# 클라우드 게이밍 시스템 개발

**팀명: 금정산삼고라니**

201724579 정현모

201724539 이재욱

201724565 전설

부산대학교 전기컴퓨터공학부 정보컴퓨터공학전공

2022년 5월 16일

**지도교수: 김 원 석 (인)**

# 목 차

# I. 과제 배경 및 목표

게이밍 시장은 날이 갈수록 성장하고 있다. 그래픽 기술의 비약적인 향상이 이루어지면서 사람들은 고사양 게임을 요구하고 있다. 암호화폐 채굴과 부가적인 요인들로 인해 그래픽카드의 수요와 가격은 가볍게 생각하지 못할 정도로 상승하고 있다. 2020년 초 발매된 NVIDIA사의 3000번대 그래픽카드는 높은 가격을 주고도 구하지 못할 정도로 수요가 폭증하였다. 이에 우리는, 게이머가 하드웨어의 사양에 구애받지 않고 인터넷만 연결되어 있다면, 고성능의 게임을 즐길 수 있는 플랫폼을 구축하고자 한다.

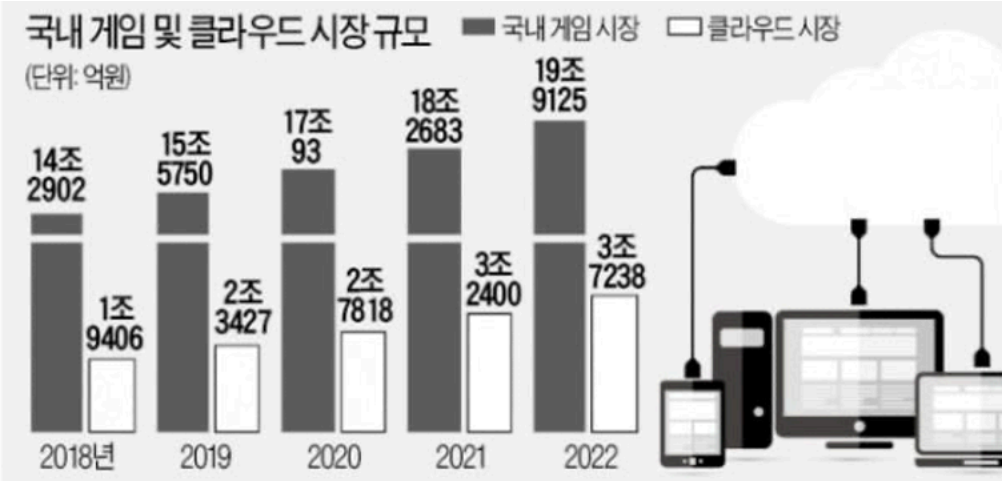


**그림 1. 클라우드 게이밍 서비스 구조도[1]**

Unity 3D 렌더링은 사용자의 하드웨어 사양에 따라 서비스 품질 편차가 있을 수 있다. 사용자들에게 동일한 서비스 품질을 제공하기 위해 개인의 하드웨어 사양에 구애받지 않고 서비스를 사용할 수 있도록 한다. 또한 OS마다 개발을 해야 하는 크로스 플랫폼 문제를 해결하기 위해 클라우드 게이밍 기술을 활용하여 서비스를 제공한다.

클라우드 게이밍이란 클라우드 컴퓨팅 기술을 이용한 게임 스트리밍 서비스이다. 사용자의 특정 하드웨어 장비 없이 네트워크 연결만 있으면 게임을 즐길 수 있는 서비스이다. 하지만 클라우드 게이밍 서비스의 문제로 네트워크 지연 시간과 많은 사용자의 트래픽을 상황에 따라 유동적으로 처리해야 하는 서비스 안정성도 중요한 문제이다. 현재 시중에 나와있는 클라우드 게이밍 서비스는 오픈소스가 아니라 성능, 버그 개선은 서비스 제공자에 의존적이다. 따라서 본 과제는 오픈소스 클라우드 게이밍 서비스를 구축함으로써 안정되고 원활한 게이밍 서비스를 제공하는 것이 목표이다.

[1] https://www.hankyung.com/it/article/2021032808091, 2021.03.28

# II. 요구 조건 분석

## i. 누구나 사용할 수 있는 클라우드 게이밍 플랫폼 제공

- 클라우드 서버와 브라우저를 활용해 사양 문제, 크로스 플랫폼 문제를 해결한다.
  - Unity Rendering Streaming

    게임 엔진 Unity의 Unity Rendering Streaming 기술을 활용하여 비디오, 오디오를 스트리밍하고 사용자 입력을 브라우저를 통해 받아 게임 내의 객체와 상호작용을 한다.
  - WebRTC

    사용자한테 비디오, 오디오 스트리밍을 제공하고 사용자의 입력을 받기 위해 WebRTC 기술을 활용한다.

