

# Games in the Cloud



Juan Davalos

Purva Choudhary

Mayur Rahangdale

Narbeh Movsesian

# Table of Contents

- History of Games
- Gaming on the Cloud
- Use Case 1: Cloud Gaming in Education
- Use Case 2: Google Stadia
- Reference Architecture of a Cloud Game-streaming Service
- Other Key Issues



# How does Cloud Gaming Work?



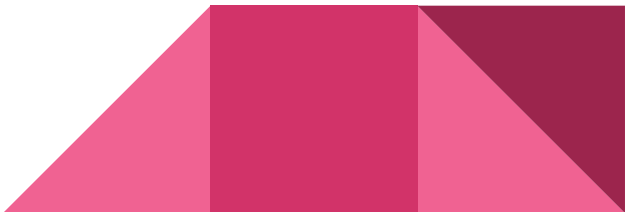
- Gaming on the go
- Drastically better performance
- No need for investing in high grade equipment
- Fraction of original costs

# Gaming

- What is Cloud Gaming?
- Date back to 1970s
- Rapid availability of the Internet in the 1990s
- Online networking features in 2000s



# History of Games

- Development of games for research and academic purposes
  - Games played via punch cards
  - Arcade video gaming
  - Gaming on home consoles and controllers
  - Computer Games
  - Mobile Gaming
- 



