Gaming on the Cloud

Authored By:

Purva Nishikant Choudhary
Juan Davalos
Narbeh Movsesian
Mayur Rahangdale

Table of Contents

Introduction and Background

Statement of Problem

The problem depicted in this paper is that of cloud gaming. Though the phrase cloud gaming may be confused with video games having online functionality such as multiplayer or cloud save, the phrase actually means gaming on the cloud without a user needing to worry about hardware limitations to play video games. This paper will look at ways of architecting a cloud gaming system using a number of cloud technologies offered at the time of writing.

Background

Cloud gaming, sometimes called gaming on demand or gaming-as-a-service, is a type of online gaming that runs video games on remote servers and streams them directly to a user's device, in other words, playing a game remotely from the cloud. A completely different approach to the traditional means of gaming, where a game runs locally on a user's video game console, personal computer, or mobile device.

Cloud gaming provides a new concept of online game organization, where the game is delivered from a server located in a cloud. The most common model for cloud gaming is “Remote Rendering Gaming as a Service”, in which the multiplayer server, the game logic, and the rendering are all located on a server, while the only main functionality left to the client is the input module. In this model a high-definition video is sent to the client and the client commands are sent to the server

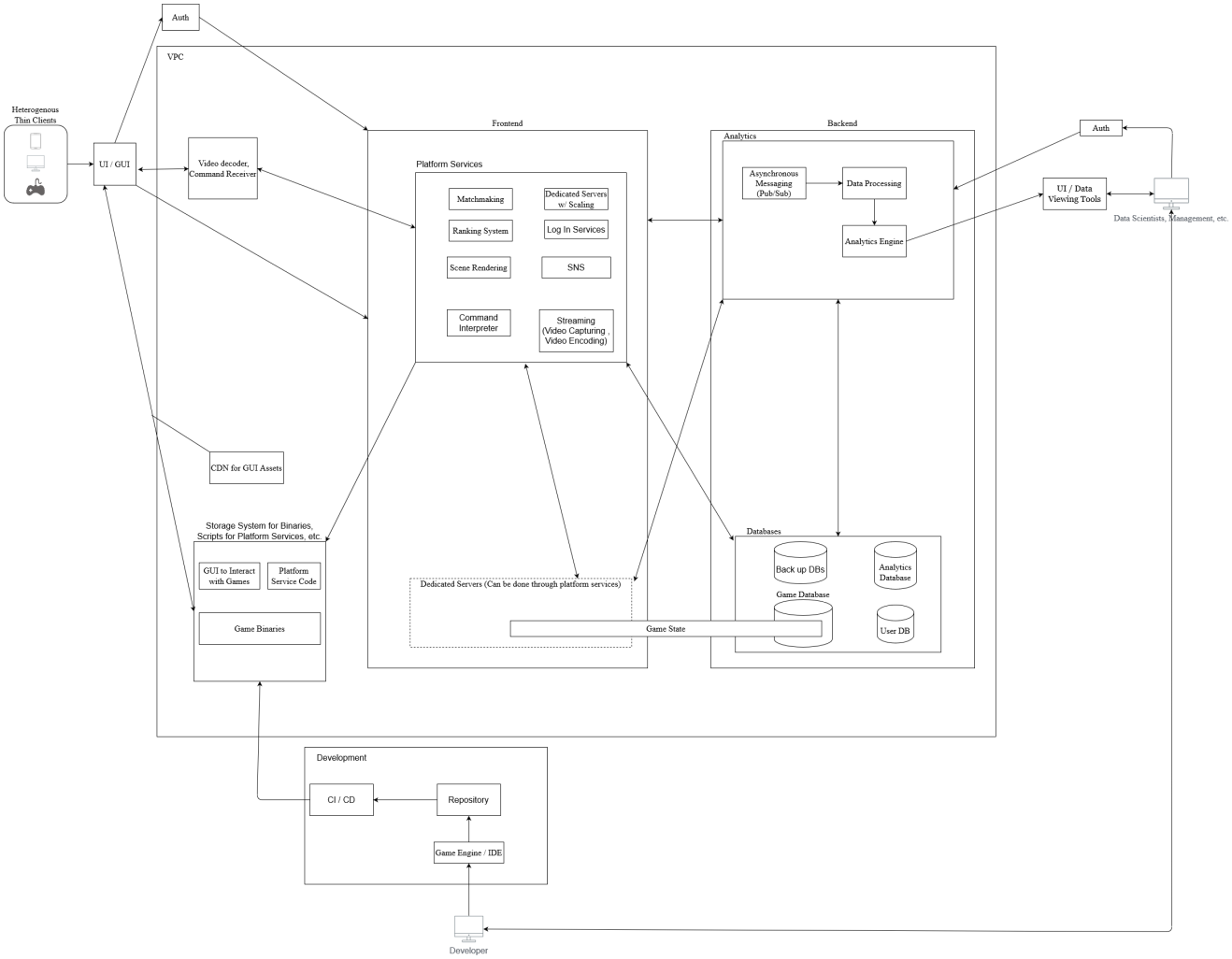
Previous and Current Work

From its conception by Robert Stevenson and Tim Wilson in 2011, cloud gaming has been gathering popularity due to its ease of use by any person trying to play games. Shortly after Stevenson and Wilson presented the ability to play games on the cloud, companies such as Sony, Microsoft, and Nvidia paved the way for cloud gaming to grow. Currently there are a number of cloud gaming services with the top three being Google Stadia, Nvidia GeForce Now, and Microsoft xCloud. Amazon also entered the cloud gaming market in 2020 in their Amazon Luna product. A notable work is that of Google Stadia and Amazon Luna which require the player to purchase a separate control to have direct connection between the controller and the game servers. Much of the previous and current work attempts to attract developers through the use of software development kits (SDK) to develop games over the cloud.

