functions [16]. The price of these services are summarized below [15], [17]–[19]. Memory time (GB-second) is calculated by multiplying the amount of memory allocated to the function and the duration of the execution of the function. CPU time (GHz-second) is calculated by multiplying the speed of the CPU(GHz) provisioned to the function and the time of execution of the function, this metric is unique to Google cloud functions between the 4 providers.

Table 0.1.1: Pricings of different providers

	AWS	Azure	Google	IBM
Free request per month	1,000,000	1,000,000	2,000,000	
Free memory time per month (GB-second)	400,000	400,000	400,000	400,000
Free CPU time per month (GHz-second)			200,000	
Price per request (US$)	0.0000002	0.0000002	0.0000004	
Price per GB-second (US$)	0.0000167	0.0000160	0.0000025	0.0000170
Price per GHz-second (US$)			0.0000100	

As you can see, different providers price their services differently, AWS and Azure both have similar cost structure with the slight difference of price per GB-second. Although Google has more free requests per month, it takes into account the CPU time used by the function. Lastly, IBM does not charge for per request, but the price for memory time is slightly higher when compared to AWS and Azure. It is clear that the cost of using serverless computing greatly depends on the service provider.

In real world applications, it is hard to estimate the amount of memory time and CPU time that will be used by the application. The amount of requests that an application will receive in a certain amount of time is uncertain. Besides, serverless auto scale feature means that the cost of any other services that are used together with the serverless compute service scales as well [20]. Therefore, it is difficult to predict accurately the cost of running a serverless architecture application. What we can do instead is compare the economics of serverless computing to a more traditional virtual machine implementation.

In a research by Owen Rogers from 451 Research, he compared the cost of using AWS EC2 t2 instances with 512MB and 1GB memory and AWS Lambda for a month, where the number of executions were not taken into account. It is found that if the Lambda code is executed for more than an aggregated 500,000 seconds, hosting the code on a small t2 instance has a lower direct cost. That is, if the lambda is running for more than 20% of the month, it makes more sense to host it on a t2 virtual machine [21].

In a nutshell, the cost of running a serverless architecture application is highly dependent on the traffic characteristics of the application. Much more analysis needs to be done before the development of an application to ensure that it makes economic sense to be using a serverless architecture.

4.2 Benefits of Serverless Architecture

There are a number of benefits that using a serverless architecture brings, we will go through a few of them and a general description of what make these benefits possible.

- **Reduced operational responsibilities**
  With the adoption of serverless architecture, the development team does not need to provision and manage servers, VMs or containers as their basic computational building block [22]. Besides, they do not need to manage operating systems, patch levels, database version upgrades and more. Instead, the development team can focus more on the business logic. With the use of Backend-as-a-Service(BaaS), developers have less logic to define, develop, test and deploy, thus reducing responsibilities of the developers [23].

- **Reduced cost**
  Serverless architecture can help reduce a few types of cost. The first one would be the labor cost as a result from the reduction of operational responsibilities as discussed in the point above, which reduces operations work and therefore labor cost [23]. Apart from that, serverless architecture can reduce the resource cost of the project as we have investigated in section 4.1 in certain cases, the use of serverless architecture is not a guarantee of decrease in resource cost and is highly dependent on the characteristics of the application as mentioned in the conclusion of this study [24].

- **Increased flexibility of scaling**
  Most serverless computing services have the ability to auto scale to the requirement of the application. Take AWS lambda for example, the functions automatically scale to support the rate of incoming requests without requiring the developers to configure anything and provide consistent performance [15]. This ease of horizontal scaling is beneficial to the type of application that has a bursty workload as developers do not need to provision more resources before a burst of workload.

- **Faster deployment and updates**
  After development, developers do not need to upload the code to the servers or do any backend configuration for the application to work. They can upload all the required functions as zip files to the serverless computing provider for the application easily for an application to be up and running. Besides, update, fix or new features can be added to a working application easily as developers do not need to update the whole system at once but they can update one function at a time [25].

4.3 The Dependence Reduction or Elimination of DevOps

To understand the relationship between DevOps and Serverless architecture, we must first try to define what DevOps is. Quoting from a paper [26], "DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organizations, in order to operate resilient systems and accelerate delivery of changes.".