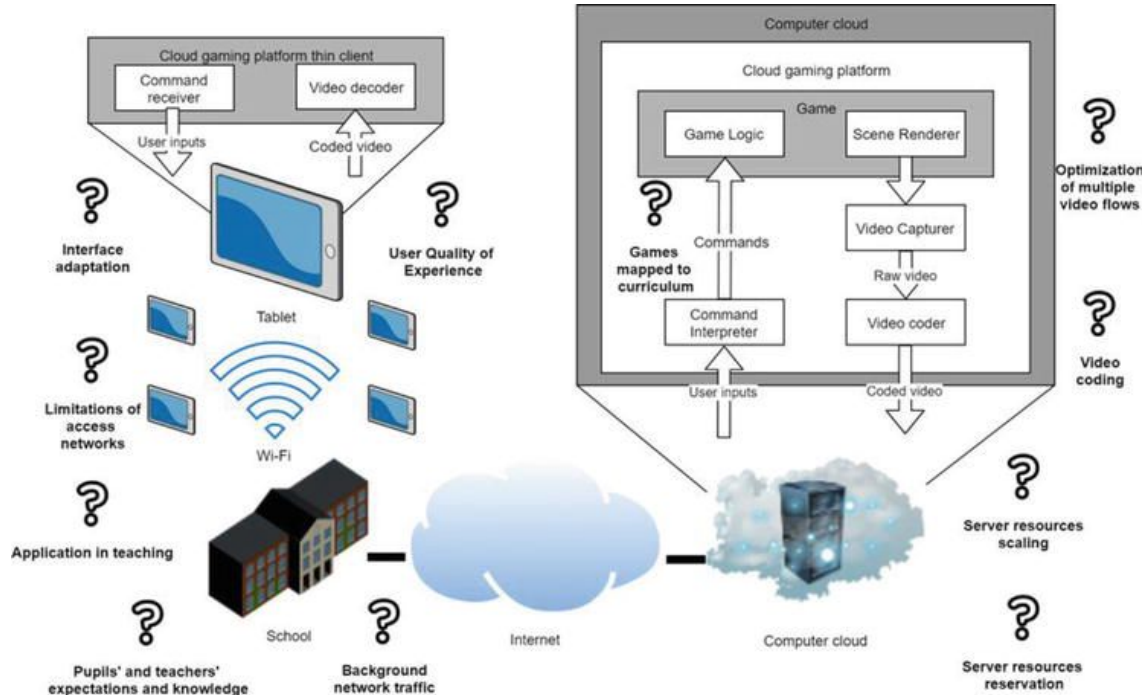
# Gaming on the Cloud

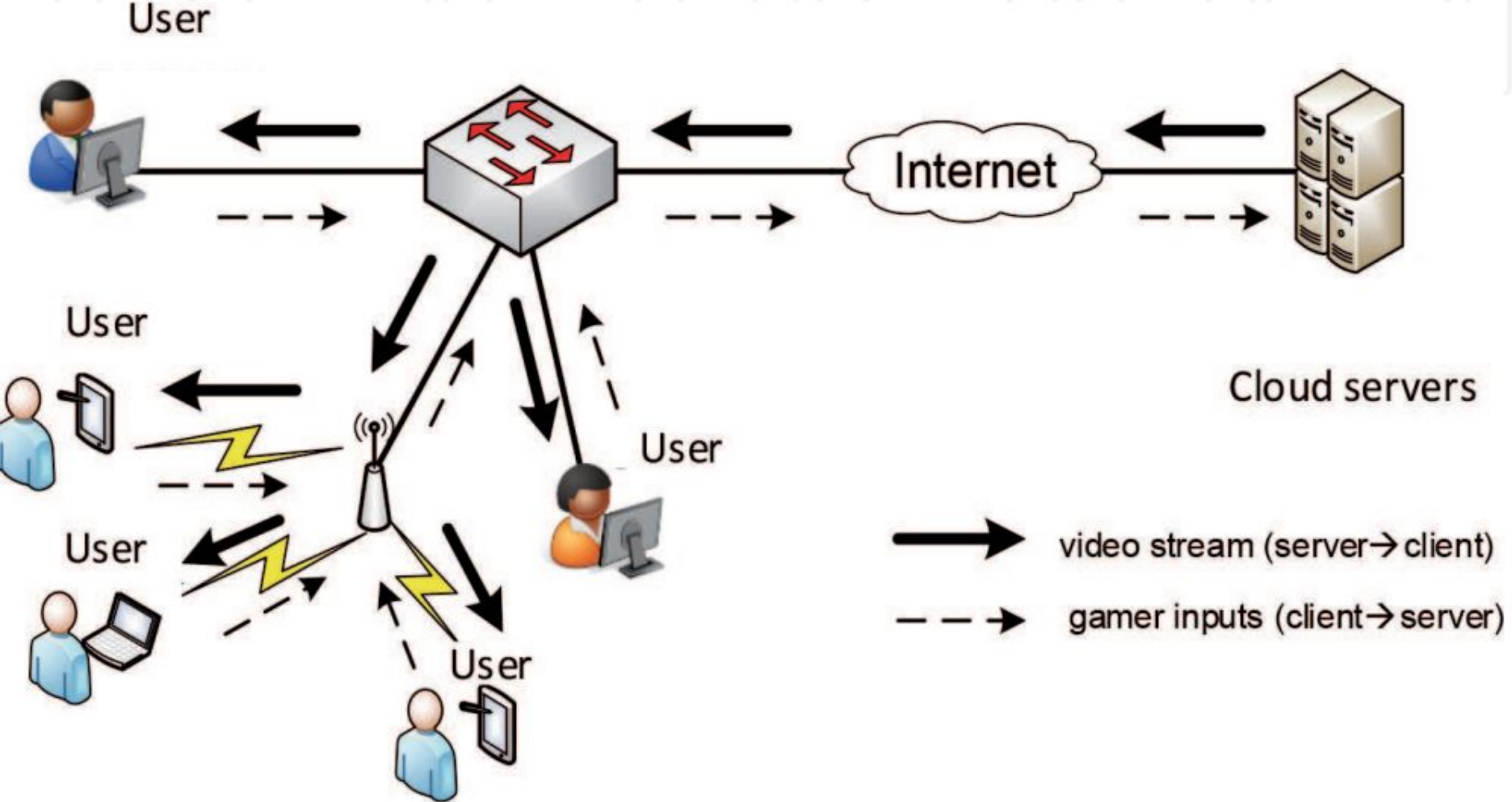
- Lower cost of technology, servers and the Internet
- Web browser as a client
- Web-based graphics technologies
  - Flash and Java
  - Ajax and WebGL