## ii. 안정한 서비스 제공

- 3D 렌더링이라는 많은 작업 연산을 효율적으로 처리하기 위해 컨테이너 자동화 도구와 유동적인 트래픽을 관리하기 위해 로드 밸런싱을 활용하여 적절하게 트래픽을 제어한다.
  - Kubernetes

    컨테이너 오케스트레이션 도구인 Kubernetes로 컨테이너를 관리하면서 안정적인 서비스 제공을 한다.
  - Nginx

    Nginx의 리버스 프록시 기술을 활용하여 사용자들의 트래픽 유동적 처리, 비정상적인 요청 분류, SSL 적용 등 트래픽 처리를 전담한다.

## iii. 서비스 구축

- 사용자들에게 제공하고자 하는 서비스를 구축한다.
  - React
    - 사용자들에게 웹 서비스 형태로 회원가입, 로그인, 게임 접속 기능을 제공한다.
  - Spring boot
    - React 웹 서비스와 게임 container의 요청을 DB에 반영해주는 API 서버를 구축한다.
  - PostgreSQL
    - 웹 서비스, 플랫폼의 회원정보 등의 데이터를 저장하는 DB를 구축한다.
  - Redis
    - 게임 정보를 빠르게 조회, 저장하기 위한 인메모리 DB를 구축한다.
  - GitHub Actions
    - 개발, 배포 프로세스 자동화하여 개발자가 서비스 품질을 높이는데 집중할 수 있도록 CI/CD 프로그램을 활용한다.

## iv. 버그 및 이슈 발생 시 빠른 대처

- 모니터링 환경을 미리 구축하여 버그나 이슈가 발생했을 시 원인을 빠르게 찾고 해결한다.
  - ELK Stack

    시스템 내에서 발생하는 로그들을 수집하고 시각화로 안정적인 운영에 도움을 준다.
  - Prometheus + Grafana

    시스템 자원들의 메트릭을 수집하고 시각화 하여 안정적인 서버 운영에 도움을 준다.

