CSCI 5409 Term Assignment Report

Name: Faizal Maulvi
Banner: B00936426

Table Of Contents

Sr. no.	Topic Name	Page No.
1	Project Introduction	1
2	Services requirements	1
3	Deployment Model	2
4	Delivery Model	2
5	Final Architecture	3
6	Data Security	4
7	Estimate Cost of replicating architecture	5
8	Monitoring Mechanism	6
9	Evolution of Application	6
10	References	7

GitLab Link: https://git.cs.dal.ca/courses/2023-summer/csci4145-5409/maulvi/-/tree/main

- **First, introduce your project. What is it? What is it supposed to do? Give the markers the information they need to evaluate whether your choices in later questions were good choices or bad choices. In other words, give them the context of your software and what it is supposed to achieve, who its users are, and what its performance targets should be.**

  The project that I have implemented is a real-time multiplayer movie-related game. This is a game where one person will decide on a movie and reveal the initials of the movie, the actor's name, and the actress's name participating in that movie. People can create a lobby and invite friends to join that lobby. Randomly, the person will be selected among the players for the selection of the movie. Once, the initials are revealed, the rest of the players will try to guess it. People can also chat amongst themselves and translate it into multiple languages. The leaderboard is also updated in real-time as the game progresses. So, the users of this game can be anyone who has a device with a good internet connection and its performance depends on how quickly the updates reflect all of the players in real-time.

- **Describe how you met your menu item requirements: you will list the services you selected, and provide a comparison of alternative services, explaining why you chose the services in your system over the alternatives.**

  List of the services used by me

  1. AWS Elastic Beanstalk **(Compute)** - This is used by me to host the front-end application. There were alternative services such as using EC2 instance, but I like Beanstalk more as it eases the process of setup and handles the increased traffic by itself.

  2. AWS Lambda **(Compute)** - This is used to create the backend logic in the lambda functions. As each of the events performed in the game, should create one specific trigger, the lambda was best because it was easy to build and deploy. Also, I have integrated lambda with an API gateway built with WebSockets.Hence, to enhance the performance of the game, it made much more sense to use lambda functions in comparison to other options such as EC2 or Beanstalk.

  3. AWS S3 **(Storage)** - In order to create a layer, I have uploaded the zip file of dependencies to the S3 bucket. Also, I have uploaded the zip file with the frontend react code that will be used by Elastic Beanstalk. I don’t really see the alternative to this by any other storage services by AWS.

  4. AWS DynamoDB **(Storage)** - I have used dynamo db to store the details of the lobby and the users. I could have used any other relational databases too, but I thought it would be simpler to use Dynamodb in case I need to add on some non-relational (array, object) fields too.

  5. AWS API Gateway **(Network)** - I have used API Gateway because it gave me the inbuilt functionality to create routes and work with web sockets and lambda directly. And no such alternative is as easy and fast to use when compared to this infrastructure.

6. AWS Secrets Manager **(General)** - I have used the secrets manager that is used during cloud formation to store the API key generated for the API Gateway. It is then accessed in the front to connect to the web sockets and the backend lambda functions to access it to send responses to the connected clients.

7. AWS SNS **(General)** - I have used the SNS feature to send the email of the results to the users. Of course, using SES is the better option to send email but as the lab role doesn't have access to SES, I was kind of forced to use SNS to implement this functionality.

8. Cloud Translation API **(General)** - As we didn't have access to AWS translate, with the permission of Professor Robert, I used this API to translate the messages to other languages. But, currently, I need to translate these messages in the backend lambda function which increases the response time sometimes. So, if I had access to AWS translate, I would have been able to perform this translation in front-end saving some latency.

- **Describe your deployment model. Why did you choose this deployment model?**

  The Model I have used is Multi-Cloud. A Multi-Cloud delivery model is the use of multiple public cloud providers or it might be private too. I have used services from both AWS and GCP. Both are public cloud providers. Most of the infrastructure provisioned and used is from AWS, except the cloud translation API is used from GCP. The reason I choose GCP for this service was because I didn't have access to AWS Translate.

