프로젝트 구성

멀티플레이 중계 서버

웹소켓 연결
클라이언트 요청 처리

세션, 대기열 정보 저장
메시지 브로커 역할

웹, DB 서버

Express.js
DB 갱신, 조회 API 처리
사용자 인증 API

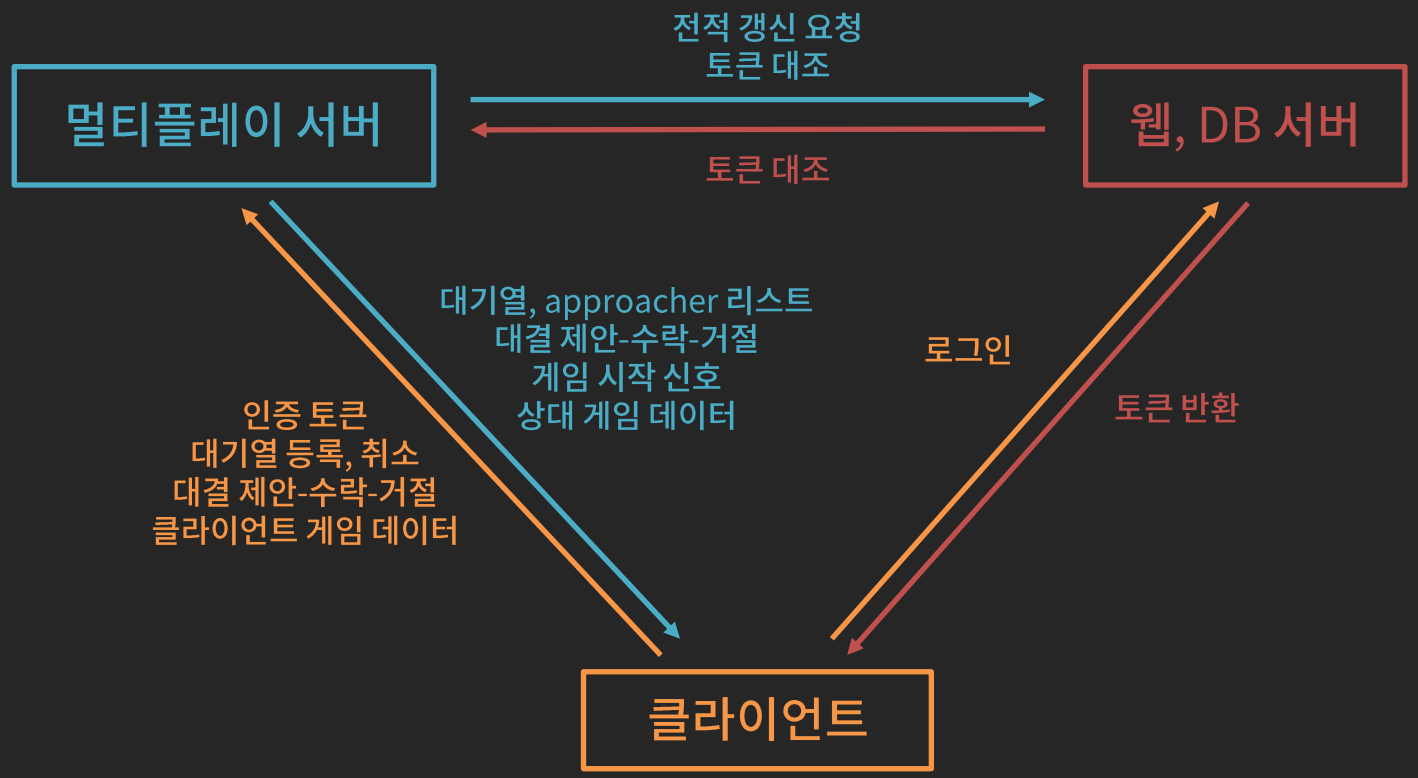
회원 DB
전적 DB

클라이언트

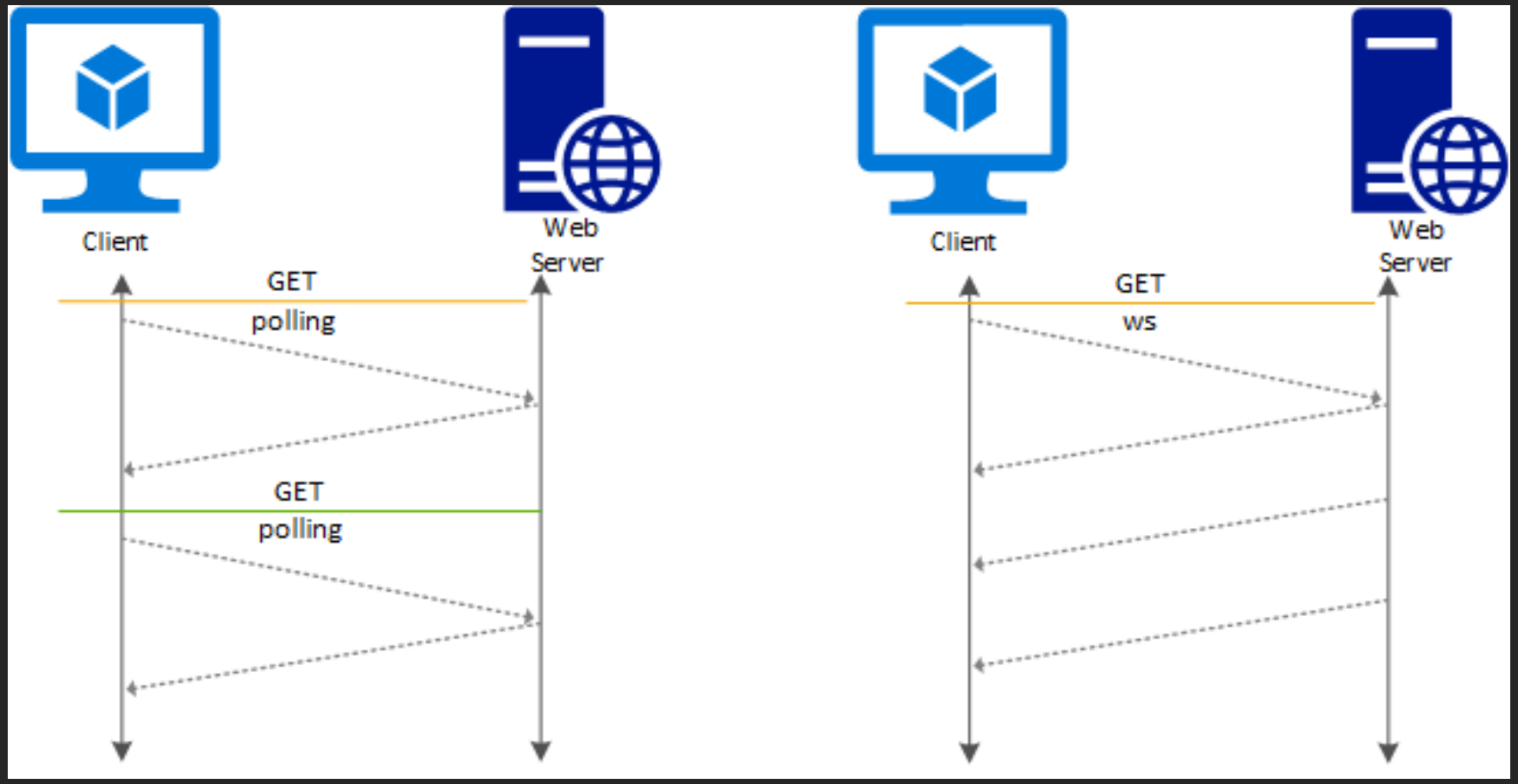
그래픽, 사운드 출력
이벤트 루프 …
등등

런처, 온라인 로비
GUI 구성

프로젝트 구성


멀티플레이 서버
웹, DB 서버
클라이언트
전적 갱신 요청
토큰 대조
토큰 대조
대기열, approacher 리스트
대결 제안-수락-거절
게임 시작 신호
상대 게임 데이터
인증 토큰
대기열 등록, 취소
대결 제안-수락-거절
클라이언트 게임 데이터
로그인
토큰 반환