Essentially, DevOps emphasizes the collaboration of the development team and operations team. The operational responsibilities of the team are reduced with the use of serverless architecture resulting in serverless architecture and is considered NoOps where there is no need for the collaboration of the development team and operations team or even an operations team at all [27].

In practice, even though serverless architecture removes the need for load balancing, autoscaling, high-availability and security maintenance of compute infrastructure, there is still more to be done to ensure the reliability, security, performance and more aspects of the application is up to standard [20]. For example, support, monitoring, deployment, security and networking are still relevant operational responsibilities when building an application with serverless architecture [23].

Therefore, serverless architecture does not entirely eliminate the need of DevOps but may reduce it to a certain extent. Thus, DevOps is still needed so that the application functions as intended.

4.4 Types of Projects suited for Serverless Architecture

The types of projects that are suited to serverless architecture are projects that can take advantage of the benefits of using such architecture.

4.4.1 Project with bursty and irregular workloads

Serverless compute service that allows for autoscaling is great for applications that have bursty and irregular workloads. Functions that can scale to the amount of requests automatically means that the developers do not need to provision additional resources ahead of time, they do not need to predict the amount of traffic that is coming in and out of the system. In fact, in a study [28], out of 89 applications that use serverless architecture surveyed, 84% of the applications have a bursty workload.

4.4.2 Project with low resource utilization

If the traffic of the project is not as high, resulting in a low utilization of resources such as server compute time, if might be more economical to switch to a serverless architecture as serverless architecture has a pay-per-use model, the team do not need to waste money to keep the server up when it is not used. The team will need to find the breakeven point where using a serverless compute service is cheaper than a virtual machine instance.

4.4.3 Project with event-based programming

With the stateless nature of serverless functions, it is suitable to be used in event-driven applications. The functions can be triggered when some event occurs, in case when a failure occurs, the functions can be executed again without side effects as they are stateless. Besides, if the function is not triggered, the function will automatically scale to zero, which reduces the resource used [22].

4.4.4 Project that focuses on features development

The ease of deployment using a serverless architecture as discussed in section 4.2 means that the team can better focus on the development of new features for the application. With no concern about maintaining a server, development and deployment of features can take place faster.

4.5 Risks of Serverless Architecture to an Organization

While the use of serverless architecture may bring benefits to an organization that utilizes them, it also comes with risks that an organization must be prepared to mitigate.

4.5.1 Loss of control

The level of abstraction that allows for the convenience of serverless architecture is also what causes the users to lose control over some aspect of the application that uses serverless architecture. Firstly, it is a loss of control over the configuration of the system. With FaaS, users do not have the ability to configure everything to their liking, the same goes for BaaS, the underlying architecture is configured by the service provider and users do not have the ability to configure it [23].

With the loss of configuration of the system may come the loss of control over the performance of the system. Since serverless platforms do not allow users to access the hardware, operating system and runtime, it is hard for the users to optimize their code to run at peak performance on the serverless architecture, it will also be hard to predict the performance of the functions. Performance of BaaS can be unpredictable as well where performance of on request is inconsistent with the performance of another request.

When a problem arises in a serverless system, the users do not have the ability to resolve the problem themselves, they can only resolve problems that are related to their code. For anything else, they must rely on the platform provider to solve any underlying problems. This may cause extra downtime to the services of the application.

Lastly, organizations do not have control over the security implementation of the serverless compute provider, users can only control aspects of the security that the platform provider has given access to the user, they will not have deeper control of the security so it may not meet the security requirements of an organization [23].

4.5.2 Vendor lock-in

The implementation of serverless features will be different from one vendor from another, therefore, if an organization wants to switch vendors, it is very likely that they will need to update their operational tools, code or even design and architecture of the application. If other services from the same vendor are used that are coupled with the serverless feature, it might mean that these implementations will need to be moved as well when changing to a different vendor. These complications during a vendor change may cause vendor lock in where an organization is forced to use services provided only by one vendor [1].