# Use Case 1 : Cloud Gaming in Education

- Limitations
- teachers' knowledge
- Quality educational games
- high cost

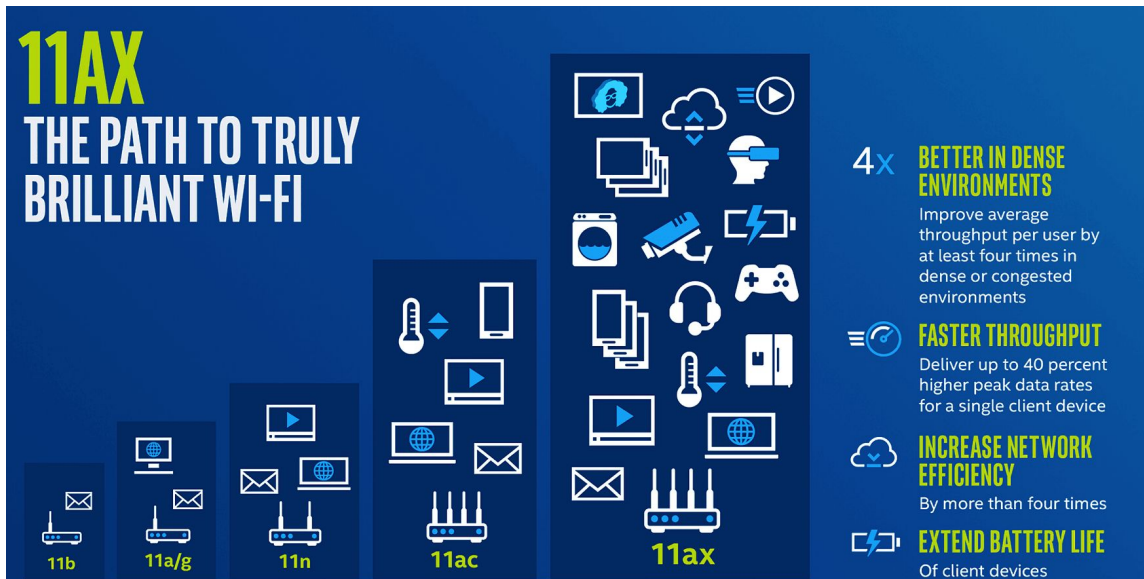






# Current Technical Problems

- WiFi connectivity as a bottleneck
- IEEE 802.11ax
- Algorithm Optimization
- Video Codecs



# Stadia- Cloud Gaming service by Google



## ***WHAT IS STADIA?***

Play any game on Chrome browser  
provided you have a high-quality  
internet connection

Requires connection speed of

**25-30** MBPS

Stadia lets you play 'any game', 'anywhere', 'anytime'.  
If the game is available on Stadia, you can play it  
in few seconds on any device with zero installation time.



● **2 out of 3** people use  
Chrome browser

● **1 out of 3** people is a  
gamer





# HOW?

Google has **three** major advantages that makes it possible to come up with such an offering

## ● Data Centers

Stadia makes use of Google's data centers all over the world with 7,500 edge nodes. Since they are spread across countries, it enables faster connectivity & seamless gameplay. And if you are curious about these nodes, they are the ones powering Google Search engine!

# 7500

EDGE  
NODES



## ● Stadia Controller

Although, Stadia can work with any controller/keyboard, it's offers an exclusive controller that can directly hook up to it's Data center over Wi-Fi. By letting the controller directly talk to the data center, it eliminates undesirable latencies that you would typically see in a traditional gaming console.

Stadia will work exclusively on Chrome browser. Considering that 63% of the world uses Chrome as its current browser, it's actually pretty good. I'd say this gives an unfair advantage to Google.

Google plans to extend support for other browsers in the future.

## ● Browser





# COMPETITIONS

Google might even go predatory by allowing **free access** upto some time and then charging eventually. Remember, if Google can get you hooked during the trial period, it is a guaranteed success!



