Cloud Development Continual Assessment (2024)

Creating a Web App Game with Amazon Web Services

Paul Kerins – T00188667

# Abstract

The objective of this project is to develop a prototype of a web application game with functionality of connecting to the Amazon Web Services (AWS) and allow for the game to run as expected while using the cloud services. The game will allow the user to log in or sign up to the web page by email and once signed up the user is brought to the main web page where the game is hosted. The game will select a random word from DynamoDB, and the user will have five guesses to guess the word. The game will inform the user if they have correct letters in their guess and tell them if they are in the correct position or the incorrect position. All the project’s technical requirements are by means of Amazon Web Services. The focus was the use of Amplify, DynamoDB, API Gateway and Lambda as they provided the architecture, storage, and functions to allow for this project to be functional.

The results of this project were a functional web application game running off the selected cloud services.

The react project code can be found at the GitHub link - *https://github.com/PKerins97/CloudDevApp*

# Introduction

The reason for choosing this topic to research is due to the increased number of web application games that have been produced such as Global, Total Battle, Happy Snakes and many more. This project was to investigate the use of cloud services to create a web app game and host it using those services. In order to be able to complete this project research and learning had to be undertaken in order to understand the services that were needed in order to make the project a success. By completing the AWS labs on the chosen services gave a good understanding of how they work and how to connect them. As some aspects were difficult to understand from the labs, there were videos that helped to make a start template (Tutorials, 2023)

# 2. Literature Review

## 2.1 Amazon Web Services.

Amazon Web Services is a comprehensive, evolving cloud computing platform offered by Amazon.com. It provides a wide range of services including computing power, storage options, networking, databases, machine learning, analytics, security, and more. These services are provided on a pay-as-you-go basis, allowing businesses to scale and grow without large upfront investments in infrastructure. AWS has become one of the leading cloud service providers globally, powering thousands of businesses, from startups to large enterprises, with the infrastructure needed to run their applications and services reliably and securely. AWS was launched in 2006 offering its first service, Amazon S3 which provided developers with access to reliable and scalable store infrastructure. Since then, Amazon Web Services has rapidly expanded its selection of services to cater for various industries looking to use cloud services.

Some of the key services offered by AWS are:

* Computer Services
* Storage Services
* Database Services
* Networking
* Machine Learning and AI Services
* Security and Identity Services

## 2.2 Lambda

AWS Lambda is a serverless computer service provided by Amazon Web Services. “AWS Lambda is a computer service that lets you run code without provisioning or managing servers” (Amazon Web Services, 2024). With Lambda, you can upload your code and the service takes care of everything required to run and scale your code with high availability.

The key features of AWS Lambda include:

* **Serverless Computing:** Lambda abstracts the underlying infrastructure.
* **Event-Driven Architecture:** AWS services can trigger lambda functions in a variety of ways, including custom events using AWS SDKs, updates to DynamoDB tables, incoming API requests through Amazon API Gateway, and modifications to data stored in Amazon S3 buckets.
* **Scalability and High Availability:** Lambda automatically scales your code by running as many instances of it as needed in response to incoming requests. It also provided built-in fault tolerance, ensuring high availability for the application.
* **Supports Multiple Programming Languages:** Lambda supports several programming languages including Node.js, Python, Java, Go, Ruby, and .NET Core. This allows developers to write functions in their preferred language.
* **Integration with AWS Services:** Lambda allows for a seamless integration of other AWS services enabling you to build powerful and scalable applications.

## 2.3 Amplify

Amazon Web Services offers a suite of tools and services called AWS Amplify that make it easier to develop safe and scalable cloud-based applications. Its goal is to speed up the creation of online and mobile applications by offering developers a variety of features and integrations along with a streamlined workflow. “AWS Amplify Hosting is a fully managed CI/CD and hosting service for fast, secure, and reliable static and server-side rendered apps that scale with your business” (Amazon Web Services, 2024).