# III. 시나리오

## i. 개발자 관점

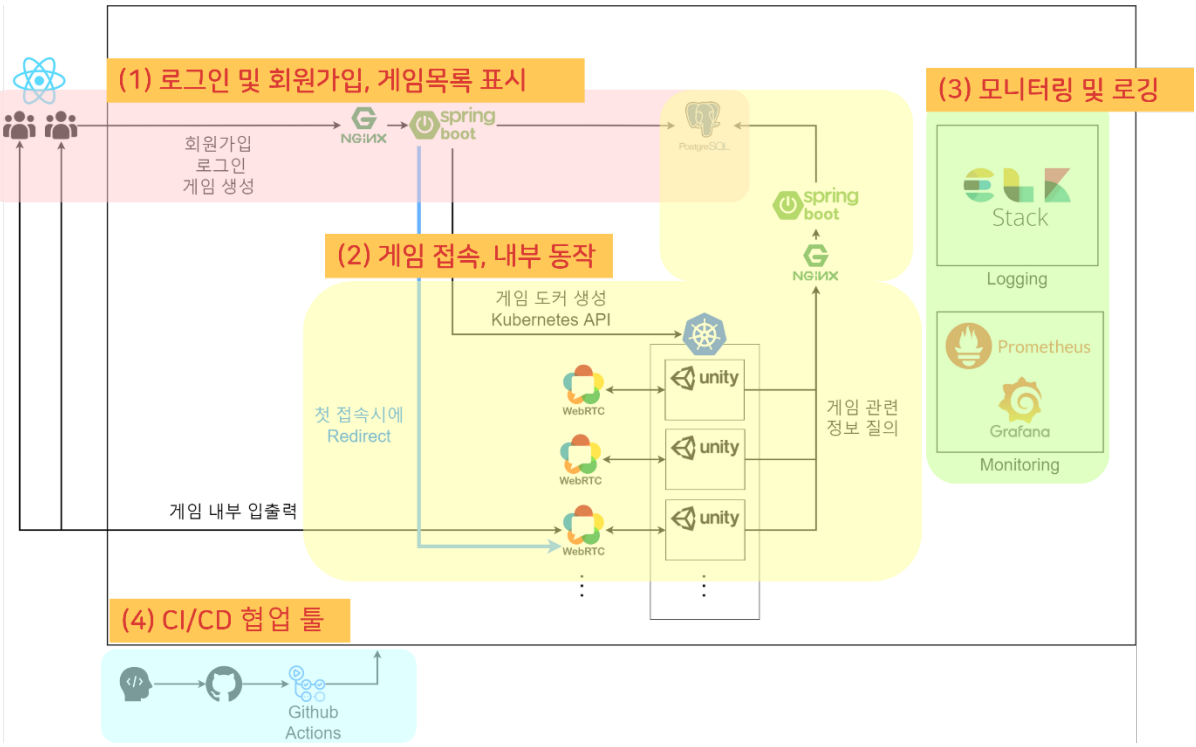


**그림 1. 클라우드 게이밍 서비스 구조도**

- Unity 로 WebRTC 기반의 Unity Render Streaming 기술을 활용하여 클라우드에서 동작 가능한 게이밍 플랫폼을 제작한다.
- 사용자, 서비스에 필요한 부분들을 기획하고 DB 를 설계한다.
- 로그인/회원가입 기능을 담당하는 API 와 게임목록 조회, 생성, 삭제를 담당하는 API 를 설계하고 개발한다.
- Unity 와 통신하며 각 게임 별 자원을 관리하고 제공하는 API 를 설계하고 개발한다.
- React 프레임워크를 이용해 사용자가 접근 가능한 프론트엔드 부분을 개발한다.
- 개발한 서비스들을 컨테이너화 시키고, 각 컨테이너들을 자동으로 관리할 컨테이너 오케스트레이션을 구축한다.
- 각 서버들과 Gateway, Kubernetes 에서 생기는 버그나 이슈를 빠르게 확인하고 처리하기 위해 ELK 를 활용한 Logging 과 Prometheus, Grafana 를 활용한 모니터링 환경을 구축한다.
- 유지보수를 위해 Github Action 을 활용해 자동 업데이트 / 배포 시스템을 구축한다.