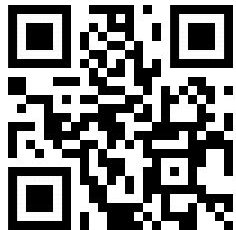
Billion-Dollar Gaming Industry



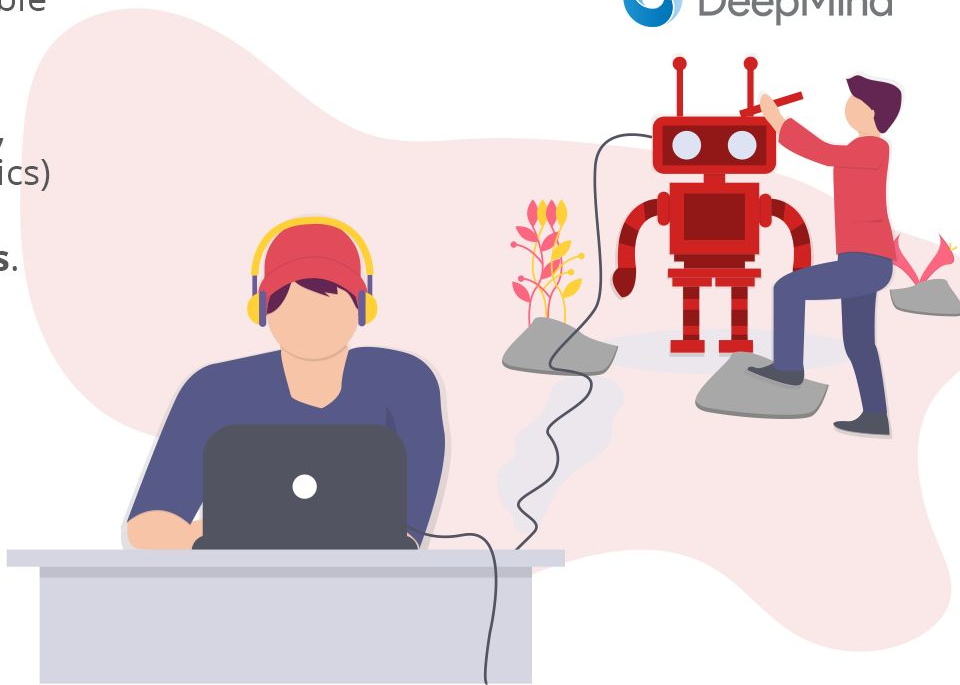
# *ARTIFICIAL INTELLIGENCE*

With Stadia, Google gets access to player data across wide range of games. This is going to be really valuable for its A.I!

Self-driving cars (Waymo), smart assistants (Duplex), Smart Navigation, Intelligent robots (Boston Dynamics) etc. can be optimized with systems that learn and understand to **solve complex real-world problems**.