Seminar Description

The seminar has discussed various sections of cloud gaming. The first began by explaining cloud gaming and looking at a potential reference architecture for developing cloud gaming systems. The "Reference Architecture" section will show the various pieces of said reference architecture in order to get a better understanding of some of the major components. The second seminar focused on the implementation of the reference architecture using AWS services and recommending other services such as Google Cloud and even Google Stadia's and Amazon Luna's software development kits (SDK) as a potential alternative to the streaming service required for cloud gaming.

Reference Architecture



Implementation Guides

The implementation guide will consist of seven major components in the reference architecture with an AWS implementation and an alternative. Note that the parts are interchangeable as long as the cloud providers support cross-provider access through policies.

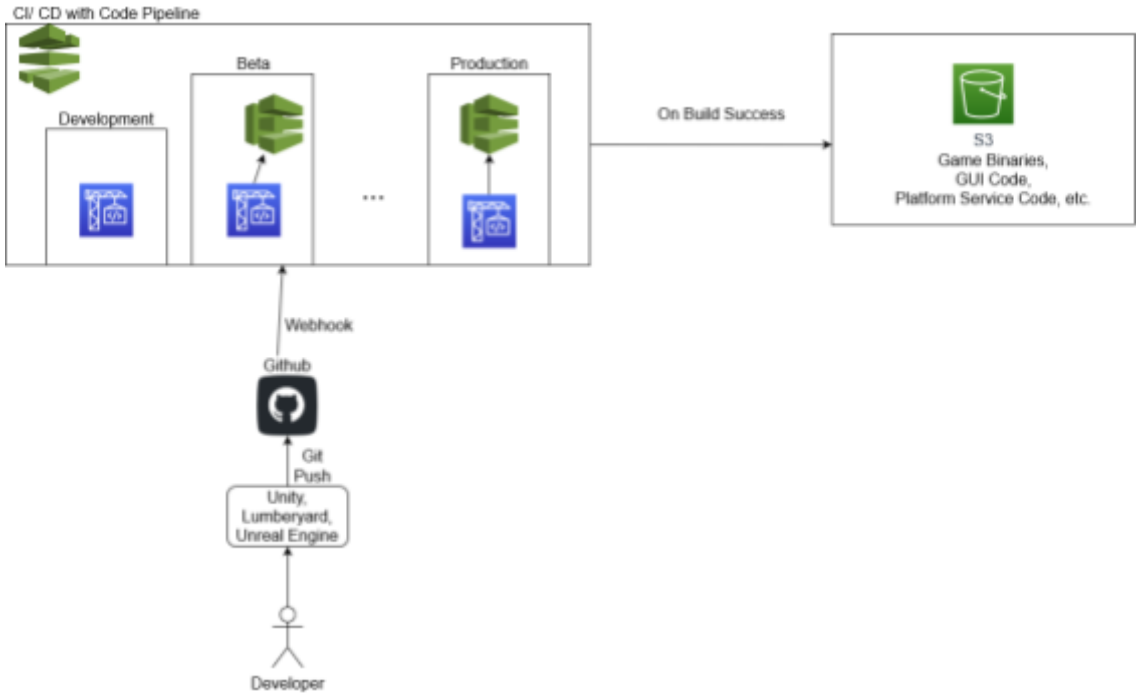
Development

Both development tracks require a development environment in order to begin any application development. For game development any number of game engines may be used. The preferred game engines in these implementations are AWS Lumberyard by Amazon.com Inc., Unity by Unity Technologies, and Unreal Engine. All three provide powerful development environments for games in a variety of languages. Note that Unity and Unreal Engine require licenses after a certain period.