## ii. 사용자 관점

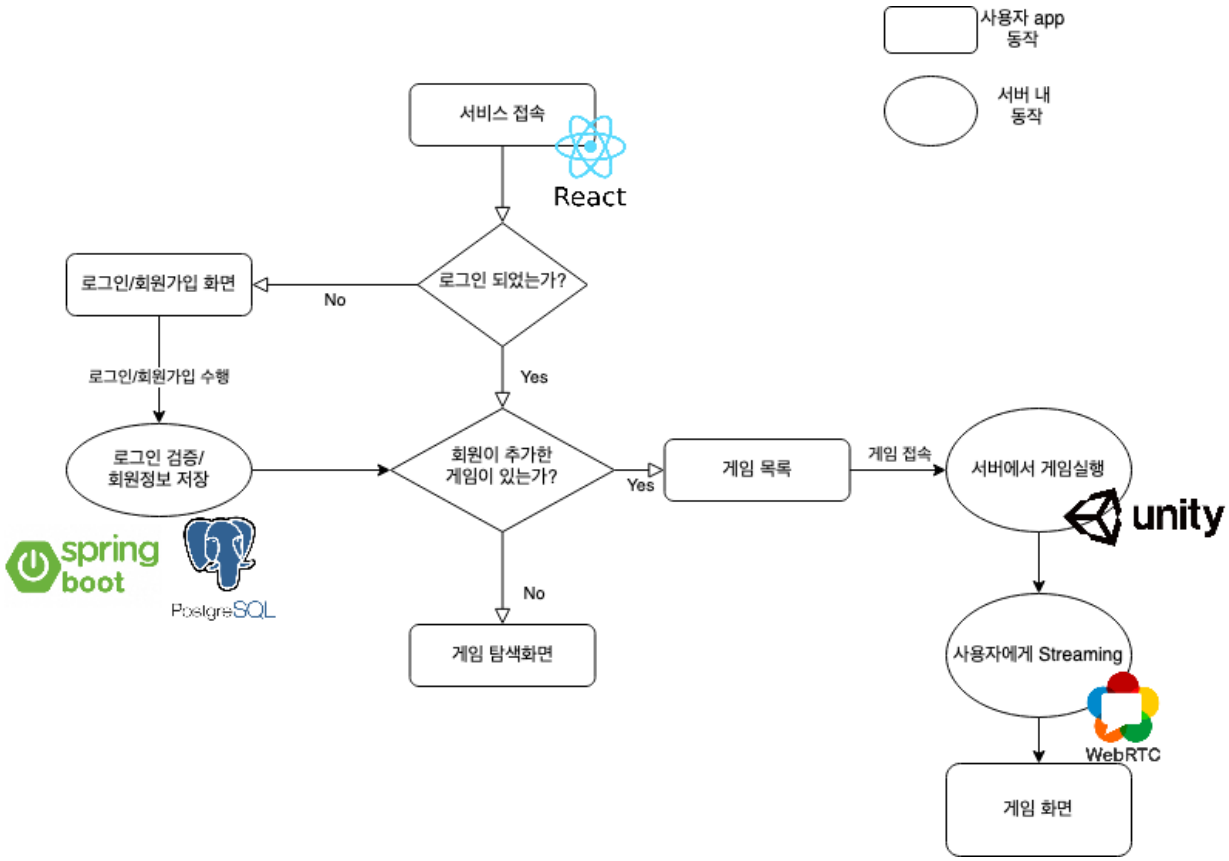


**그림 2. 클라우드 게이밍 서비스 사용자 관점 플로우 차트**

- 사용자는 웹 브라우저를 통해 서비스에 접속한다.
- 로그인 / 회원가입을 통해 회원 권한을 얻는다.
- 회원이 추가한 게임이 있다면 게임 목록 페이지를 표시한다(메인 페이지).
  - 그렇지 않다면 게임 탐색 페이지를 표시한다.
- 탐색 페이지를 통해 추가한 게임을 클릭하여 해당 게임으로 접속한다.
- 서버에서 Streaming 되는 게임 화면을 브라우저를 통해 중계 받는다.
- 키보드, 마우스 입력을 서버의 게임에 전달하여 게임을 조작한다.
- 게임 종료 시, 메인 페이지로 이동하는 링크를 표시한다.

# IV. 개발 일정 및 시나리오

## i. 개발 일정

| 5월 | | 6월 | | | | | 7월 | | | | 8월 | | | | 9월 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3주 | 4주 | 1주 | 2주 | 3주 | 4주 | 5주 | 1주 | 2주 | 3주 | 4주 | 1주 | 2주 | 3주 | 4주 | 1주 | 2주 | 3주 | 4주 | 5주 |
| 환경구축 | | | | | | | | | | | | | | | | | | | |
| | | Unity 개발 | | | Unity 개발 | | | | | | | | | | | | | | |
| | | | | | | | WebRTC 개발 | | | | | | | | | | | | |
| | | | | | | | 컨테이너 구축 | | | | | | | | | | | | |
| | | | | | | | 서버 및 DB 구축 | | | | | | | | | | | | |
| | | | | | | | | | 중간보고서 준비 | | | | | | | | | | |
| | | | | | | | | | | 로깅 및 모니터링 구축 | | | | | | | | | |
| | | | | | | | | | | | 안정화(테스트 및 디버깅) | | | | | | | | |
| | | | | | | | | | | | | | | | | | | 최종보고서 준비 | |

## ii. 역할 분담

| 이름 | 역할 분담 |
|---|---|
| 정현모 | API gateway 서버(Spring Boot) 개발<br>웹 서버 프록시 및 로드 밸런싱(Nginx) 개발 |
| 이재욱 | Unity 및 WebRTC 개발<br>모니터링 구축(Prometheus, Grafana)<br>로깅 시스템 구축(ELK)<br>DB, Redis 설계 |
| 전설 | Unity 및 WebRTC 개발<br>프론트 개발(React)<br>CICD (Github Actions) 구축<br>컨테이너 구축(k8s) |
| 공통 | Unity Render Streaming 기술 연구<br>아키텍처 설계 |

# V. 참고 문헌

[1] About Unity Render Streaming, 2022. 2. 28 작성.

https://docs.unity3d.com/Packages/com.unity.renderstreaming@3.1/manual/index.html

[2] Overview| Prometheus.

https://prometheus.io/docs/introduction/overview/

[3] 마이크로서비스 데이터베이스 분리 설계, 2021.11.22 작성.

https://waspro.tistory.com/729

[4] MDN, WebRTC API, 2021.12.09 작성.

https://developer.mozilla.org/ko/docs/Web/API/WebRTC_API

[5] 한경, 폭풍 성장 한국 게임사 잡아라, 2021.03.28 작성.

https://www.hankyung.com/it/article/2021032808091