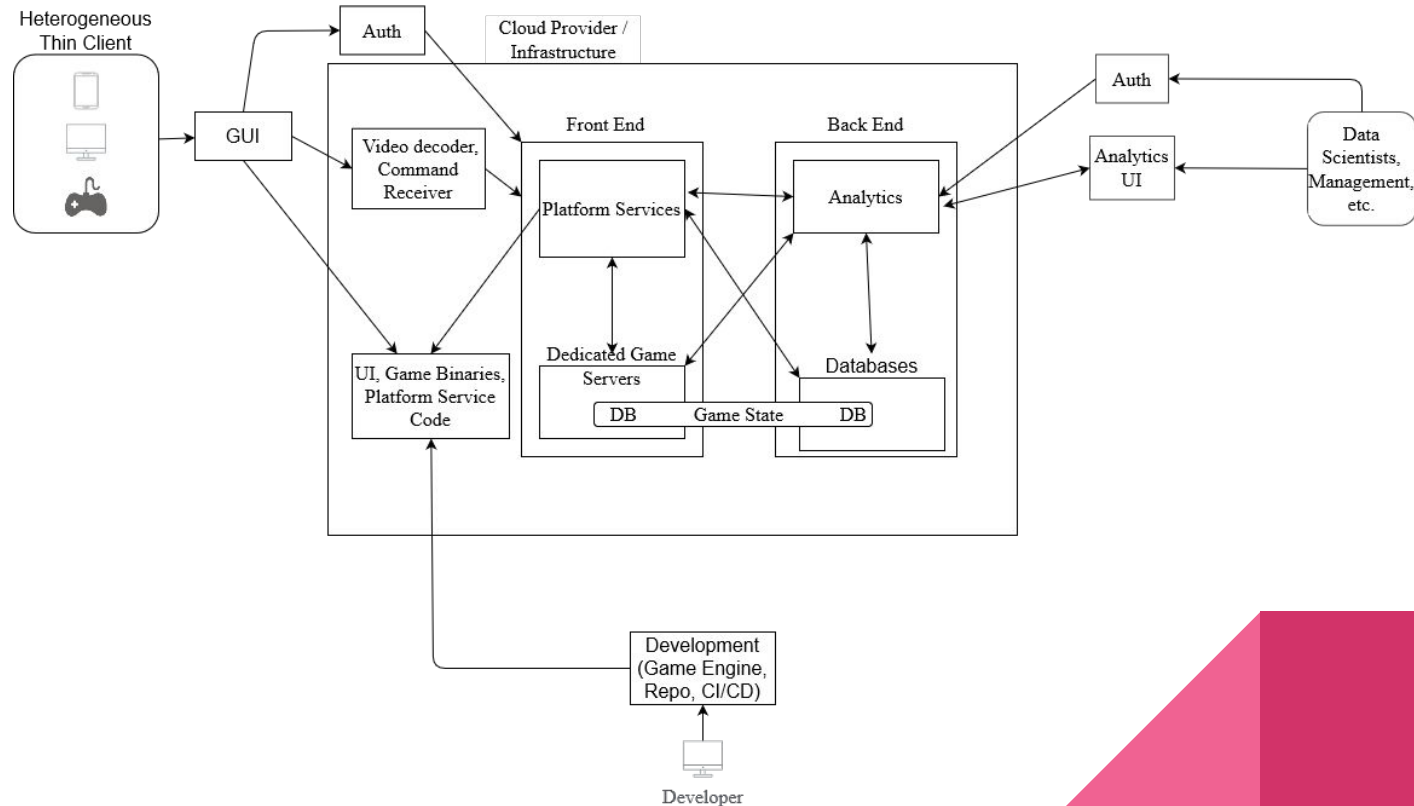


For us it's just a matter of playing game. For Google, it's an important step towards creating general purpose A.I that can solve many foundational problems.





# Reference Architecture - The World Map: A Brief View




# Development Land

- Some of the key points in this area:
  - Game Engines
  - Repositories
  - CI / CD and some modern practices
  - Releasing the games



# Development Land - Creating and Saving

- Developers typically need game engines to develop high quality games as they provide a number of benefits:
    - Ability to interact with the GPU independent of the GPU manufacturer.
    - Ability to develop for various platforms, allowing for cross-compatibility.
  - Scripts and assets need to be stored in a repository but may have the issue of being too large for standard Git version control. Software like Git LFS solves this issue.
  - The code for the various dev teams might include Game Platform Service Code (more on that later) and Infrastructure code (preferably) requiring multiple repositories and pipelines for each.
- 

# Development Land - Game Engines / Game Frameworks

- There are a number of Game Engines to work with along with IDEs
  - Amazon Lumberyard
    - Very new to industry but has the perk of being completely royalty free.
    - Amazon Lumberyard is a rework and extension of the CryEngine.
  - Unity Game Engine w/ Visual Studio 20XX
    - Popular engine offering both 2D and 3D development.
  - Unreal Game Engine w/ Visual Studio 20XX
    - The upcoming Rider by JetBrains can replace Visual Studio.
  - Crytek CryEngine w/ Visual Studio 20XX
    - But can it run Crysis...?



# Development Land - Building

- In the CI/CD pipeline we go through a number of steps:
  - Testing in a number of ways: Functionality, Combinatorial, Ad Hoc, Compatibility, Clean Room, Tree, Regression, Performance
  - Packaging versions with different configurations for: QA, Game Stores, Demo purposes.
  - Send builds to storage (S3, etc.)
  - Send builds to beta testing software.
- Code pushed to the repository should always go through a Continuous Integration / Continuous Delivery pipeline.
  - This may also have issues with size of the game as games can range anywhere between a couple of megabytes (MB) to tens of gigabytes (GB).
- Size of game can cause issues with builds in the pipeline and running the pipeline at every commit is inefficient
  - A modern approach to builds to break up the build in to containers.
  - These smaller builds can then be collected by the server running the streaming service as pieces put together on the server end, reducing fetching/download times.
  - When updates occur the user only needs to update certain smaller binaries.





# Development Land - Releasing

- Typically after testing builds must be pushed somewhere and the server instance needs to access these in some way to stream.
- As mentioned previously pushing to storage services such as AWS S3 (sufficient for smaller games) or breaking the binaries into multiple containers to be fetched when the user requests a game.



# Users' Republic

- Some of the key points in this area:
  - Accessing the games as a user.
  - Authentication for the user's data.



# Users' Republic - Accessing a Game

- A new method of accessing is via streaming the game similar to Netflix or Hulu Services.
  - This method requires low latency between the client and the backend services.
  - Users will also need a larger bandwidth depending on the game played.
    - First Person Shooters(FPS), Real-time Strategy (RTS), and other multiplayer games will require the user less delay.
    - Turn based games may require lower bandwidths and can have slightly higher latency
- The user accesses owned games via a web client or native application.
  - To access their games the user must provide credentials.
  - Once the user has selected their game events will trigger to launch the game on an instance via the front end services.