  Considering the low latency requirement, and factors such as cost, public cloud or hybrid cloud is the way to go for sure. Building translation APIs or handling traffic and managing WebSockets in any other deployment model such as a private or community model can consist of more effort, time-consuming, and costs too.

- **Describe your delivery model. Why did you choose this delivery model?**

  The delivery model I think for my project would be a combination of Platform-as-a-Service (PaaS). It cannot be considered under Software-as-a-Service (SaaS) because there is no leasing type mechanism available for this software. So, the game that I have developed can be considered as a platform that provides services to the user to engage with other users by playing this game remotely. The user doesn't have to worry about the underlying infrastructure and uses and interacts with the software directly. Also, building such applications is not possible with the use of FaaS or IaaS. Hence, I decided to go with Platform-as-a-Service.

- **Describe your final architecture:**

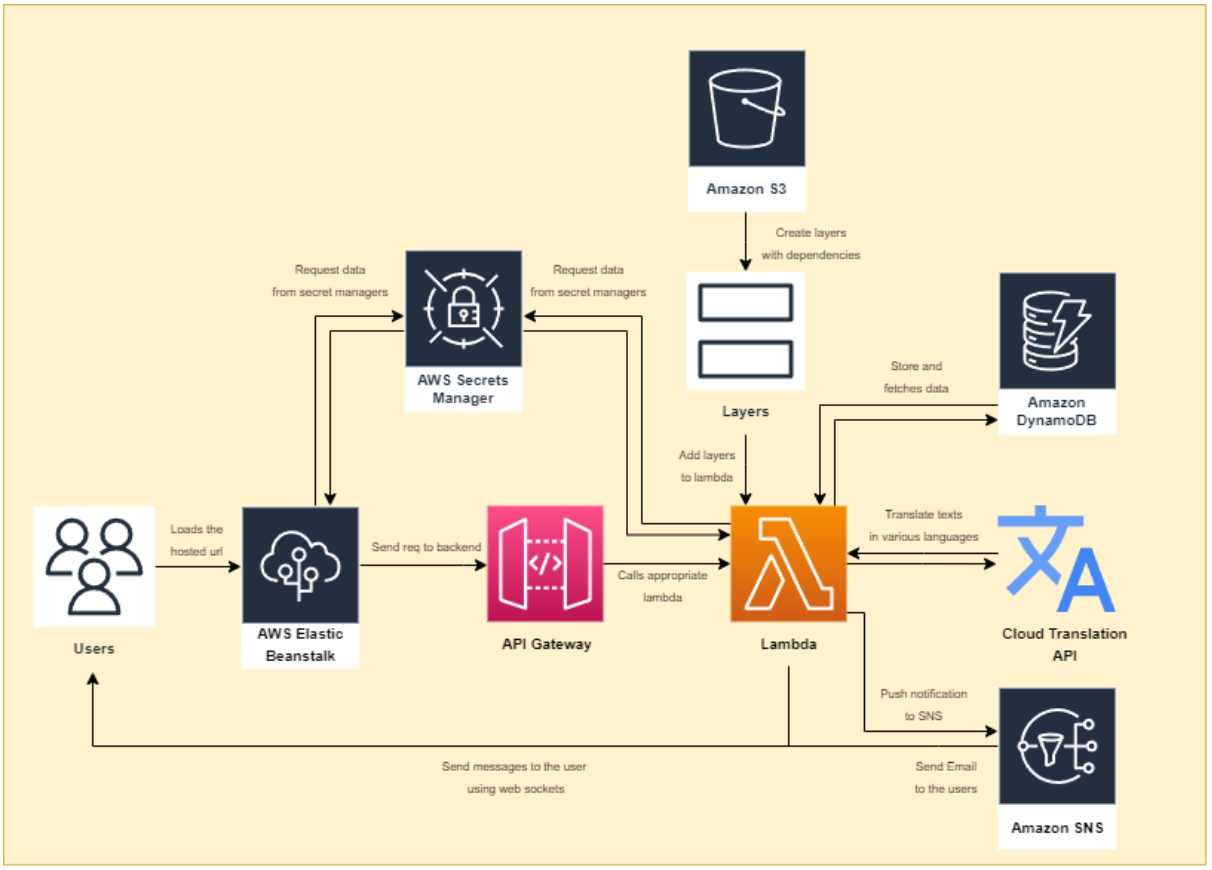

Fig 1: T-Time Cloud Architecture

- **How do all of the cloud mechanisms fit together to deliver your application?**