4.5.3 Multitenancy problem

Multiple instances of software for different customers are run on the same machine, or even in the same hosting application by a serverless compute provider. This may lead to some security concerns such as one customer might be able to see another's data. Besides, issues with robustness is possible where an error from one customer causes a failure in another customer's software. The performance of one customer's software might be affected by a high traffic customer on the same machine. These issues are not expected with mature serveless vendors but organisations have to be careful to avoid these problems [1].

5.0 Mini Case Studies

Following the investigation and research on serverless architecture, studying successful companies implementing such architectures can help turn theories and hypotheses into a potential realization, possible by understanding why such companies have adapted to serverless architecture, the benefits they have gained from such a thing, and nonetheless, the cost they have endured. Below, three successful companies, Netflix, Coca-Cola, and iRobot, have adapted serverless architecture [29], and as a result, will act as the subject of this mini case study.

5.1 Netflix

For most, Netflix needs no introduction. It is a worldwide video streaming service that streams films, TV productions, documentaries, and many more. With over 160 million users registered by the end of 2019 [30], managing all these account information on a server can become daunting. Netflix decided to migrate to AWS serverless cloud architecture, but the process was demanding. In fact, Vice President of Cloud Computing and Platform Engineering at Netflix, Yury Izrailevsky, has written about the process, and in conclusion, mentions that it took 7 years for Netflix to successfully complete the migration [31].

Nevertheless, the choice to migrate proved a benevolent one, with noticeable improvements on how they run the business. One good use is the core functions provided by AWS Lambda. One of these functions helps split the files uploaded into 5-minute chunks that gets encoded into parallel streams that Netflix needs. Without this serverless architecture solution provided by AWS Lambda, Netflix workforce team would have needed to multiply in size to enable such a smooth and fast delivery [32]. Another benefit of migrating to a serverless architecture is the ability to use the backup feature for their system. With thousands of files being updated on a daily basis,it is crucial to check the validity and integrity of these files. In case anything fails, backtracking to the source of the problem is easily achievable. Finally, in respect to security, AWS Lambda becomes responsible for validating instances of processes to ensure they are configured and constructed in accordance to system's rules and regulations.

Aside from all the features provided by AWS Lambda serverless cloud services, Netflix has also reported enjoying the general benefits of moving to such an architecture, such as the better scalability, the reduction of cost, and the time spared in which the company can focus on pushing more content to the market and improve their user experience [33].

However, all these perks come at a cost, and for Netflix, the cost was the time spent to migrate to such a serverless architect due to the learning curve, which has created some challenges and unwanted issues along the way, and have unwillingly moved these issues to their new host, AWS Lambda, before ultimately resolving them [31].

In conclusion, the decision to move to a serverless architecture for Netflix was a wise choice, but a bit daunting to finally realize.

5.2 Coca-Cola

Coca-Cola is known for its production of the cola flavored soft drinks, so it might arise the question on how they are related to such a case study. Nevertheless, Coca-Cola relates to this study as their vending machines have an integrated communication system with the Coca-Cola headquarters. This communication allows the monitoring of the vending machine to report low stocking or any malfunctions [34].

Coca-Cola mainly benefited from moving to a serverless architecture in the financial area. Coca-Cola reported that before adapting AWS lambda, they used to pay around 13,000 USD to run and maintain their machines in the United States. After moving to a serverless architecture solution, the cost dropped down to less than 4,500 USD per year.

This was possible as they have offloaded all the business logic behind a transaction to AWS Lambda, and as a result, both the customer and the company experience a smoother completion of a transaction as shown in the diagram below [34].