Here are some of the key features and components of Amplify:

* **Backend Services:** Amplify offers a variety of backend services that can be easily integrated into applications, including authentication, storage, databases, analytics, and APIs. These services are fully managed by AWS, reducing the complexity of backend development.
* **Authentication:** Amplify provides authentication services that enable developers to add user sign-up, sign-in, and access control to their applications quickly. It supports authentication methods such as username/password, social identity providers such as Google or Facebook, and multi-factor authentication (MFA).
* **Data Storage:** Amplify provides storage options including Amazon Aurora Serverless (relational database), Amazon DynamoDB (NoSQL database), and Amazon S3 (file storage). These storage choices are simple for developers to integrate into their applications for user data management, media file management, and application state management.
* **APIs:** Amplify allows developers to create and deploy GraphQL APIs using AWS AppSync. GraphQL APIs provide a flexible and efficient way to interact with backend data, enabling clients to fetch only the data they need in a single request.
* **Hosting:** Amplify provides hosting services for web applications, allowing developers to deploy their front-end code to a global content delivery network (CDN) with a single command. This simplifies the process of deploying and scaling web applications while ensuring low latency and high availability for end users.
* **CI/CD:** Continuous integration and continuous deployment (CI/CD) workflows are supported by Amplify, which enables developers to automate the build, test, and deployment of their applications. Popular CI/CD tools like GitHub Actions and AWS Code Pipeline are integrated with it.
* **Frontend Framework Support:** Amplify supports popular frontend frameworks and libraries such as React, Angular, Vue.js, and React Native. Developers can leverage Amplify's libraries and UI components to integrate AWS services seamlessly into their front-end applications.

## 2.4 API Gateway

Developers can construct, publish, maintain, monitor, and protect APIs at any scale with ease thanks to Amazon Web Services' fully managed solution, AWS API Gateway. Applications can access data, business logic, or functionality from backend services like AWS Lambda functions, Amazon EC2 instances, or other HTTP endpoints through it as a front door. “Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications” (Amazon Web Services, 2024).

API Gateway has many key features such as:

* **API Creation and Management**: Developers can construct, publish, maintain, monitor, and protect APIs at any scale with ease thanks to Amazon Web Services' (AWS) fully managed solution, AWS API Gateway. Applications can access data, business logic, or functionality from backend services like AWS Lambda functions, Amazon EC2 instances, or other HTTP endpoints through it as a front door.
* **Integration with Backend Services:** API Gateway supports integration with a variety of backend services, including AWS Lambda, Amazon EC2, Amazon S3, Amazon DynamoDB, and HTTP endpoints hosted anywhere. This enables developers to build serverless or traditional applications and expose them as APIs without managing infrastructure.
* **Security and Authentication:** Built-in security features in API Gateway allow for API request authorization and authentication. AWS IAM (Identity and Access Management), Amazon Cognito user pools, API keys, and custom authorizers are just a few of the authentication mechanisms that it offers. For safe connection between clients and the API, you can also activate HTTPS.
* **Request/Response Transformation:** API Gateway allows you to transform requests and responses using mapping templates written in Apache Velocity Template Language (VTL). This enables you to modify request and response payloads, headers, and status codes to meet the requirements of your backend services or clients.
* **API Monitoring and Logging:** API Gateway provides built-in monitoring and logging capabilities to help you track API usage, performance metrics, error rates, and more. You can use Amazon CloudWatch to monitor API Gateway metrics and logs, set up alarms, and troubleshoot issues in real time.
* **Scalability and High Availability:** API Gateway is designed to scale automatically to handle any amount of traffic, from a few requests per day to millions of requests per second. It is deployed across multiple AWS Availability Zones for high availability and fault tolerance.

## 2.4 Cognito

AWS Cognito is a managed authentication, authorization, and user management service provided by Amazon Web Services It helps developers add user sign-up, sign-in, and access control to their web and mobile applications quickly and easily. “With Amazon Cognito, you can authenticate and authorize users from the built-in user directory, from your enterprise directory, and from consumer identity providers like Google and Facebook” (Amazon Web Services, 2024). Cognito simplifies the process of implementing user authentication and authorization in applications.

Key features of AWS Cognito include:

* **User Authentication:** Several authentication techniques are supported by Cognito, including federated identity providers (like SAML and OpenID Connect) and social identity providers (like Facebook, Google, Amazon, and Apple). The authentication techniques that developers select will depend on the needs of their application.
* **User Pools:** Developers can establish and maintain user accounts for their apps with the help of Cognito User Pools, which offer a user directory. User Pools maintain user accounts, including updating attributes and changing passwords, as well as handling user registration, authentication, and account recovery.
* **Integration with AWS Services:** Building safe, scalable apps with end-to-end authentication and authorization capabilities is made possible by Cognito's smooth integration with other AWS services like API Gateway, AWS Lambda, and Amazon S3.
* **Scalability and Performance:** With minimal latency and maximum availability, Cognito is built to grow automatically to accommodate millions of users and authentication requests. It makes use of AWS infrastructure to guarantee performance and dependability.
* **Analytics and Monitoring:** With the built-in analytics and monitoring tools in Cognito, developers can keep an eye on user sign-in behaviour, analyse authentication metrics, and quickly spot security issues. To track and examine user authentication events, developers can make use of Amazon CloudWatch and Amazon Cognito Events.