  As shown in the above introduction and also the above image, my application/game heavily relies upon the use of web sockets in order to implement a fully duplex connection between the server and the client. Hence, as soon as the user opens the home page using the deployed URL of the elastic beanstalk, a new WebSocket connection gets opened. For the routing, an API gateway is used and data of users and games is stored in dynamoDB. The layers are used for the dependencies in lambda functions. For translation of the messages, Cloud Translation API is used whereas to send email Amazon SNS is used. As the URL of API Gateway is generated on cloud formation, it is stored in AWS Secrets manager and then on is used in front-end and back-end to send messages using the web sockets.

- **Where is data stored?**

  The data for the creation of Layers or the zip file for AWS Elastic Beanstalk is stored in the S3 Bucket. Whereas the storage of data for users or the games is stored in the DynamoDB.

- **What programming languages did you use (and why) and what parts of your application required code?**

  I have used ReactJS in Frontend as I wanted to learn React and it also has benefits such as quick development and easy to learn. For the Backend, I have used NodeJS in the Lambda functions as I am familiar with node js and I thought it would be easier to implement it in NodeJS because of the packages available.

- **How is your system deployed to the cloud?**

  The system was deployed and provisioned to the cloud as per the given requirements. The yaml file for the cloud formation was provided to the AWS platform as a stack and it created the resources and returned the final deployed elastic beanstalk URL of the system. I haven't manually provisioned any of the resources.

- **If your system does not match an architecture taught in the course, you will describe why that is, and whether your choices are wise or potentially flawed.**

  Based on my understanding, I think that the architectures implemented are Cloud Bursting Architecture, Rapid Provisioning Architecture, and Workload Distribution Architecture. Elastic Beanstalk can be considered a rapid provisioning architecture because all we need to do is to upload the source code and it takes care of creating an instance and maintaining it. Also, it scales in and out based on the need and hence is also a cloud-bursting architecture. The API Gateway acts as a firewall and passes the request to the appropriate lambda function as per the conditions and is probably a workload distribution architecture.

- **How does your application architecture keep data secure at all layers? If it does not, if there are vulnerabilities, please explain where your data is vulnerable and how you could address these vulnerabilities with further work.**

  The entry point for the application's backend is the API Gateway. It allows if the request is suitable or not and will pass on the data to the lambda functions. For the Lambda functions, all communication is encrypted with Transport Layer Security (TLS) and it always encrypts files including deployment packages and layer archives [1]. DynamoDB by default encrypts data at rest providing enhanced security [2]. The S3 bucket created to upload the cloud formation yml or the zip file for layers doesn't have encryption and block public access is off and hence it can be accessed publicly. Also, the elastic beanstalk is not configured to use https and hence it is more vulnerable to attacks. So these are the two vulnerabilities that can be addressed in the future.

- **Which security mechanisms are used to achieve the data security described in the previous question? List them, and explain any choices you made for each mechanism (technology you used, algorithm, cloud provider service, etc.)**

  All the services used by me are provided either by AWS or GCP and there are specific security protocols inherited by default by them for both, data at rest and in transit. The data in DynamoDB is by default encrypted by the AWS giving us the security for data at rest. Also, the code for the lambda functions is encrypted giving us better security. If a WebSocket remains idle, without sending or receiving a message for 10 minutes, it automatically gets disconnected from the server and hence resulting in managing only the active connections. Also, google cloud has its own encryption policies to protect the data sent for the translation API.

- **What would your organization have to purchase to reproduce your architecture in a private cloud while providing relatively the same level of availability as your cloud implementation? Try to give a rough estimate of what it would cost, don't worry if you are far off. These systems are complicated and you don't know all the exact equipment and software you would need to purchase. Just explore and try your best to figure out the combination of software and hardware you would need to buy to reproduce your app on-premise.**

  To reproduce the architecture in a private cloud, with nearly the same level of availability, the resources need to be provisioned are approximate as given in the below table.