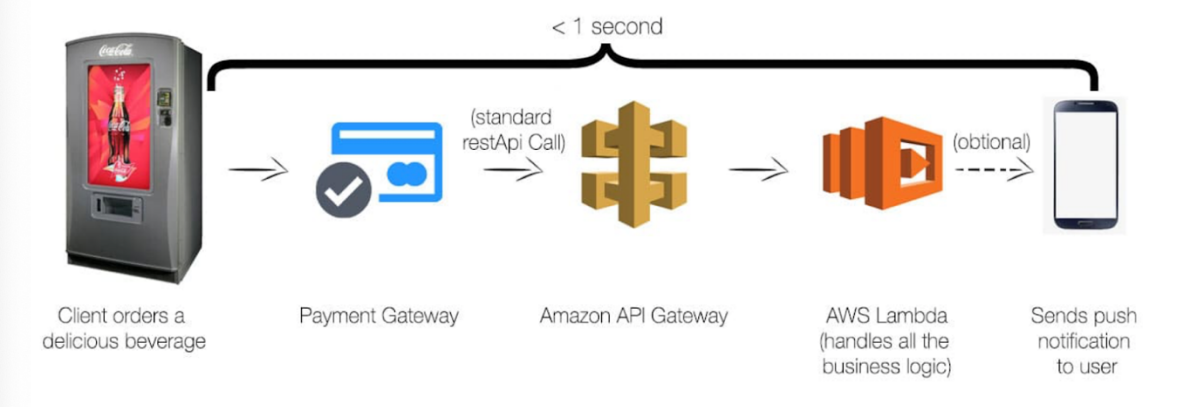

Figure 5.2.1: Coca-Cola Vending Machine Transaction

Of course, with the deep learning curve the AWS Lambda came with, Coca-Cola is still dealing with both the server-dependent and serverless vending machine as they train and help their team adapt to this new technology [34]. Once again, the deep learning curve proves to be one of the main costs and obstacles standing in the way of enterprises and the serverless architecture concept.

The main point behind including Coca-Cola in the mini-case studies is to show that not only heavy technology based companies can benefit from moving to a serverless architecture, but also non-tech focused companies can do so as well, proving that serverless architecture can truly make a difference in how businesses can run and operate.

5.3 iRobot

As the name suggests, iRobot is a leading consumer robot company that designs and builds robots to empower people to do more inside and outside their homes. They are primarily focusing on building house-cleaning smart robots [35].

Being a global company, their main challenge was to be able to deal with the large number of connection requests sent by the APP of vacuum robots users, creating a large volume of traffic through the app. After the company has decided to move to a serverless Architecture by choosing AWS IoT and AWS Lambda, iRobot is now able to process trillions of messages between billions of devices and AWS, which provides them a connectivity layer between the smart robots and the iRobot cloud platform [36]. Additionally, by using the serverless architecture, iRobot was able to keep the cost of the cloud platform relatively low, which in returns, eliminates the need for subscription services and allows them to manage the solution with a handful of people instead of having a large team focusing on running the server locally. As with other businesses, this allows the company to focus on customer needs rather than the tedious operation management procedures, allowing the incrementing of business value [36].

By adapting the serverless architecture provided by AWS, the company also offloads the burden of having the consumer calibrating the robots every now and then by making use of the smart functions provided by AWS in terms of maintaining a steady connection between the associated device and the cloud platform. This also allows the company, which is mainly a hardware vendor, to easily have a reliable software operation and support without the necessity to hire a team of software engineers and Information technology technicians to run and maintain the cloud services [36].

Serverless architecture proves once again that it is an ideal solution for businesses that are not software focused, yet want to integrate into the world of networking and cloud-based communication. By offloading all the server-related technical operations from the business company, businesses can continue to grow their business value and focus on their customer requirements without having to accept a poor performing simple server structure or investing a large fortune in building their own modern server structure. Instead, they can adapt to a serverless architecture.

Although not mentioned, it can be inferred that serverless architecture might not be ideal for tech-businesses that revolve around micro-level customization as the functions and features provided by serverless architectures might be generally standardized. For example, iRobot is mainly concerned about having their devices connected to the cloud for the enabling of smart features, remote control, and analysis reports [37]. Other companies might be more interested in establishing various limitations and constraints on how different devices should connect to the server, in which case, pre-existing serverless architectures might not be the ideal solution. Afterall, it all goes back to the requirements of the enterprise and how flexible they are.

6.0 Conclusion

6.1 Summary of Findings

To summarize the investigation that has been carried out and outlined above, this report incorporates information regarding serverless architecture as a whole and its relation to DevOps. First and foremost, a clear definition of what a serverless architecture is and how it differs from microservice architectures and containers is documented. Subsequently, this report also comprises investigation done regarding the historical trends in software architecture along with how and when serverless architecture came about. We have also carried out an investigation in regards to the cost, benefits, risk and types of project that are suited to a serverless architecture. The abstract of benefits, risks and types of project that suits the utilization of a serverless architecture is as such.

