



[서버] 제1차 비정기회의 (로드맵)

기록 담당자	박영서
분류	서버
진행일자	@2023년 9월 1일
참석자	김하림 박영서 이건

0 목차

0 목차

1 서버 팀의 프로젝트 목표

2 서버 팀 To Do List

- [1] 형상 관리
- [2] API 서버 구현
- [3] CI/CD
 - 1. CI
 - 2. CD ⚠
- [4] 공식 사이트
- [5] 스터디
- [6] 신기술 도입 및 네트워크 최적화

3 회의 의견

- [1] Perforce 관련
- [2] 서버 관련

4 주차 별 로드맵

- [1] 3주차 (09.04 - 09.10)
- [2] 4주차 (09.11 - 09.17)
- [3] 5주차 (09.18 - 09.24)
- [4] 6주차 (09.25 - 10.01)
- [5] 7주차 (10.02 - 10.08)

1 서버 팀의 프로젝트 목표

- 건 : 테스트
 - 하림 : CI/CD
 - 영서 : 전부 다(욕심쟁이) + TDD + CI/CD
 - 신기술 도입 ⇒ 화요일에 전문가 리뷰 받아보고 추가(3, 4주차에)
 - 공식 사이트 만들기
-

2 서버 팀 To Do List

[1] 형상 관리

- UE5 소스코드 및 에셋 관리는 **Perforce** 사용
- API 서버 + 공식사이트 FE는 **GitLab** 사용
- EC2에 Perforce 서버, API 서버, 공식사이트 FE 서버를 올려서 소스 컨트롤
- 이후 Jenkins + Docker를 이용해 CI/CD

[2] API 서버 구현

- **유저관리** - 회원가입, 로그인, 로그아웃
- **유저데이터** - 업적, 최단 클리어 시간, 클리어 횟수, 플레이 타임
- **버전 확인** - 패치 및 업데이트
- **세션 연결 관리** - 방 생성, 방 목록, 유저의 세션 출입 관리
- **SSE**를 이용한 게임 내 공지
- 일반 채팅 + 보이스 채팅 구현
- **빠른 시작** - 방 생성 시 빠른 시작 허용/비허용으로 나누어야 함 or 공개/비공개방

[3] CI/CD

1. CI

- Perforce/GitLab + Jenkins를 이용한 CI

2. CD

- NGINX를 사용하여 **프록시 서버** 구축 필요 ⇒ 공식사이트 FE 서버 + API 서버
- **패치 및 업데이트** 구현 필요 (패치 관련 공식 문서)
 - 1차 배포(5주차) 이후 테스트 기간에 계속해서 버그를 수정하고 콘텐츠를 추가하게 될 것.
 - 리슨 서버 ⇒ 클라이언트 사이에 버전 동기화 필요
 - 유저 클라이언트의 버전과 배포된 최신 게임 버전을 확인하고, 두 버전이 맞지 않는 경우 업데이트를 강요해야 함.
 - 따로 게임 버전 컨트롤 기능을 제공하지 않으면, 수정될 때마다 MM으로 “zip파일 다운 받아주세요” 해야 함.
 - 클라이언트 버전과 배포된 최신 버전 확인해보고 **업데이트 요청 ⇒ 수락 시 즉시 업데이트, 거부 시 클라이언트 종료**

▼ GameLift에서는 무중단 업데이트 지원 ⇒ 직접 구현


Accelerate your game development



플레이어 트래픽에 따라 서버 용량 자동 스케일링
갑작스러운 플레이어 피크 트래픽에 대비하면서도, 트래픽이 감소했을 때 유휴 서버 자원에 대해 비용 지출을 최소화.



유연한 매치메이킹
직접 개발한 매치메이킹 서비스를 활용하거나, 커스터마이징 가능한 FlexMatch를 활용하여 정의한 규칙에 따라 경쟁력 있는 매치메이킹을 수행.



무중단 업데이트
GameLift를 활용하여 새 콘텐츠나 긴급 패치를 위한 서버 업데이트를 무중단으로 배포.



크로스 플랫폼 플레이 지원
여러 디바이스들을 동시에 지원하여 플레이어 풀을 확장.