# Users' Republic - Authentication

- Users need to be authenticated in order to save their data on the cloud, view data from the cloud (account info, games owned, etc.)
  - Single Sign-On from the client to access user information.
  - What if the client offers games from different game developers?
    - SSO can solve that issue as it allows for a single authentication into multiple applications and can be used alongside developer's APIs to authenticate the user.
  - What about "in-house" games?
    - SSO can still be used as games developed under the same company will just verify the user exists in the databases.



# Users' Republic - Authentication (Cont.)

- To add more security using Multi-Factor Authentication (MFA) or Two-Factor Authentication (TFA) when accessing sensitive such as credit card information when buying games or in-game content.
  - Using online payment systems' APIs such as PayPal's APIs can reduce the complications of maintaining user payment information or at least offer another option.





# The Thin Client

**Thin Clients** are computing devices available at user's disposal

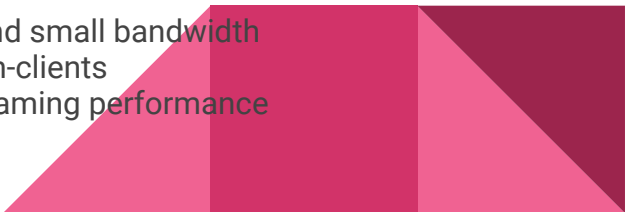
## A thin client usually has

- Low-performance hardware
- Browser-based Operating System
- Diverse I/O system support
- High network bandwidth support

## Benefits

- Cross platform support over different computing devices(Desktops, Smartphones, Smart TVs etc.)
- Cost and Energy Efficiency

## Design Challenges

- Robustness against network impairments such as network delay, packet loss, and small bandwidth
  - Optimizing for network latency as opposed to bandwidth issues in designing thin-clients
  - Display frame rate and frame distortion at the client side both being critical to gaming performance
- 

# Front End Empire

- Some of the key points in this area:
  - Game Platform Services so users can access online functionality.
  - The game servers and the game state.



# Front End Empire - Game Platform Services

- What are game platform services?
  - Services provided by various cloud providers to handle in-game and out-of-game functionality.
  - At this layer load balancing occurs to ensure the user does not run into latency issues or is left in a large waiting queue.
    - Without auto-scaling from the platform the servers can run into unfavorable issues such as flooding the server with too many requests, potentially breaking the infrastructure.
- What are those services?
  - A number of in game services including but not limited to: chat, matchmaking, leaderboard, authentication, penalization, player profile, and analytics (more on analytics later).



# Front End Empire - Game Platform Services (Cont.)

- How does the platform service code work?
  - Developers / Infrastructure team needs to create scripts based on the services used such as scaling.
  - The service is run in a serverless function, connecting the user to the appropriate platform service and/ or server.
  - The client requests one of the services provided by cloud providers and set up by the developers.
  - The service will then connect the user with the requested information, whether it be in game information or link to a server instance in the case of matchmaking.



# Front End Empire - Game Platform Services (Cont.)

- Why not create the platform services if we're already making the games?
  - Many modern games are catered for the multiplayer experience gameplay and require scaling when more and more users attempt to access a server/ match instance.
  - Good examples of multiplayer games with many players per match:
    - Warframe (up to 3 players per match)
    - Super Smash Bros. Ultimate (up to 4 players per match)
    - League of Legends (up to 10 players per match)
    - Overwatch (up to 12 players per match)
    - Apex Legends (up to 60 players per match)
    - Sea of Thieves (up to 99 players per server)
    - Minecraft (up to  $2^{31}$  players per server in theory)
      - Both Java and Bedrock Editions



# Front End Empire - Game Platform Services (Cont.)

- Why not create the platform services if we're already making the games?
  - The games mentioned tend to have thousands of players to millions of players live at any point in time.
  - All these require multiple instances of the server and housing all instances of the game at once can create high latency and cause dissatisfaction on the user end.
  - Cloud providers offering game platform services auto-scale horizontally to avoid the worry that too many server instances exist on a single computer.
  - Not to mention this can be extremely costly due to hardware when scaling "in-house."
  - We also don't want to have to build all of the hardware to host all of the streaming. It's an expense that will be high, much greater than paying for a GPU service to stream.



