Andrew Song - 1204822

CIS4010

Assignment 4 – Report

03/26/23

The argument that Lambda is simply a serverless computing platform, and therefore could can be seen as a duplicate form of what is already provided by services like Fargate, has *some* merit. This argument is derived from the fact that both Lambda and Fargate are marketed as serverless platforms sharing much of the same features (or at least similar), such as their pricing model, dynamic scaling (Lambda being based on concurrency limits), and containerized deployments. However, although both services provide a serverless computing platform, they have different strengths and weaknesses based on the specific requirements and use-cases of the application in question. In this report, I will try to tackle the big factors that play into choosing either Lambda or Fargate + ECS/EKS and the reasons for selecting one over the other. Additionally, I will delineate the key facets of each of the services that will help me show that the proposed claim is a vast oversimplification.

AWS Lambda is a serverless computing platform that allows users to run code without the need for creating and maintaining/managing servers. It is built to handle individual requests or tasks in response to specific, predefined events/triggers – such as API Gateway requests or object uploads to an S3 bucket, both of which were employed in parts one and two of this assignment, respectively. Lambda is a good choice for simple, lightweight jobs that require quick fast execution and response times (up to a maximum of 15 minutes) and can benefit from Lambda's dynamic concurrency limit scaling. Lambda functions can be deployed as docker containers, though there are a fair amount of restrictions with regard to configurations that limit you to the Lambda infrastructure.

On the other hand, Fargate + ECS is designed specifically for running containers in a serverless environment. You have a lot of flexibility when it comes to configuring your containers the way you want them. It provides more control and customization options for deploying and managing containerized applications, making it a solid option for applications that require more complex configurations, or even just those that need to run longer-running tasks (can run 24/7).

Lambda's simplicity and ease of use make it an ideal choice for developers who want to focus on writing code without worrying about server management. Lambda functions are triggered by events, and AWS takes care of scaling and managing the infrastructure needed to execute them. It is a good choice for use cases such as running a small API or processing incoming data. The user simply needs to take care of the application code (the functions), letting Lambda handle all the server, hardware, virtualization stuff for you. However, Lambda's simplicity also means that it lacks some of the advanced customization options provided by Fargate. For example, it does not allow users to customize the underlying infrastructure or the operating system.

Fargate, on the other hand, provides far more control and customization options, making it a good choice for more complex applications. ECS (docker-based containers) and EKS (Kubernetes-based containers) in conjunction with Fargate allows users to customize the underlying infrastructure, such as the container instance type and size, and the operating system. They also allows for longer-running tasks, making it suitable for workloads that would otherwise struggle with Lambda’s 15 minute timeout limit.

Another factor to consider when choosing between Lambda and ECS with Fargate is cost. Lambda's pay-as-you-go model can be more cost-effective for applications with low usage, but can become more expensive as usage scales up. ECS with Fargate provides more cost control options, despite also using a pay-as-you-go model, such as setting task or service-level budgets; however, this tends to require more upfront investment in infrastructure and setup.

Finally, it is worth noting that Lambda can be part of a microservices architecture. Many microservices can be broken down into smaller, individual tasks that can be executed by Lambda functions. In this way, these services are not coupled with the application layer, maintaining a modular design and workflow. These services can be scaled massively and independent of the application, reducing bloat and potential bottlenecks. Lambda functions can be connected using event-driven architecture and AWS services like API Gateway, SNS, or SQS. However, Lambda's maximum runtime limit of 15 minutes can be a limitation for some microservices use cases that require longer processing times, though this often is not the case.

In conclusion, choosing between Lambda and ECS with Fargate depends on the specific requirements of an application. Lambda is a good choice for simple and lightweight tasks that require quick response times and benefit from Lambda's automatic scaling and pay-per-use pricing model. ECS with Fargate is a better choice for more complex applications that require more control and customization options, such as customizing the underlying infrastructure or running longer-running tasks. And although both services are serverless computing platforms, Lambda definitely fits the bill as being able to facilitate microservice architectural patterns. It allows the services to scale independent of the main application / server layer, allowing for a less-coupled and modular workflow/design conducive to the microservice architecture.