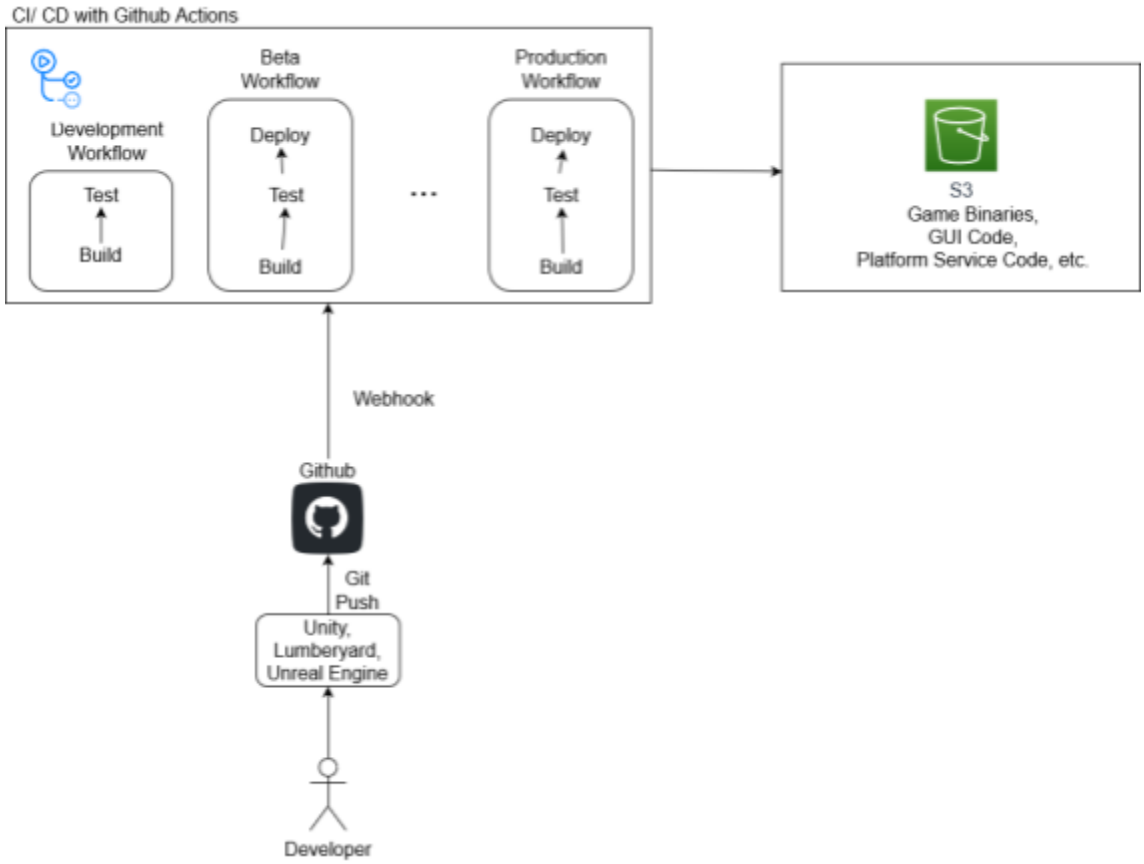
A repository is also needed to store game code, platform code, and UI code. For both implementations any repository can be used so long as it can interface properly with the other technologies. For simplicity sake Github will be mentioned as the primary repository to use along with Git as version control software.

Note that in order to push to a repository proper authentication must be included. The CI/CD also requires the proper authentication in order to use the S3 Buckets or any other AWS products. In order to do this a user pool must be created and S3 must have a policy in which the CI/CD can read and/ or write to the S3 Buckets. Each workflow may require its own policy and identification based on the appropriate Bucket binaries and assets will be placed.

