Assessment Report

AWS Launchpad Program

From: Emilio Baguide

To: Daniel Wilkinson

Project Goals

Design and implement a solution that is highly available, resilient and secured microservice architecture spanning multiple accounts using infrastructure as a code.

## Business Case

### Backgrounds

The company A has a monolithic application which allows users to store photos and the company wants to create a multi-accounts environment where each account will belong to an individual business unit and will manage different parts of the application.

### Requirements

The solution must be:

* Highly available
* Resilient
* Secure
* Scalable
* Recovery
  + Recovery in timely and orchestrated manner
  + Simple recovery architecture
  + Cloud native
  + Highly available
  + Resilient
* Perform multiple release (during the day) without down time

## Solution Strategy

### Reliability

Multiple availability zone was used to build the platform infrastructure for the application. For instance, the Auto Scaling Groups spans to multiple availability zones as well as the RDS database providing high availability in case of zone failure and fault tolerant implementation.

We have used EC2 Auto-Scaling for the web application servers which allows to avoid the application to and quick recovery if that occurs. The instances, when launch, use Launch templates that standardise the configurations for the application servers.

The application servers are fronted by an Application Load Balancer that check the health of the instances and balance the traffic among the healthy instances. Whenever, the instances get unhealthy, load balancer instructs the Auto Scaling services to launch new instances.

We have used the Elastic File System to centralize the application packages to ensure that the new instances that are launched can mount the file system and get the required packages to run the application. The improves the application resiliency for the file system.

For the object’s storage, we have used S3 where the Lambda packages and the uploaded photos are kept. Additional S£ storage are also created to store the processed images.

To ensure database resiliency, I have used multi-AZ implementation of RDS that keeps the primary database instance in one availability zone and a standby instance in different availability zone. However, this solution needs to be reviewed as DynamoDB could also be used as it stores tables in multiple availability zones and is highly available. The use of RDS database in this project is related to the application code in use and limited time to make to required change to use DynamoDB

### Performance

The ALB and Auto-Scaling help to monitor the server’s performance and automatically adjust the resources as needed. Additionally, the Microservices services helps to decouple the workload and have different services managing the processes separately. For this, I have used Lambda, S3, SNS, Queue and API Gateway. However, this implementation can be improved by triggering Lambda straight from the S3.

The CloudFront that sits in front of the Load-Balancer, allows to manage the client requests and improve performance by caching files at the edge. This minimizes the number of requests that the application servers need to manage.

The database implementation also helps to keep a good performance as the standby RDS instance can quickly take over the role of primary database instance if a failure occurs.

The Transit Gateway is used to route traffic between the VPCs and is meant to centralize the applications traffic. VPC Peering is another option, but I opted for the Transit Gateway to reduce management with the VPC peering and keep the infrastructure ready to the business increase and the subsequent expansion.

The overall application performance also benefits of the use of Infrastructure as Code (CloudFormation) to deploy the infrastructure as well as helps to make changes in the infrastructure with minimal or no downtime. The majority of configurations are done via code except some specific configurations such as AWS Organization setup.

### Infrastructure Security

The identity and access management are centralised via AWS Organizations that helps managing the accounts setups and apply security policies. The access to resources is managed by AWS managed security policies and Customer-managed policies whenever required. We also use roles to temporarily provide access to principal and services to allow them to perform tasks and when it is required only. This helps with the implementation of least-privileges access.

The traffic flows are set to accept users request via the CloudFront using HTTPs and then the internal traffic flows within the AWS networks. Users are not supposed to have access to the internal application infrastructure.

To control or monitor changes in the infrastructure, we have setup the Cloud Trail to log the activities in the accounts. CloudWatch was also configured and is mainly watching into the RDS resources but can be extended to other services as needed.

### Cost Optimization

We have setup AWS Budget alarm to help monitor the expending o the infrastructure resources. As we have multiple accounts, their costs can be consolidated into a simple point of management using AWS Organisations. With AWS Organisations, it is easier to also control the security of the environment by enforcing policies rule to govern the security bests practices such as multifactor authentications, strong passwords and more.

Graphical user interface, text, application, email

Description automatically generated

### Operations

As mentioned before, we have used CloudFormation to build the infrastructures. With CloudFormation we have defined the infrastructure as code, and it can be deployed multiple times and in different regions. This works very well not only for day-to-day operations but also for recovery due to catastrophic events.

Another tool that was used was GitHub which is code repository where we keep the code for this project.

## Issues and Challenges

### Devops Tasks

To execute this project, some Devops and code skills were required. This was quite challenging for me that was not familiar with the tools and had limited skills to code the infrastructure. For example, most of the time spent in this project was to troubleshoot errors. This constrained my performance in this project as I was not able to provide a fully working infrastructure. However, this was beneficial in some way as it gave me the ability to code infrastructure and gain experience troubleshooting issues.

The ability to understand how some AWS services work, was also a challenge as during the Launchpad program, the time to focus on specific subject was limited due to the tight calendar. For example, it was not a straightforward process to understand how to link API Gateway, Lambda and SQS in the context of the project requirement by using CloudFormation.

### Project Outcomes

Unfortunately, I was unable to provide a workable infrastructure, but the challenges gave me additional capabilities to build and manage AWS services and feature.

The requirement of decoupling the application into multiple accounts was one of the main challenges I found in this project as I could not manage the cross-account reference. This limited my ability to conclude the project that I would be able to do more if it was a single account deployment. This was an opportunity to familiarise with the type of tasks that I need to work in projects.

The Auto Scaling/Load Balancer/EC2 setup did work at all due to an unknown error that was preventing the web server to run. This started with the instances not being healthy and other experienced colleagues joined but that could not be resolved. Later I discovered that the httpd server was not coming up after the instances were launched. This seem to be a broken code for the provide application. I was unable to contact who contact who provided the code to confirm the root of the issue/seek for help as it was weekend.

By taking the decision of using transit gateway that could be shared via Resource Access Manager seemed to be one solution connect the accounts but at the same time it prevented me to setup the connection properly as the sharing was not working. Potentially due to my miss-understanding on how to progress with this configuration.

### Lesson Learned

* AWS documentation provides is a good source information that helps a lot. I managed to learn CloudFormation quickly from there. I just need to keep working on it to master my capabilities
* Client request varies and the best solution depends on many factors such as cost, security, business needs, etc.
* Attention to details is essential while interpreting the client requirements.

### What is Next

1. Discussion with Daniel Wilkinson
2. Attempt to conclude this project on my free time
3. Work with Danny and the team to accelerate my abilities/capacity to join into projects if possible.
4. Conclude Adrian Cantrill AWS Architect Course