Table 0.1.1: Benefits and Risks of Project Types

Project Types	Benefits	Risk
Project with bursty and irregular workloads	Reduced operational responsibilities	Loss of control
Project with low resource utilization	Reduced cost	Vendor lock-in
Project with event-based programming	Increased flexibility of scaling	Multitenancy problem
Project that focuses on features development	Faster deployment and updates	-

Last but not least, we have also done a few case studies regarding the use of serverless architecture.

6.2 Recommendation of Usage for FIT3170

In terms of recommendation for usage of serverless architecture for FIT3170, we think it is important that we revisit the handbook for FIT3170 [38]. In the overview section, the handbook states that, “The project will be managed through a heavyweight process model such as the Spiral Model, to ensure students are exposed to a representative example of both heavyweight and lightweight processes (which are covered in FIT2101) through the BSE core. For the first time in their degrees, students will solicit and document requirements from client proxies who are not IT professionals. This builds their communication skills with other stakeholders in preparation for the industry-based project or IBL.”. This means that the focus of the unit is placed on the process of self-managing teams, managing requirements and understanding the common software process model. With the benefit of reduced operational responsibilities in the use of serverless architectures, students will be able to pivot their focus into the achieving the learning outcomes of this unit rather than misusing their time on developing and preserving servers. As a team, we felt that we were able to learn a variety of new things, from managing requirements to communication between teams and clients. This is due to the fact that we are not required to spend time creating and maintaining servers for our application, allowing for us to expand our time elsewhere. That being said, if suited, the use of serverless architectures in future FIT3170 projects is highly recommended to ensure that the time spent by students would be to achieve the learning outcomes that has been set by Monash University and obtain the full benefit of enrolling for the unit.

6.3 Recommendation of investigation for FIT3170 and other units

As reported in Section 3.3 above, Platform-as-a-service(PaaS) has started becoming a major concern for vendors to help abstract away the complexity of architectural decisions for programmers. In addition to that, German company Statista has reported [39] that the market value of Platform-as-a-service(PaaS) is expected to grow to 69 billion US$ in the year 2022. This means that Platform-as-a-service(PaaS) is expected to garner more attention from colossal industrial companies. As software engineering students, we are always learning, putting ourselves in unfamiliar positions to learn as much as possible to prepare ourselves for the real world industries. Seeing that Platform-as-a-service(PaaS) is an industry with such a positive growth rate, it is definitely recommended that serverless architecture is discussed in other units such as FIT3077 as it is a unit that has emphasis on software architecture and design. However, exposure of serverless architecture could even begin as early as FIT2101- Software engineering process and management. By briefly going through serverless architectures in FIT2101, it equips the students with awareness regarding serverless architecture which could assist them in making their decisions in FIT3170. The addition of the topic serverless architecture in the units mentioned above will definitely benefit the student, in terms of course work and in terms of preparation for the real world as well. Therefore it is highly recommended for the teaching team to consider discussing serverless architectures in other units.

References

[1] Mike Roberts. “Serverless Architectures”. MartinFowler.com. https://martinfowler.com/articles/serverless.html (Accessed on Jan. 8,2021)

[2] Google Cloud. “Serverless Architecture”. Google Cloud. https://cloud.google.com/serverless/whitepaper (Accessed on Jan. 8,2021)

[3] CloudFlare. "What is BaaS? | Backend-as-a-Service vs. serverless". ClourFlare. https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/. (Accessed on Jan. 20, 2021).

[4] AWS. “Building Applications with Serverless Architectures”. AWS. https://aws.amazon.com/lambda/serverless-architectures-learn-more/

[5] Azure. “Azure Serverless”. Microsoft Azure. (Accessed on Jan. 8, 2021). https://azure.microsoft.com/en-us/solutions/serverless/

[6] The IBM Cloud Education. "FaaS (Function-as-a-Service)". IBM. https://www.ibm.com/my-en/cloud/learn/faas . (Accessed on Jan. 20, 2021).