## 2.5 DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service, offers smooth scaling along with quick and reliable performance. With DynamoDB, you can delegate the administrative responsibilities of managing and expanding a distributed database, eliminating the need for hardware provisioning, setup and configuration, software patching, replication, and cluster scaling. Additionally, encryption at rest is provided by DynamoDB, which removes the complexity and operational load of protecting sensitive data.

Key features that DynamoDB offers include:

* **Fully Managed Service:** DynamoDB offers a fully managed service, as it handles the provisioning, scaling, and maintenance of the database.
* **NoSQL Database:** DynamoDB is a NoSQL database, which provides a flexible schema design and scalability. It allows for developers to store and retrieve data in flexible, schema-less JSON-like documents, which makes it more suitable for applications with evolving data models and complex data structures.
* **Performance and Scalability:** DynamoDB offers fast and predictable performance with very low latency for read and write operations. It automatically scales to handle growing workloads by adding and removing capacity based on demand.
* **Flexible Data Model:** DynamoDB supports key-value and document data models, allowing developers to store and retrieve data using simple primary keys. It also supports secondary indexes, allowing for efficient queries and indexing data based on non-primary key attributes.
* **Security and Access Control:** DynamoDB offers fine grained access control through Identity and Access Control Management (IAM), which allows developers to define permissions at the table and item level.
* **Integration with AWS Services:** DynamoDB integrates seamlessly with other AWS services such as Lambda, API Gateway, and S3 enabling developers to build serverless architectures and real-time data processing pipelines.

## Methodology

There were several Amazon Web Services used within this project. The services used were Amplify, Lambda, API Gateway, DynamoDB, IAM and Cognito. Each service will be run through in this section and what was done with them.

A diagram of a software application

Description automatically generated

*Fig 1: AWS Architecture*

The start of this project included setting up an Amazon Web Services account which allowed access to the AWS services needed to carry out this project. Once the account was set up, the next thing that was done was setting up a react project in Visual Studios and then creating an environment in AWS Amplify. The app was developed using the coding languages HTML to design the web page, CSS to style and design the look of the web page and JavaScript which gave functionality to parts of the web page and calls the lambda function. The enviroment was done by navigating to the Amplify console and creating a new app and selecting a host web app. This project was from code in a GitHub Repo so that was the option chosen for the files. The repository that the code being used was chosen and the code on display was edited by removing *‘- npm ci – cache .npm –prefer-offline’* and adding *‘- nmp run build’*.

The next step in the project was to create a lambda function. After navigating to the AWS Lambda console from the main console, clicking on the create function option, here you will select the *Author from scratch* option and enter in the name you wish to call the function and the desired coding language. For the project Python coding language was chosen. Once that is done select create and the next page is where the lambda function is created. In the Code Source section is where the lambda function for the project was added. The code below connects to the database in DynamoDB and retrieves a random word that is stored by getting the ID and returning the ID and word connected to that ID.

A screenshot of a computer program

Description automatically generated

*Fig 2: Lambda Function*

The next service that was used was AWS API Gateway, by navigating to the API Gateway console and creating a new API. For this project the API type that was chosen was a REST API. Giving the API a relative name and creating the API. Then by selecting the ‘/’ and create method to create a GET and POST method connecting to the Lambda function that was created. Once the methods have been liked to the Lambda function, it is necessary to click on the Enable CORS button and selecting the method types that was created. Enable CORS is needed as it allows web applications hosted on a different domain to access resources.

The next service used was DynamoDB. This was used to store the words that will be used for the web app game that will be hosted on the server. Once navigated to the DynamoDB console, the create table option was clicked, a relative name was entered, and a partition key was entered. For this project the key ID was used and then the table was created.

A screenshot of a computer

Description automatically generated

*Fig 3: DynamoDB Table*

Once the table had been created items were entered into the database in order for there to be an object to be retrieved by the lambda function. Once this was done in order to allow the lambda to access the items in the database, there must be a policy set up. By navigating back to the Lambda console and clicking on the configuration title and selecting ‘permissions’ on the side menu and then clicking the role name. Then having to create a policy by clicking on ‘Add permissions’ and then selecting create inline policy and editing the JSON file to allowing for certain tasks to be performed in regard to DynamoDB and adding the ARN code gotten from DynamoDB into the JSON file