기존 게임 엔진 및 워크플로우와 통합
Unreal, Lumberyard 등과 같은 AAA 게임엔진을 사용하거나 자체 개발 C++기반 게임엔진을 사용하는 경우 모두 Amazon GameLift SDK를 통해 Cloud상에서 게임서버를 구동하고 운영하기 위한 통합작업을 구현.

[4] 공식 사이트

- 레퍼런스

1. 게임 소개

- 소개 영상(시연 영상)

- 간단 조작법
- 스토리 요약

2. 패치 노트

- 버그 수정 사항 및 추가 콘텐츠 공지

3. 게임 기록

- 로그인 한 회원의 클리어 시간, 업적, 플레이 타임 등
- 전체 랭킹

4. 게임 다운로드

5. 고객의 소리함

- 테스트 피드백

6. 굿즈(헉! 대박!)

[5] 스터디

- 소켓, 멀티쓰레드, 네트워크 프로그래밍 스터디
- 간단하게 IOCP 서버 구현

[6] 신기술 도입 및 네트워크 최적화

- 화요일(09.05) 전문가 피드백을 받은 이후에 결정
 - Kafka를 이용한 분산 처리
 - K8S를 통한 컨테이너 오케스트레이션
 - Perforce, Jenkins, API server, DB 모두 docker container로 관리
- 프레임드랍이 생기거나, 문제 발생 시 ⇒ 테스트 (레이턴시, GPU 부하, etc)

3 회의 의견

▼ 8/31 노션 회의

<https://www.inflearn.com/questions/970556/게임서버-엔진-섹션을-듣다가-궁금한게-있습니다>

🐝 할 게 없는 이유는... 일단 난이도가 중간이 없는 게 큰 것 같군요

- 🐝 짧은 시간에 전문성을 익히기가 어려움
- 🐝 그렇다고 툴을 쓰자니 대부분 다 해주고, 얻어가는 게 없음

🐝 하지만..... 서버는 어렵지만 도전해야 하는 부분이라고 생각합니다

🐝 간단하게라도 최소한 멀티플레이어 네트워크 연결은 해보는 것이 맞지 않나.. 생각보다 이 부분에서 게임 오류가 많이 발생하는 것 같아요

🐸 근데 이것도 사실 다 언리얼 엔진으로 해주는 거라 어떤 걸 해줘야 하는지

- 🐸 동기화 정도가 있겠으나 로우 레벨에서 할 수 있는 건 없는 거 같고
 - 얼마나 로우 레벨로 가게 될지
 - 🐸 저희 서버 구조 어떤 거 쓸지 정해 놓고 뜯어서 구현 해보는 건 좋을 거 같은데요👍
 - 🐝 다 써보는 건 어떠신지 / 네트워크 Optimize 하는 경우보면 각 서버 장점 살려서 하이브리드 방식으로 쓰는 것 같습니다

🐸 ⇒ 프로젝트에 적용하는 건가요 아님 스터디로 하는 건가요

🐝 ⇒ 일단 프로젝트 적용에.. 그래서 생각한건

1. 🐝 언리얼로 네트워크 구현 (각 방식으로)
2. 🐝 뜯어보기 or 활용하기
 - a. 🐝 중앙서버 단에서 관리해야 할 것(보스 관련)은 데디케이트 방식, 이외 부하(잡몹 설정 등)는 리스 방식 등

🐝 근데 결국 뭘 하든지 언리얼 사용법은 익혀야 할 것 같은

- 🐝 네트워크를 구현하고, 하나하나 뜯어가며 로우 레벨로 구현해보는 건 어떠신지
 - 🐝 각 함수를 뜯어서 이해하면, 동작 방식에 대해 알 수 있고 활용 가능하지 않나

🐝 사실 이런 거에서 전문가 피드백이 필요한건데, 이 부분에 대한 피드백은 못 받음
(싸피에서 언리얼 활용 게임 개발에 해박한 사람이 없)

🦩 언리얼 엔진 소스 코드 받기 : <https://netmarble.engineering/unreal-engine-dev-build-setting/>

🐰 언리얼 엔진 소스 코드 받는 방법 :
<https://m.blog.naver.com/sorang226/221754874279>

🐝 유데미 네트워크 강의 : <https://www.udemy.com/course/unreal-engine-5-cpp-multiplayer-shooter/>



언리얼 엔진 (UE)에서 멀티플레이 서버 환경을 구축하는 방법에는 여러가지가 있습니다. 언리얼 엔진 자체에는 멀티플레이를 지원하기 위한 네트워킹 기능이 포함되어 있습니다. 그렇기 때문에 언리얼만을 사용하여 기본적인 멀티플레이 환경을 만들 수 있습니다. 하지만, 상업적인 큰 규모의 프로젝트에서는 추가적인 서버 솔루션을 검토하게 될 수 있습니다.