[7] Microsoft Docs. “Microservices architecture style”. Microsoft Docs. https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices (Accessed on Jan. 8,2021)

[8] Eamon Tang. “Microservices vs. Serverless”. fathom. https://fathomtech.io/blog/microservices-vs-serverless/ (Accessed on Jan. 8,2021)

[9] Valeriy Ilchenko. “Serverless vs. Microservices Architecture: What Does The Future of Business Computing Look”. ByteAnt. https://www.byteant.com/blog/serverless-vs-microservices-architecture-what-does-the-future-of-business-computing-look/ (Accessed on Jan. 8,2021)

[10] Joydip Kanjilal. “When should I choose between serverless and microservices?”. TechTarget. https://searchapparchitecture.techtarget.com/answer/When-should-I-choose-between-serverless-and-microservices (Accessed on Jan. 8,2021)

[11] Docker. “What is a Container? A standardized unit of software”. Docker. https://www.docker.com/resources/what-container (Accessed on Jan. 8,2021)

[12] CloudFlare. “Serverless computing vs. containers | How to choose”. CloudFlare. https://www.cloudflare.com/learning/serverless/serverless-vs-containers/ (Accessed on Jan. 8,2021)

[13] Eoin Woods, “Software Architecture in a Changing World”, *IEEE Software*, vol. 33, no. 6, pp. 94-97, Nov. 2016. doi: 10.1109/MS.2016.149 (Accessed on Jan. 8,2021)

[14] AWS. “On-Demand Pricing”. AWS. https://aws.amazon.com/ec2/pricing/on-demand/ (Accessed on Jan. 12,2021)

[15] AWS. “AWS Lambda”. AWS. https://aws.amazon.com/lambda/ (Accessed on Jan. 12,2021)

[16] QYResearch. Global Serverless Computing Market Size, Status and Forecast 2020-2026. QYResearch. Jan. 13 2021. [Online] Available: https://reports.valuates.com/market-reports/QYRE-Auto-27W1601/global-serverless-computing

[17] Microsoft. “Azure Functions pricing”. Microsoft. https://azure.microsoft.com/en-us/pricing/details/functions/ (Accessed on Jan. 13,2021)

[18] Google. “Google Cloud Functions pricing”. Google. https://cloud.google.com/functions/pricing (Accessed on Jan. 13,2021)

[19] IBM. “IBM Cloud Functions pricing”. IBM. https://cloud.ibm.com/functions/learn/pricing (Accessed on Jan. 13,2021)

[20] A. Eivy and J. Weinman, "Be Wary of the Economics of "Serverless" Cloud Computing," in IEEE Cloud Computing, vol. 4, no. 2, pp. 6-12, March-April 2017, doi: 10.1109/MCC.2017.32.

[21] Owen Rogers. “Economics of Serverless Cloud Computing”. 451 Research.Jan. 13 2021. [Online] Available:http://americanmarvels.com/cloud/economics.pdf

[22] Baldini I. et al. (2017) Serverless Computing: Current Trends and Open Problems. In: Chaudhary S., Somani G., Buyya R. (eds) Research Advances in Cloud Computing. Springer, Singapore. https://doi.org/10.1007/978-981-10-5026-8_1

[23] M. Roberts and John Chapin. “What is Serverless?”. Sebastopol, CA, USA: O’Reilly Media, Inc., 2017 ch.3, pp. 19-25 [online]. Available: https://symphonia.io/what-is-serverless.pdf

[24] Álvaro Alda Rodríguez et al. “Economics of 'Serverless'”. BBVA. Jan. 13 2021. [Online] Available: https://www.bbva.com/en/economics-of-serverless/

[25] Cloudflare. “Why Use Serverless”. Cloudflare. https://www.cloudflare.com/learning/serverless/why-use-serverless/ (Accessed on Jan. 13,2021)

[26] A. Dyck, R. Penners and H. Lichter, "Towards Definitions for Release Engineering and DevOps," 2015 IEEE/ACM 3rd International Workshop on Release Engineering, Florence, 2015, pp. 3-3, doi: 10.1109/RELENG.2015.10.