웹 소켓 연결

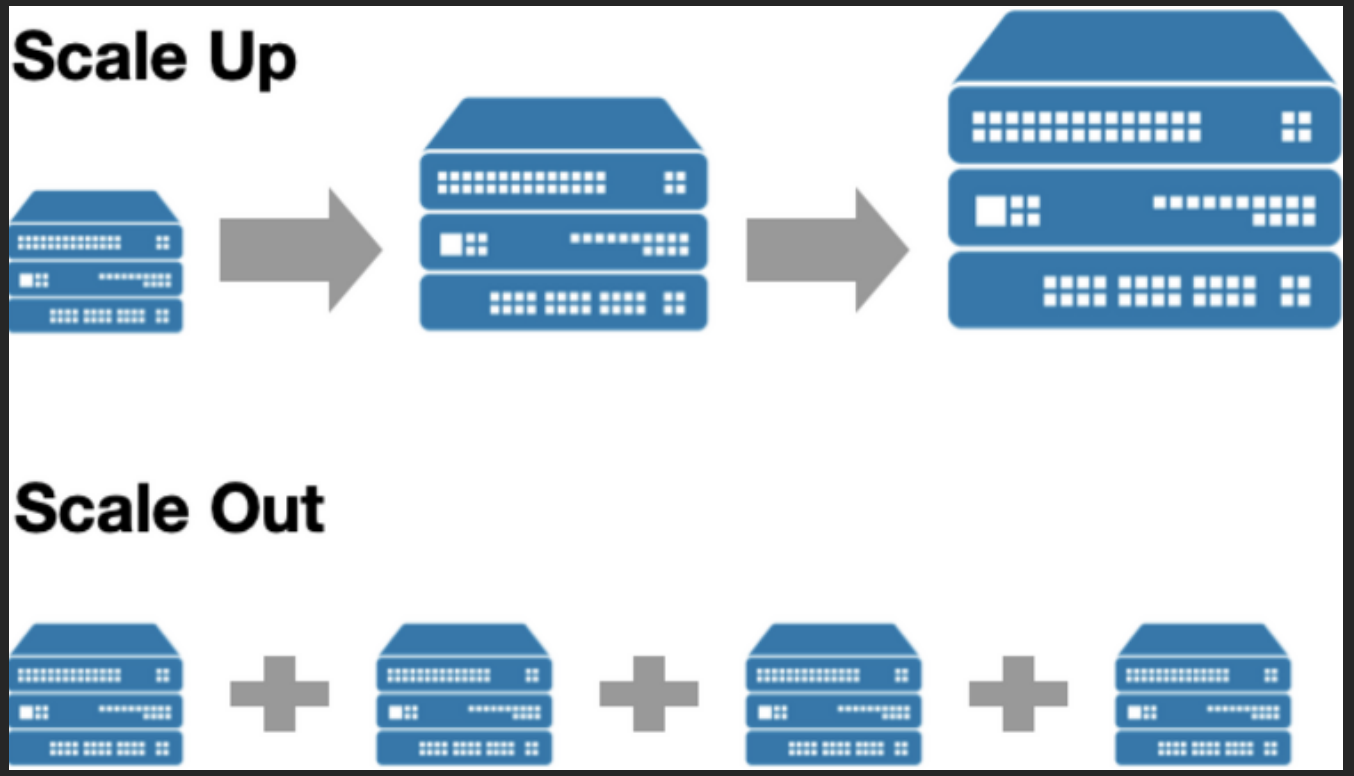


일반적인 http 요청

웹소켓

- 보다 오버헤드가 적고 지연시간이 짧음
- 클라이언트와의 연결이 지속되는 특성상, Scale-Out을 위해서 추가적인 절차가 필요함.