A screenshot of a computer program

Description automatically generated

*Fig 4: Lambda Policy code*

The last service that was used was AWS Cognito. This was used in order to add authentication to the web app. When in the Cognito console, the option chosen was to create a user pool. When starting to create the table the option chosen for user pool sign in was E-mail alone. For the password, the cognito defaults was set and No MFA was ticked. The default settings for User account recovery were kept. The next page was kept the same with no changes. For the Email page, send email with Cognito was selected. A relative name was given to the user pool, Hosted UI was selected, the client’s name was ‘webappclient’. In order to connect this to the Amplify project, there had to be further setting up in the Amplify console. Once in the App and navigating to the backend environments and launching the studio. Once in the studio the option for setting up login and sign up. Selecting reuse existing Amazon Cognito resources and selecting all the services created previously to import. Once that is deployed there are lines of config codes given which were entered in the command line once in the application folder in order to configure the app with the needed files. After that code was ran the following code was executed ‘*npm install aws-amplify @aws-amplify/ui-reac’.* Once that was run the authentication was complete and set up for the app.

## 4. Results

The result of this project proves that a web application game can be hosted using Amazon Web Services. The results show that it is a very efficient and effective way of host a web app game. From using AWS, it was possible to host the game of a web app by connecting all the relative services together. By using a react project in my repository and connecting that to Amplify it was able to host the game on the web app. While the project was connected to the cloud it was able to access the data in the DynamoDB table and use the items in the game. The game was able to refresh the item each time the web page was accessed allowing for the word to be different for each user. It was able to access the data due to the API Gateway connecting to the lambda function that was given a policy to access the data stored in the cloud. The app was made to have a security measure in place which was Cognito which was connected to the user pool created to save the users detail information that was entered.

## 5. Discussion

As the first initial intention from this project was to try and use GameLift that changed as the integration was too complex and hard to follow from the documentation. This changed to approach taken in completing this project as the methods changed from creating a multiplayer game to creating a live web app-based game.

A problem that raised when using AWS was figuring out how to integrate Cognito into the app. Many methods where researched such as using authorizers and trying to manually insert the Cognito functions. This problem was resolved by using Amplify backend environment to connect the project with the functions needed to integrate Cognito. Amplify backend produced functions to perform on the command line such as CURL functions and configuration functions.

Throughout this project there was break throughs and halts. One of the biggest halts to this project was trying to integrate GameLift to the project to allow for a more complex multiplayer game to be developed. Configuring GameLift into Unity was the major problem as the documentation was extremely difficult to follow and available videos online were outdated as GameLift is new and constantly being upgraded. If GameLift was integrated to this project, it would have granted many advantage to the project. As GameLift manages the infrastructure needed to host multiplayer games and handles player matchmaking, session placement and session management. It provides ready to use game servers for popular game engines such as Unity and Unreal Engine, which would have allowed for a more complex game could have been uploaded to the server. It also handles auto scaling of the server capacity based on the players demands which helps with the fluctuation in player traffic.

## 6. Conclusion

The conclusion of this project is that by using the relative services a web application game was successfully hosted using AWS cloud. Enhancements that could be made to the project in the future would be creating a more detailed feedback in the game such as displaying all the letters used and colour marking them if they were not in the word, correct but in wrong place or correct and in the right place. A For further research would be to investigate the use of cloud services while host real-time multiplayer games on a web application. This feature could be achieved by using the web service GameLift which is a server that allows for game server hosting and scaling for multiplayer games.

A screenshot of a green square with white text

Description automatically generated*Fig 5. Web App UI*

# References

Amazon Web Services, 2024. *Amazon API Gateway.* [Online]   
Available at: https://aws.amazon.com/api-gateway/#:~:text=Amazon%20API%20Gateway%20is%20a,functionality%20from%20your%20backend%20services.  
[Accessed 6 April 2024].

Amazon Web Services, 2024. *AWS Amplify Hosting.* [Online]   
Available at: https://aws.amazon.com/amplify/hosting/#:~:text=AWS%20Amplify%20Hosting%20is%20a,Deploy%20web%20content%2C%20fast  
[Accessed 6 April 2024].

Amazon Web Services, 2024. *What is Amazon Cognito.* [Online]   
Available at: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html  
[Accessed 8 April 2024].

Amazon Web Services, 2024. *What is AWS Lambda.* [Online]   
Available at: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html  
[Accessed 2 April 2024].

Tutorials, T. T., 2023. *Youtube.* [Online]   
Available at: https://www.youtube.com/watch?v=7m\_q1ldzw0U&list=RDCMUCVpWDEFirsEfz2WGJMVd4mA&start\_radio=1&rv=7m\_q1ldzw0U&t=1054  
[Accessed 10 April 2024].