Sr. No	Name	Type	Cost
1	Servers	Hardware	$3000 (once) $200-300 (monthly)
2	SSD (~35TB)	Hardware	$1000 (once)
3	Load balancer	Network	$1995 (once)
4	Translation and email services	Software	$50 (monthly)
5	Hosting and maintenance	Software	$50 (monthly)
6	Software developers	Employees	$5500-6000 per person (monthly)

  Hence, the one-time cost for this provisioned architecture as per my understanding would be around $6000. Whereas there would be a cost of $6000 approximately considering the software is already ready and there is just a single person working on the maintenance of the project.

- **Which cloud mechanism would be most important for you to add monitoring to in order to make sure costs do not escalate out of budget unexpectedly? In other words, which of your cloud mechanisms has the most potential to cost the most money?**

  Google's Cloud Translate API and the AWS DyanmoDB are the most important from the aspect of the budget. If the use of translate API exceeds the threshold, it can cost more than USD 45$ per hour. Also, the usage of DynamoDB if increased highly, can cost a lot more than the expected amount. Hence, we should keep them under watch by adding a monitoring level to these services.

- **How would your application evolve if you were to continue development? What features might you add next and which cloud mechanisms would you use to implement those features?**

  If I continue the development in the future, I would like to have a better appealing UI/Frontend so that the user can like to play the game. I would like to add a timer such as if 5 minutes pass by and no one is able to find out the answer, scores will be given to the person who chose the word. I would also prefer to have the translation module done in the front end instead of the back end, and that will help me in decreasing the latency of some APIs. In addition to these, I will also implement the feature of using SES to send emails instead of SNS. Currently, there is no mechanism for a player to join the game if disconnected. So in the future, this feature can also be implemented. Also, not to forget I need to overall decrease the latency in order to have smooth gameplay. One more thing that can be added from a security point of view is to add the security group and manage VPC, subnet, etc for all of the provisioned resources resulting in higher security.

- **References**

[1] D. Musgrave, “Lambda,” *AWS*, 2022. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/security-dataprotection.html. [Accessed: 01-Aug-2023]

[2] “DynamoDB encryption at rest - amazon dynamodb,” *AWS*. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/EncryptionAtRest.html. [Accessed: 01-Aug-2023]

[3] “How much does website hosting cost?,” *Shopify*, 21-Apr-2022. [Online]. Available: https://www.shopify.com/ca/blog/hosting-costs. [Accessed: 01-Aug-2023]

[4] “ThinkSystem ST550: Enterprise 4U tower servers,” *Lenovo CA*. [Online]. Available: https://www.lenovo.com/ca/en/p/servers-storage/servers/towers/thinksystem-st550/77xx7trst50. [Accessed: 01-Aug-2023]

[5] “WD BlueTM sata internal SSD hard drive 2.5"/7mm cased,” *Western Digital*. [Online]. Available: https://www.westerndigital.com/en-ca/products/internal-drives/wd-blue-sata-2-5-ssd#WDS200T2B0A. [Accessed: 01-Aug-2023]

[6] “Load balancer enterprise ADC pricing, packages, plans 2023,” *G2*. [Online]. Available: https://www.g2.com/products/load-balancer-enterprise-adc/pricing. [Accessed: 01-Aug-2023]

[7] “How to build a chat using Lambda + WebSocket + Api Gateway? (nodejs),” *YouTube*, 07-Jun-2021. [Online]. Available: https://www.youtube.com/watch?v=BcWD-M2PJ-8. [Accessed: 01-Aug-2023]

[8] “Packing node.js functions for lambda (node_modules),” *YouTube*, 21-Feb-2022. [Online]. Available: https://www.youtube.com/watch?v=XydK2g4zQ9E. [Accessed: 01-Aug-2023]

[9] “AWS resource and property types reference - AWS CloudFormation,” *AWS*. [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html. [Accessed: 01-Aug-2023]

[10] “Translating text (BASIC) | cloud translation | google cloud,” *Google*. [Online]. Available: https://cloud.google.com/translate/docs/basic/translating-text. [Accessed: 01-Aug-2023]