Scale Out



보다 유연하고 비용이 적게 드는 Scale Out

중계 서버 구성

서버 프로세스

- FastAPI – Uvicorn 구성
- 다수 존재 가능
- 수평 확장(Scale Out) 가능
- 메시지 브로커를 통해 다른 서버 프로세스에 데이터 전송 명령

In-memory DB
Message Broker

- 다수의 서버 프로세스가 Redis에서 데이터를 가져오는 구조이기 때문에 서버 프로세스의 수평 확장이 가능함.
- Redis도 Scale-Out을 지원하지만, 본 프로젝트에선 Redis 클러스터 까지는 고려하지 않음.

프로세스 매니저

- 프로세스 매니저가 복수의 서버 프로세스를 pre-fork 방식으로 실행함.
- 본 프로젝트에선 gunicorn을 사용하였으나, 쿠버네티스나 Docker Swarm같은 컨테이너 오케스트레이션 시스템으로도 대체 가능

웹, 리버스 프록시 서버

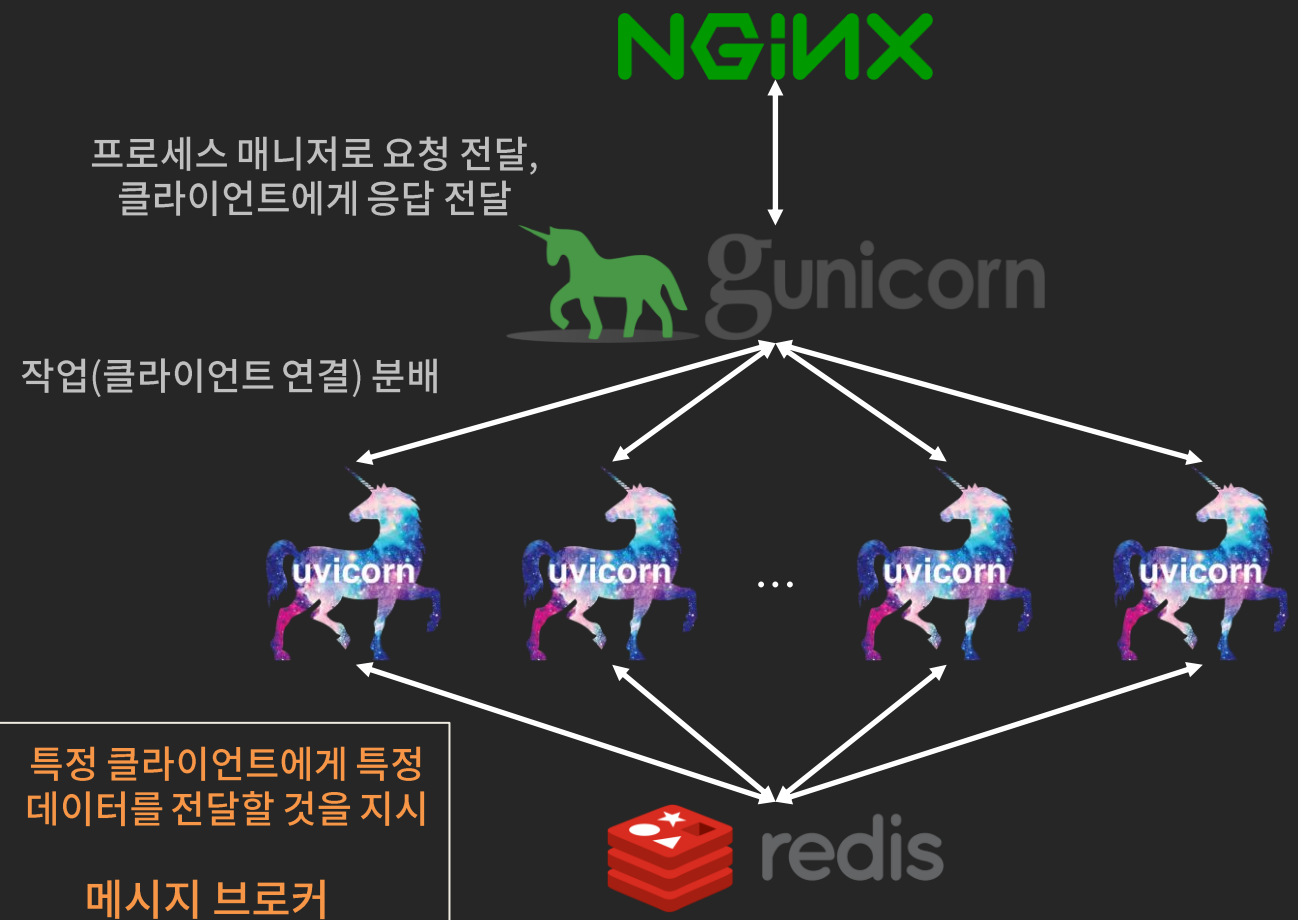
- 특정 주소로 온 요청을 알맞은 내부 주소로 전달함.
- SSL 암호화