# Front End Empire - Game Platform Services (Cont.)

- These games serve multiple regions across the globe. How do platform services help?
  - Multiple regions depends on the cloud provider.
  - In the case of AWS GameLift, Google Cloud Gaming, and Heroic Labs Nakama allow for multiple regions across the world.
  - The regions can be customized to cater to specific regions with a set number of players playing per server.
  - Databases are also made region specific per instance (more on that later).



# Front End Empire - The Servers and the Game State

- Using platform services server instances can be created with the services providing a way to keep the game state stored and updated in a database.
- The created server will communicate with a database to keep the state as up to date as possible.
- Servers in the front end?
  - In a sense, yes. The thin client needs to interact directly with the servers based on user commands.
  - The user does not directly interact with the servers but through the proxy thin client. Hiding the server instances further leads to high latency.





# The Game State Diplomacy

- What is the game state?
  - Video games can be considered large state machines and game state a token at a certain point in the state machine.
  - When users interact with one another or with the environment the state changes and must be updated for all users in the server instance.
- Transferring the game state between users causes plenty of issues.
  - Latency when transferring the game state between users is further increased especially with cloud gaming due to the middleman thin client.
- Solutions
  - Predict other client interaction before it is received.
  - Waiting for all clients to receive the new state.
    - Terribly slow and can cause latency.



# Back End Alliance

- Some of the key points in this area:
  - Databases and the game state.
  - Using databases for other purposes in games.
  - Game Analytics (Game Telemetry)
    - A focus on how analytics changes the state of the game and services used.



# Back End Alliance - Databases and the Game State

- In order to make sure we can distribute the game state to all users we need to somehow save it to a database.
- But why a database?
  - For the most part it works like a cache system where we can quickly store and distribute.
  - The database also helps when the game session crashes. Once the game session is returned we can distribute the last known state to all the clients.
  - Another benefit is that many cloud providers simplify the creation of a backup for in case something goes wrong.
- Cache system? Just use that.
  - There are plenty of updates and it might work but we also might want to store information about the game for analytics purposes.
  - If the ship goes down, so does the captain. If an issue occurs in caching system the game state and other data will be lost.



# Back End Alliance - Databases for Other Purposes

- As mentioned before we might want to want to analyze the current state of the game from a bigger picture.
  - In-game changes and internal infrastructure changes may happen as a game increases in size.
- User information such as games owned, time played, or any other purchases should be kept for both user purposes and analytics purposes.
- Backups, backups, backups.
  - Backups to data should always be considered.
  - Distribute these geographically and on a regular bases.
- Archiving when?
  - It might be good to archive game data on a daily basis if the game garners a large audience due to the sheer amount of data produced.
  - User data might be better to hold off.




# Back End Alliance - Analytics and Changes

- In terms of analytics there are many areas to observe.
  - Monitoring and analysis of servers, devices, user behavior.
  - Even user behavior can be broken up into more sections such as purchasing behavior interaction with other users, interaction with services in the application.
- A lot of these areas will change the state of the game and even the infrastructure.
- But what do we save and look at?
  - Really just a bunch of raw data to be parsed down from the database.
    - A good example is looking at how players behave in certain areas of maps in games.



# Back End Alliance - Analytics and Changes (Cont.)

- Before analyzing any data we can create variables based on the player such as ID, session lengths, characters used, areas explored, etc.
    - This data is a “snapshot” in time of the game state.
  - After parsing the data we can display it via a UI for any number of internal employees (data scientist, management, even game designers).
    - Any number of data analytics services from data visualization platforms such as Tableau.
  - The data can lead to removal of services such as in-game group chat.
    - Case of this was in 2020 when Riot Games, creators of League of Legends and Valorant, removed the “clan” service where users could create groups to discuss. A number of reasons such as misuse of the service, private selling of “tags,” and expense prompted analysts to push for the removal.
- 

# Back End Alliance - Analytics and Changes (Cont.)

- More on analytics in *Game Analytics: Maximizing the Value of Player Data* published by Springer London.
  - Although an older book it paves the way to modern game analytics and is a good read to understand the importance of game analytics.