a. AWS



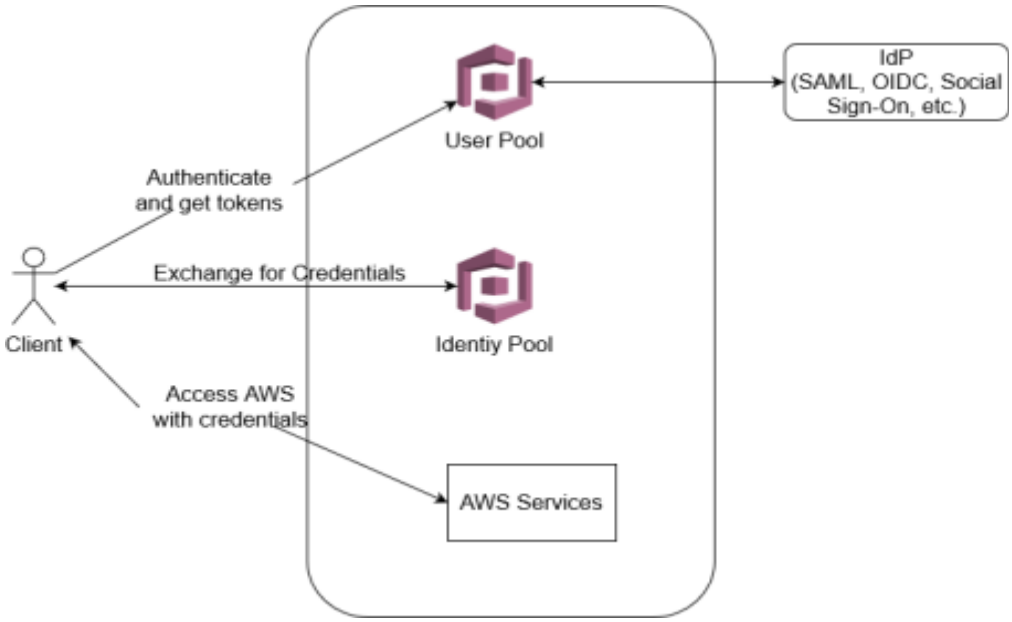
b. Alternative



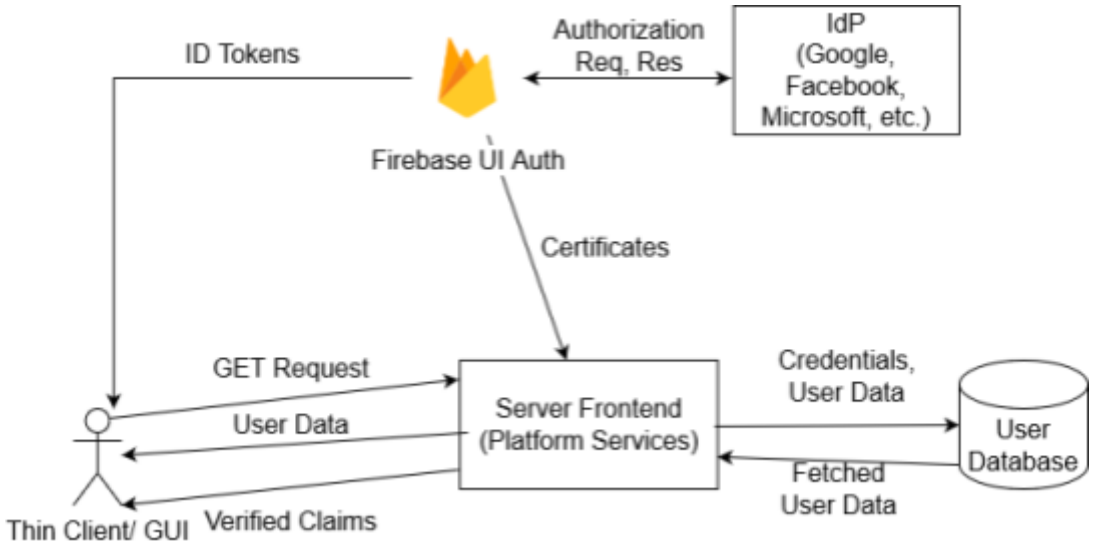
User Authentication

Both tracks defer in their providers but have the same basic idea. With AWS user pools and identity pools must be set in order for users to access any AWS services. These pools can also use any number of authentication services such as SAML, OIDC, and so on as well as authentication provided by services such as Google and Amazon. With FirebaseUI Auth by Google a database must be created for users which FirebaseUI Auth will query for user authentication.

a. AWS



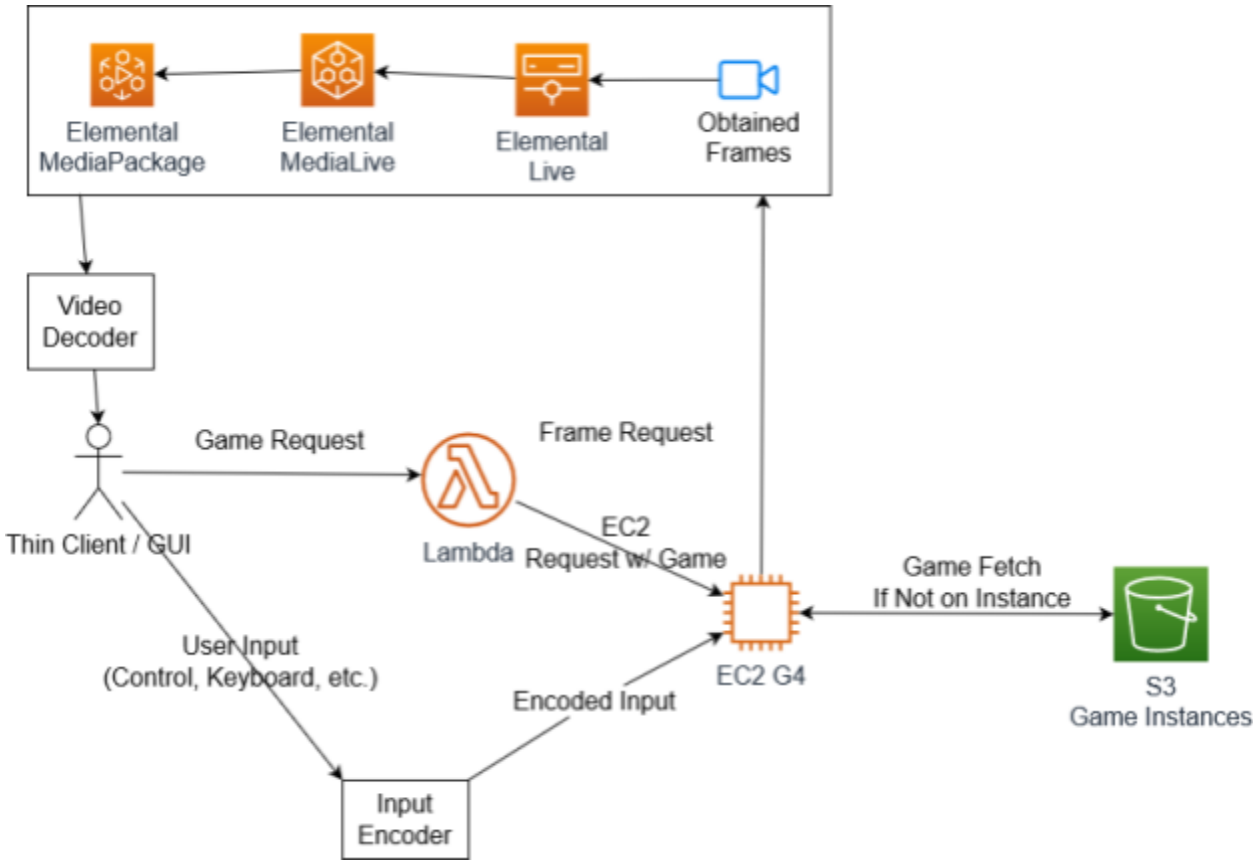
b. Alternative



Streaming Service

Both implementations for the streaming service assume the user has gone through the authentication process. Just as previously mentioned any service used requires a policy and user pool to request any AWS services in the implementation. EC2 instances in AWS are wrapped around VPCs by default so there is no need to add another VPC on the EC2 instance.