1. **언리얼의 내장 네트워킹:** 언리얼 엔진은 기본적으로 멀티플레이어 게임을 위한 서버/클라이언트 아키텍처를 제공합니다. 게임에서 "호스트" 역할을 하는 플레이어가 서버를 만들고, 다른 플레이어들이 이 서버에 접속하는 방식입니다.
2. **Dedicated Server:** 언리얼 엔진은 전용 서버 (Dedicated Server)를 지원합니다. 이는 게임 로직만을 실행하는 서버로, 그래픽 렌더링 없이 백그라운드에서 작동합니다. 이 방식은 큰 멀티플레이어 게임에서 흔히 사용됩니다.
3. **Epic Online Services (EOS):** Epic Games는 개발자들을 위해 Epic Online Services를 제공합니다. 이는 크로스 플랫폼 멀티플레이어 서비스와 다양한 온라인 기능을 지원합니다.
4. **외부 솔루션들:**
 - **PlayFab:** Microsoft의 PlayFab는 멀티플레이어 서버 호스팅, 데이터 분석, 플레이어 데이터 관리 등 다양한 서비스를 제공합니다. PlayFab Multiplayer Server와 같은 서비스를 사용하여 언리얼 엔진과 통합할 수 있습니다.
 - **AWS GameLift:** Amazon의 GameLift는 멀티플레이어 게임 서버를 위한 스케일링 가능한 서비스를 제공합니다. 언리얼과 잘 통합됩니다.
5. **플러그인 및 중개 솔루션:** 언리얼 마켓플레이스에서는 다양한 네트워킹 및 멀티플레이어 플러그인을 찾을 수 있습니다. 그 중 몇몇은 기존 솔루션의 확장이나, 특정 기능에 초점을 맞춘 것들이 있습니다.

언리얼 엔진을 사용하는 프로젝트의 규모와 필요성에 따라 적합한 멀티플레이 서버 환경을 선택할 수 있습니다. 초기 프로토타입 단계에서는 언리얼의 내장 네트워킹을 사용하다가, 상용 프로젝트로 나아갈 때 외부 솔루션으로 확장하는 것이 일반적입니다.

[1] Perforce 관련

1. EC2 올리기 전에 **적응 기간** 필요하다고 판단
 - a. 오라클 클라우드는 100GB라 얼마나 올라갈 수 있을 지 모름
 - b. EC2 받기 전까지 임시로 사용 가능할 듯

➡ 🦆 **Oracle Cloud 가입이 안 됩니다... 고질적인 문제가 있다고 합니다. (이유는 모름)**

2. 어떻게 할 건지 (세부적으로)
 - a. Docker 위에 설정
 - b. Git Connector 안 씬 ⇒ 소스 코드는 **수동 커밋**으로 관리.
 - c. 그냥 퍼포스 다운 받아서 포트 등만 지정해주면 됨

[2] 서버 관련

1. **언리얼 자체 서버 사용**
 - a. 풀스택 / 클라 쪽에서 Actor 개발할 때마다 붙어서 네트워크 관리해줘야 함
 - b. 서비스에 대해서는 리슨 서버가 가장 어울리는 방식
2. **리슨 / 데디케이티드 사실 상관없다**
 - a. 데디 쓰면 클라이언트도 다 우리 중앙 서버에 연결해야 함 (게임 방 목록)
 - b. 리슨을 쓰면 호스트가 방을 만들면 API 서버로 요청보냄
 - i. 사람 드나들 때마다 요청보내고 게임 시작할 때 요청을 보내야 함 ⇒ 방 목록 관리 위함
 - ii. 클라이언트 접속은 호스트 서버에 접속
 - iii. 게임 시작 시 호스트가 서버 쪽에 API 요청 보내는 방식이 되지 않을까
 1. 처음에 훌륭한 집 에셋에 일단 다 들어가는데
 2. 이거는 사실 (리슨인 경우) 호스트가 만든 게임에 사람들이 다 접속하는 거니까
 3. **이렇게 할 때, 이것도 분리하는 게 맞지 않나...(시작 방과 스테이지 1)**
 - a. 이게 같이 있으면 어떤 버그로 사람들이 집 밖을 나갈지도 모름 (꼭 나가려는 사람이 있음)
 - iv. 게임 시작하면 방 목록에서 삭제, 보이지 않게 해야 함
 1. 🦆 강제로 끊어진 **Connection**에 대한 관리를 해줘야 함