[27] Claburn, T.: From DevOps to No-Ops: El Reg Chats Serverless Computing with NYT’s CTO. https://www.theregister.com/2017/11/30/from_devops_to_noops_nyt_cto/ (Accessed on Jan. 13,2021)

[28] S. Eismann et al., "Serverless Applications: Why, When, and How?," in IEEE Software, vol. 38, no. 1, pp. 32-39, Jan.-Feb. 2021, doi: 10.1109/MS.2020.3023302.

[29] Mart Laul. "The Biggest Companies Using Serverless Right Now". *Dashbird*. https://dashbird.io/blog/companies-using-serverless-in-production/. (Accessed Jan. 8, 2021).

[30] Mansoor Iqbal, “Netflix Revenue and Usage Statistics”. *Business of Apps.* https://www.businessofapps.com/data/netflix-statistics/. (Accessed on Jan. 9, 2021).

[31] Yury Izrailevsky. “Completing the NetflixCloud Migration”. *Netflix*. https://about.netflix.com/en/news/completing-the-netflix-cloud-migration. (Accessed Jan. 9, 2021).

[32] Mart Laul. “Serverless Case Study - Netflix”. *Dashbird.* https://dashbird.io/blog/serverless-case-study-netflix/. (Accessed Jan. 9, 2021).

[33] Mike Chan. “Serverless Architectures: Everything You Need to Know”. *ThornTech.* https://www.thorntech.com/2017/03/serverless-architectures-everything-need-know/. (Accessed Jan. 9, 2021).

[34] Taavi Regemagi. “Serverless Case Study - Coca-Cola”. *Dashbird*. https://dashbird.io/blog/serverless-case-study-coca-cola/. (Accessed Jan. 9, 2021).

[35] iRobot. *Company Information*. iRobot. https://www.irobot.com/about-irobot/company-information (Accessed Jan. 9, 2021).

[36] iRobot. “IRobot Case Study”. AWS. https://aws.amazon.com/solutions/case-studies/irobot/. (Accessed Jan. 9, 2021).

[37] Evan Ackerman. “iRobot Announces Major Software Update, Shift From Pure Autonomy to Human-Robot Collaboration”. IEEE Spectrum. https://spectrum.ieee.org/automaton/robotics/home-robots/irobot-home-autonomy-update/. (Accessed Jan. 9, 2021).

[38] Monash University. “FIT3170-Software Engineering Practice”. https://handbook.monash.edu/2020/units/FIT3170 (Accessed Jan. 17, 2021).

[39] Statista Research Department. “Public cloud application infrastructure services market 2015-2022”. https://www.statista.com/statistics/505248/worldwide-platform-as-a-service-revenue/ (Accessed Jan. 17, 2021).

Appendix

Team Contribution

Tasks and Time Tracking For Technical Report 3			
Name	Task done	Time Spent (hours)	Notes
Stuart	Writing about What is Serverless Architecture	1	
	Writing about Difference and Similarities to Alternatives	1.5	
	Writing about Relation to historical trends	1.5	
	proofreading	0.5	
	Total Hours	4.5	
	Total Tasks	4	
Jeng Yue	Readings on serverless architecture	1.5	
	Reading materials written by others	1	
	Writing up conclusion	1	
	Fact checking	1	
	proofreading	0.5	
	Total Hours	5	
	Total Tasks	5	
Nassar	Reading about Serverless Architecture	1	
	Setting Up Document	0.5	
	Case 1: Netflix	1	
	Case 2: Coca Cola	1	
	Case 3: iRobot	1	
	Final Formatting and Fixes	2	
	proofreading	0.5	
	Total Hours	7	
	Total Tasks	7	
Wei Han	Research on cost of serverles	1.5	
	Research on benefit of serverles	1.5	
	Research on devops and serverless	1	
	Research on serverless suitability	2	
	Research on serverless risk	2	
	write up	4	
	proofreading	0.5	
	Total Hours	12.5	
	Total Tasks	7	
Chee Feng	Research about Serverless architecture	1.5	
	Read the report	1	
	Write up the introduction	0.5	
	Write up the executive summary	0.5	
	Write up the research limitation	0.5	
	proofreading	0.5	
	Total Hours	4.5	
	Total Tasks	6	