a. AWS



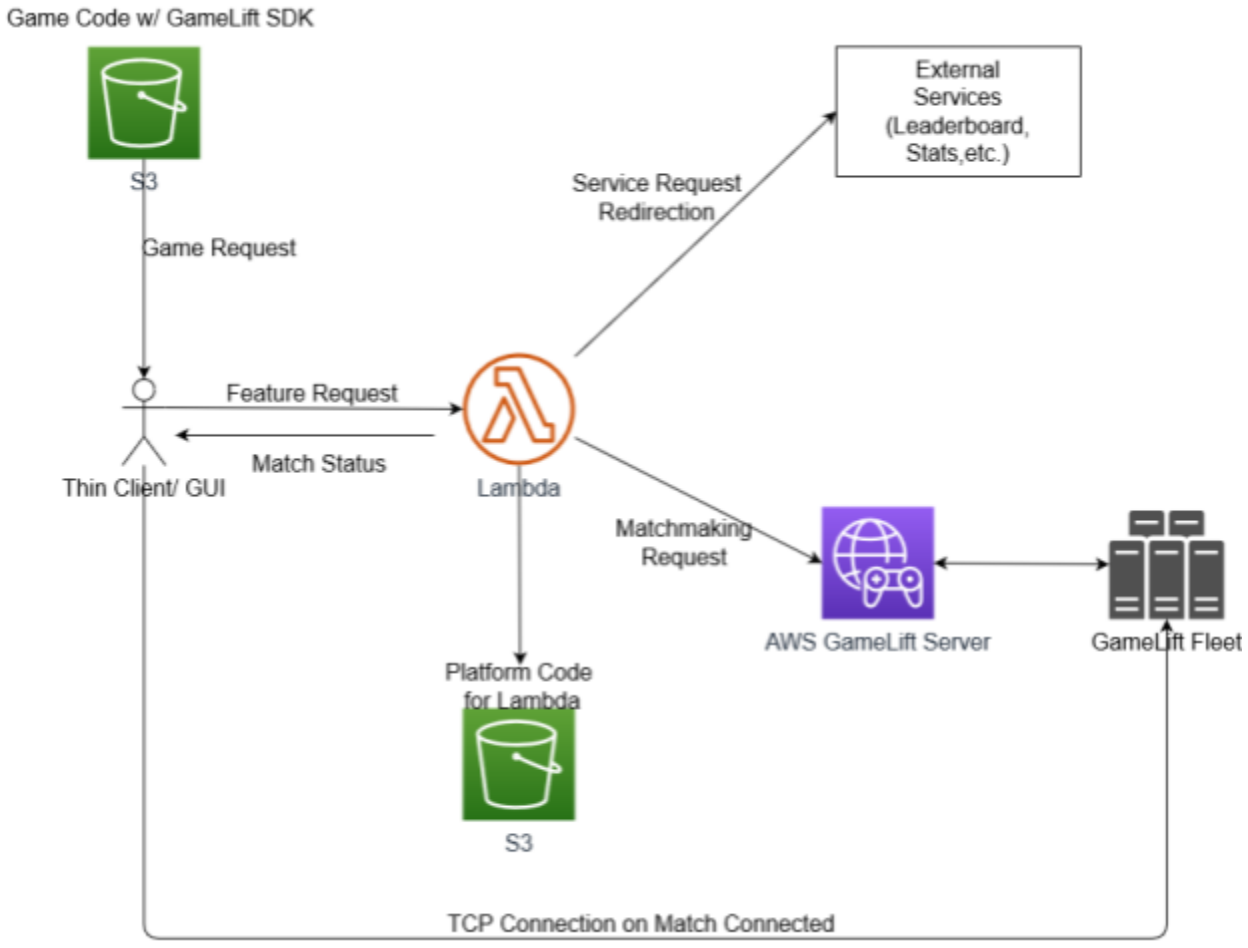
b. Alternative

The simplest alternative to developing a cloud game would be to obtain the software development kits (SDK) from companies currently offering, most notably Google Stadia, to integrate with a game. This would remove the need for GPU instances and streaming service. The developer only needs to embed the SDK within the game and direct input properly to the servers. Binaries can still be stored in S3 Buckets or any storage system of choice (might depend on the SDK provider).

Game Platform Services

Both implementation tracks assume the user has gone through the authentication process before accessing any game platform services. This section will also assume the user has already requested a game stream via the streaming service implementation section. If a game does not require a multiplayer session then the GameLift session may be removed entirely for the specific game. Just as previously mentioned any service used requires a policy and user pool to request any AWS services in the implementation. The GPU streaming service is considered as a game platform service but is covered in a separate section as it requires many components to create.

a. AWS



b. Alternative

A powerful alternative to AWS GameLift and other game platform services is Nakama by Heroic Labs. Nakama provides services such as matchmaking, leaderboards, in-game transactions, social sign-on, and more removing the need to create individual scripts for such services via lambda functions. The

developer would only need to use the API provided by Heroic Labs and add it to the game code in a similar fashion to GameLift.