3. AWS 사용

- a. 비용 문제가 있긴 함
- b. DB 사용할 때만 사용

4. 최적화

- a. <https://luv-n-interest.tistory.com/1345>
- b. 계산 부하 - 레이턴시 고려 예측 동기화 / 블로그에 정리 잘 되어있다고 함

4 주차 별 로드맵

[1] 3주차 (09.04 - 09.10)

- 서버팀 공통
 - 기초 강의 섹션 3까지 선별적으로 수강
 - 언리얼 C++ 멀티플레이어 마스터 : 중급 게임 개발
 - 언리얼 엔진 내에서 서버와의 통신 어떻게 해야 하는지 빠삭하게 알아야 함
 - 퍼포스 컨벤션 찾아봐야 함 ⇒ 월요일
 - 시간 정해서 서버팀만 30분 정도 찾아봐도 될 듯
 - 영서: 형상관리 ⇒ 월요일까지
 1. 퍼포스 서버 도커파일로 이미지화
 2. 우선 젠킨스 달아봄
- 서울 공통1반 A104 포팅매뉴얼.pdf
- 3. 오라클 클라우드 이슈로 이번 주는 퍼포스 사용 불가 확률 높음
 - 4. 따라서 화요일 : 퍼포스 사용법 숙지 기간 ⇒ 전체 리뷰는 영서님께서 해주실 겁니다
- 서버 아키텍처 설계
 - API 서버 설계 + 구현 30% ⇒ 주말까지 완성

- ERD ⇒ 목요일
- 호스트 방 생성 + API서버에서 관리 ⇒ 이 부분이 많이 애매해서 알아봐야 함 ⇒ 금요일까지
- **세션 연결 관리** - 방 생성, 방 목록, 유저의 세션 출입 관리
- **유저관리** - 회원가입, 로그인, 로그아웃
- **유저데이터** - 업적, 최단 클리어 시간, 클리어 횟수, 플레이 타임

[2] 4주차 (09.11 - 09.17)

- SSAFY에서 받은 EC2에 Perforce 설치
- API 서버 구현(TDD 적용)
 - **버전 확인** - 패치 및 업데이트
 - **SSE**를 이용한 게임 내 공지
 - 일반 채팅 + 보이스 채팅 구현
 - **빠른 시작** - 방 생성 시 빠른 시작 허용/비허용으로 나누어야 함 or 공개/비공개방
- ⚠ CD ⇒ 무중단 배포 ⚠ (**매우 중요**)
- 언리얼 기초 강의(12시간) 수강 완료 (서버팀 1/3 준선험시다)
 - 오전(~11시) : 기초강의 수강
 - 점심 이후 15분 : 기초 강의 간단하게 리뷰
- 클라 지원(맵, UI 등)

[3] 5주차 (09.18 - 09.24)

- MVP 1차 배포
- 클라이언트 추가 개발
- 프레임드랍이 생기거나, 문제 발생 시 ⇒ 테스트 (레이턴시, GPU 부하, etc)
- 공식 사이트 개발
 - 여기는 게임 사이트 레퍼런스 구해서 디자인은 따라하고 기능은 공통 때 썼던 코드 재활용.
 - 만약 만들면 위 API 서버에서 사용하는 DB와의 연동도 필요 (회원, 기록 등)
 - 게임 소개 시 사용

- 클라 지원(맵, UI 등)

[4] 6주차 (09.25 - 10.01)

- 추석
- 클라 지원(맵, UI 등)
- (IOCP 서버 라이트하게 구축해보기, 로우레벨로 뜯어보기)
- (소켓, 멀티쓰레드, 네트워크 프로그래밍 스터디)

[5] 7주차 (10.02 - 10.08)

- 디버깅, UCC, 그 외에 바쁨