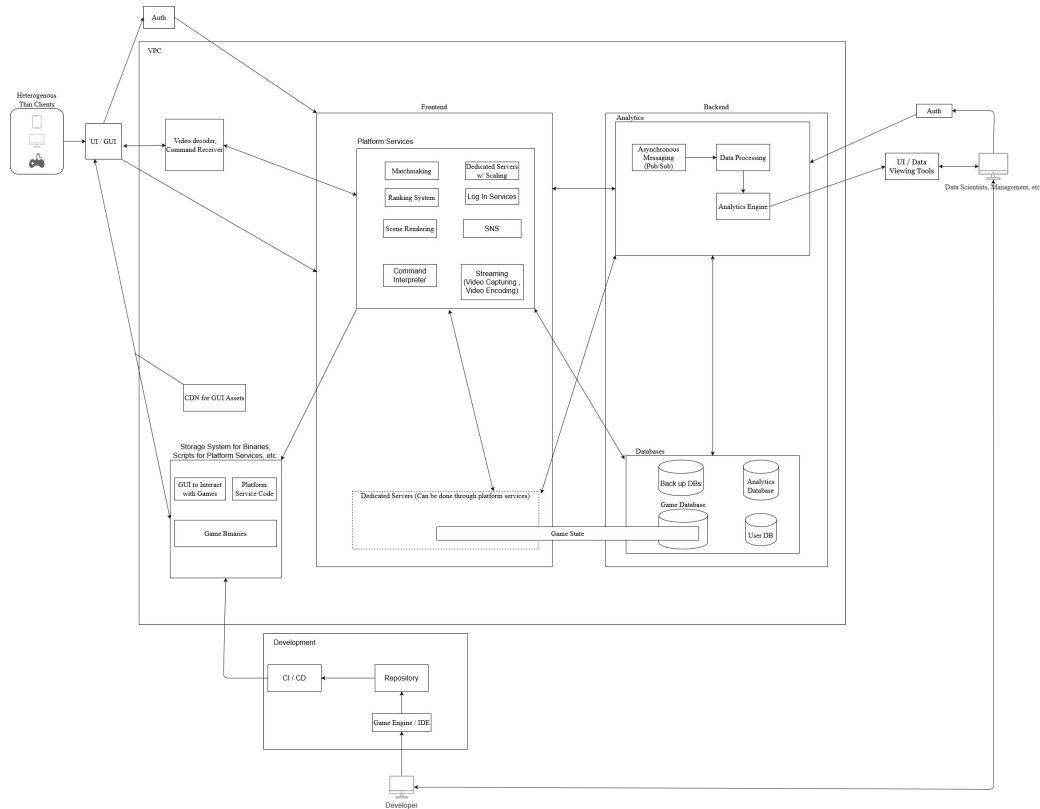
# Back End Alliance - Analytics and Changes (Cont.)

- Like in any other application you will want a number of tools to simplify the data analytics.
  - Asynchronous messaging systems like a Pub/Sub system to be able to collect data during any game session.
  - Data processing tools to view the data passed by the messaging system.
    - The data will be parsed and sent to any number of analytics engines.
  - Analytics engines allows us to have processed data in a simple to view way.
  - All of the data can then be viewed through a data visualization tool.





# Reference Architecture: The Complete World Map



# Some Key Issues You're Probably Wondering

- What about multiple users when they access the website / application that streams?
  - Typically you would want a load balancer between the client and the services so the user has a little latency as possible in any part of the application.
- What manages the infrastructure anyways?
  - That can be upto the architect. We recommend the Infrastructure as Code (IaC) approach as it promotes a healthy infrastructure without needing to manually set scripts and configurations.
- What about setting up those servers for rendering?
  - This is the tough part and requires the creation of a service to stream.
  - One option as mentioned is to rent GPUs that communicate with both the platform services and the client.
  - At the time of this research Google Stadia and Nvidia GeForce Now are two of the dominating providers that can be used for cloud gaming.
  - Amazon Luna is an upcoming service but is currently in early access.



# Some Key Issues You're Probably Wondering (Cont.)

- So, about setting up analytics?
  - In this seminar we didn't not go into depth on how to set up an analytics section on the backend. For the majority of the section there are areas that are similar to any other application's analytics.
  - Potentially the Pub/Sub model can be substituted if desired.
- So what's the CDN for in the reference architecture?
  - Naturally if we have a web page we probably want to cache static assets somewhere where all users can access quickly.



# Questions