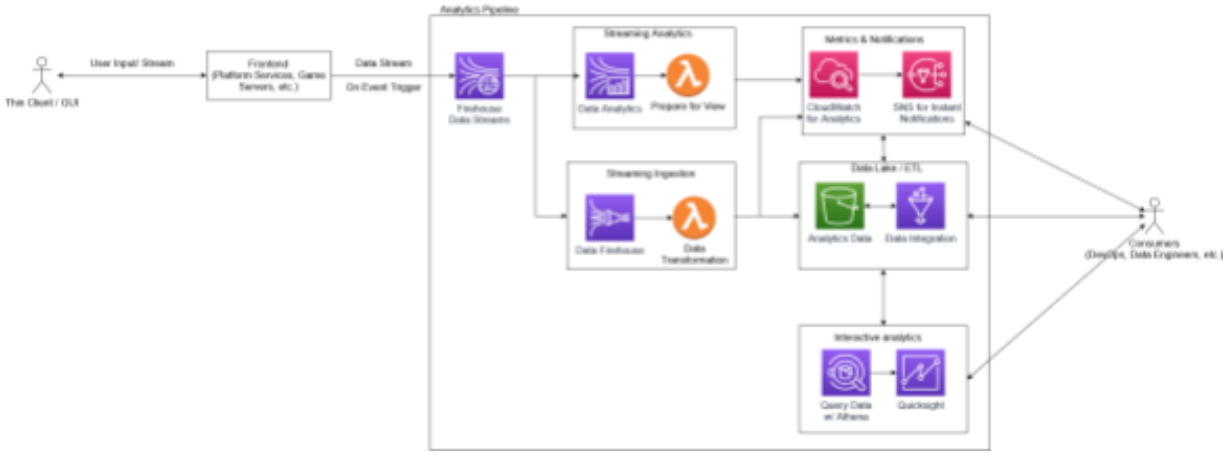
Databases

Databases in both implementations are interchangeable. Here there is no explicit implementation guide as there are a number of databases that can be used in each section. Many sections have already shown various databases to use with mostly NoSQL databases used. One important issue to note in each section is that there is no single database type to use, that is to say SQL and NoSQL databases have their benefits in different areas of game development. The same can be said for each company and their respective databases.

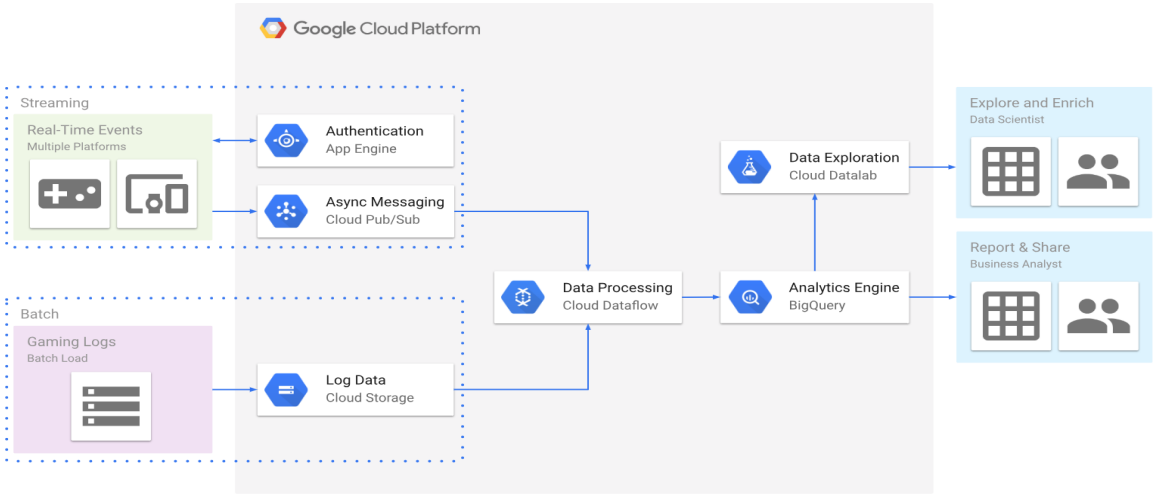
Analytics

Both tracks require a storage system for the data analysis. S3 and other databases may be interchangeable with some effort and proper policy creation. This section assumes the data is on a publish-subscribe model and has the proper rules are created for each model. The rules may depend on the game and server type used. Rules may include obtaining data of multiplayer games in matches, interactions between player chat logs, and so on.

a. AWS

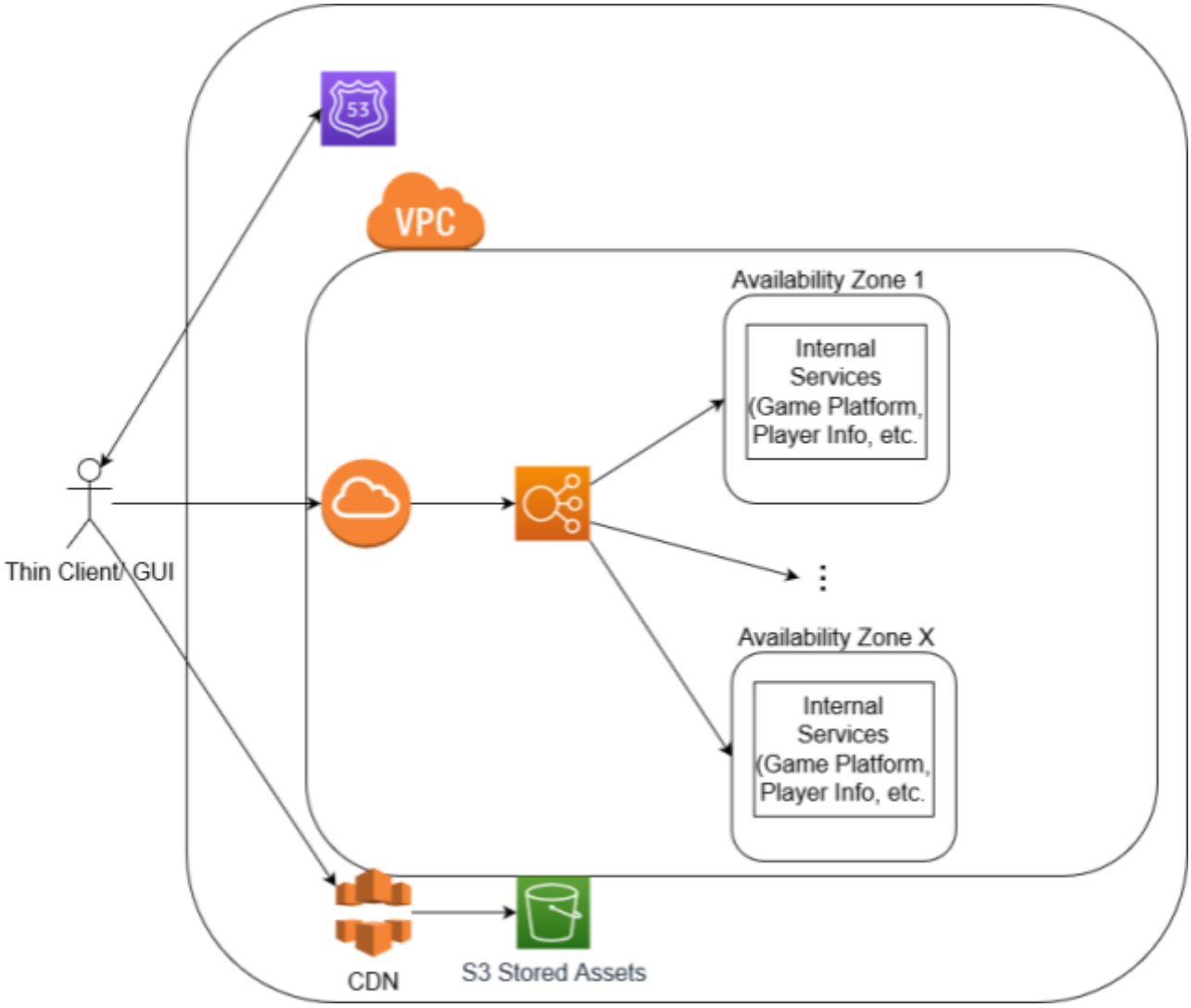


b. Alternative



Multi-Region and Load Balancing

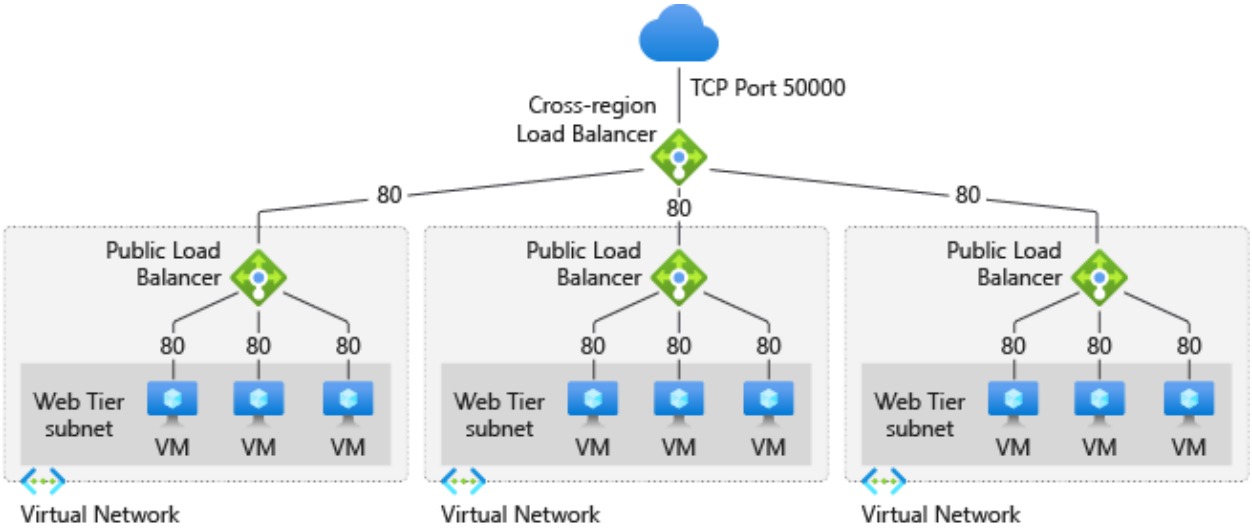
a. AWS



b. Alternative

Cross-region load balancing offers the same benefits of high performance and low latency as regional standard load balancers. The frontend IP configuration of your cross-region load balancer is static and advertised across most Azure regions.

The health probe of the cross-region load balancer gathers information about availability every 20 seconds. If one regional load balancer drops its availability to 0, cross-region load balancer will detect the failure. The regional load balancer is then taken out of rotation.



Conclusion

Summary

Overall much research went into the seminar. Game development has proven to be difficult due to the learning curve of many of the game technologies. The world of cloud gaming is also a relatively new concept compared to many other of the existing technologies and requires careful thought when architecting any cloud gaming system to ensure the user obtains the content they purchase. Current technology such as AWS GameLift is in a growing state with some issues and expenses that add another layer of difficulty.

The implementations presented are only a subset of the implementations which exist. As time goes on the cloud gaming industry is projected to grow along with all the technology provided by cloud providers such as Google and AWS.

Problems Encountered and Solutions

A major problem encountered while experimenting and developing with cloud gaming was the cost of both matchmaking services and GPU services for streaming games. Although the cost may seem miniscule per hour, over time the servers must be kept live for matchmaking systems to ensure there can be matches for each player. The GPU services for streaming games are costly and require proper implementation in order to make sure players have the lowest latency possible. Solutions may include finding cheaper services at the cost of compute power.

Better Approaches to the Project

A better approach to the reference architecture would be to look at how various components with different cloud providers interact with one another to estimate costs of creating such a service. Adding infrastructure code to automate which services are needed and require scaling would also improve the overall performance from a system perspective.

Suggestions for Future Extensions

To further extend on the project and seminar would be to use current cloud gaming SDKs as mentioned in the implementation guide. Unfortunately obtaining SDKs from companies such as Google and Amazon can be difficult even with the status of a game development company but once obtained a game can be easily created for the cloud. With time these SDKs will be more easily obtainable by individual game developers to release their games on the cloud.

References

"Add FlexMatch to A Game." *Amazon GameLift*, Amazon, docs.aws.amazon.com/gamelift/latest/flexmatchguide/match-client.html.

"Akamai Game Delivery." *Akamai*, Akamai Technologies, www.akamai.com/us/en/solutions/industries/gaming.jsp#game-delivery.

"Amazon EC2 G4 Instances." *AWS*, Amazon, aws.amazon.com/ec2/instance-types/g4/.

Amazon. "Build a Turn-Based Game with Amazon DynamoDB and Amazon SNS." *Amazon*, aws.amazon.com/getting-started/hands-on/turn-based-game-dynamodb-amazon-sns/.

Amazon. *Episode 1: Intro to Analytics (AWS Game Tech Tutorial Series). YouTube*, YouTube, 5 Feb. 2021, www.youtube.com/watch?v=t34bj5tUZZE&list=PLuGWzrvNze7LPM6y8vbGTvhbtNxhRokkl&index=1.

AWS Databases for Games, Amazon, aws.amazon.com/gametech/databases/.

"Building a Mobile Gaming Analytics Platform - a Reference Architecture." *Google Cloud*, Google, cloud.google.com/solutions/mobile/mobile-gaming-analysis-telemetry.

Burney, Sabrina, et al. *Optimizing Download Delivery for Gaming*. Akamai Technologies.

"Creating a Realtime Script." *Amazon GameLift*, Amazon, docs.aws.amazon.com/gamelift/latest/developerguide/realtime-script.html#realtime-script-examples.

"Game Analytics - The Basics." *Game Analytics Maximizing the Value of Player Data*, by Seif Magy. El-Nasr et al., Springer London, 2013, pp. 13–40.

"Game Developers." *Game Developers | IBM*, IBM, www.ibm.com/cloud/gaming.

Google Developers. *Building a Scalable Multiplayer Backend in 5 Minutes. YouTube*, YouTube, 22 Mar. 2017, www.youtube.com/watch?v=3uqMdsrpzEU&t=4s.

"GPU Cloud Computing Solutions from NVIDIA." *NVIDIA Cloud & Data Center*, Nvidia Corporation, www.nvidia.com/en-us/data-center/gpu-cloud-computing/.

Limbachiya, Niranjan. "7 Different Types of Game Testing Techniques." *DZone*, DZone, 13 Aug. 2020, dzone.com/articles/7-different-types-of-game-testing-techniques.

Narumoto, Masashi, et al. "Caching Guidance - Best Practices for Cloud Applications." *Guidance - Best Practices for Cloud Applications | Microsoft Docs*, Microsoft, 4 May 2017, docs.microsoft.com/en-us/azure/architecture/best-practices/caching.

"Overview of Cloud Game Infrastructure | Cloud Architecture Center." *Google*, Google, cloud.google.com/solutions/gaming/cloud-game-infrastructure.

Riesgo, Ignacio. "Improving and Securing Your Game-Binaries Distribution at Scale." *AWS Compute Blog*, Amazon, 12 June 2019, aws.amazon.com/blogs/compute/improving-and-securing-your-game-binaries-distribution-at-scale/.

Scott, Lee. "Scalable Cloud Gaming Architecture and Engineering for Mobile, Social Games Using Azure." *Channel 9*, MSDN, 2 Dec. 2014, channel9.msdn.com/Blogs/The-Game-Blog/Scalable-Cloud-Gaming-Architecture-and-Engineering-for-Mobile-Social-Games-Using-Azure.

"THE STATE OF ONLINE GAMING – 2020." *The State of Online Gaming – 2020*, Limelight Networks, 2020, www.limelight.com/resources/white-paper/state-of-online-gaming-2020/.