Scale Out

리버스 프록시

프로세스 매니저

서버 프로세스



Redis
In-memory DB
메시지 브로커

클라이언트에게 전송할 데이터 GET	특정 클라이언트에게 특정 데이터를 전달할 것을 지시
In-memory DB	메시지 브로커

서버 구성 상황

Amazon Lightsail

Ubuntu 20.04 인스턴스

docker

uvicorn FastAPI gunicorn

중계 서버 프로세스 + 프로세스 매니저 컨테이너

redis

Redis + RedisJSON 모듈 컨테이너

node.js

DB API, WAS node.js 컨테이너

NGINX

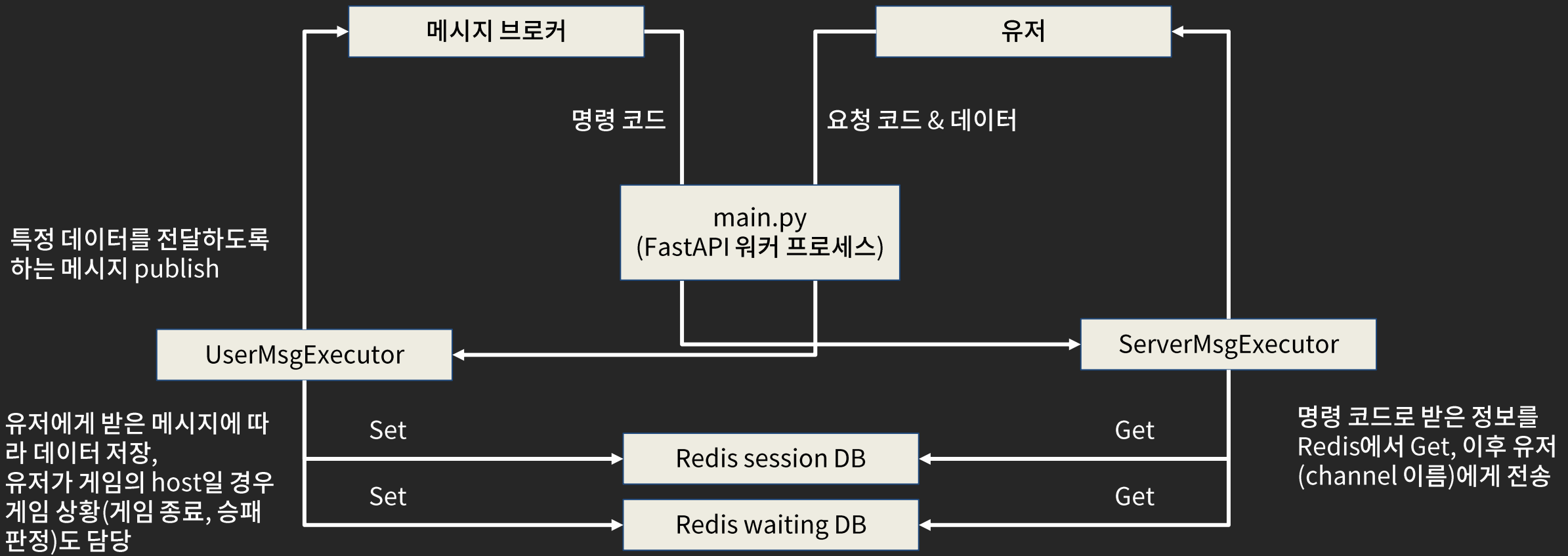
리버스 프록시 서버

Amazon RDS

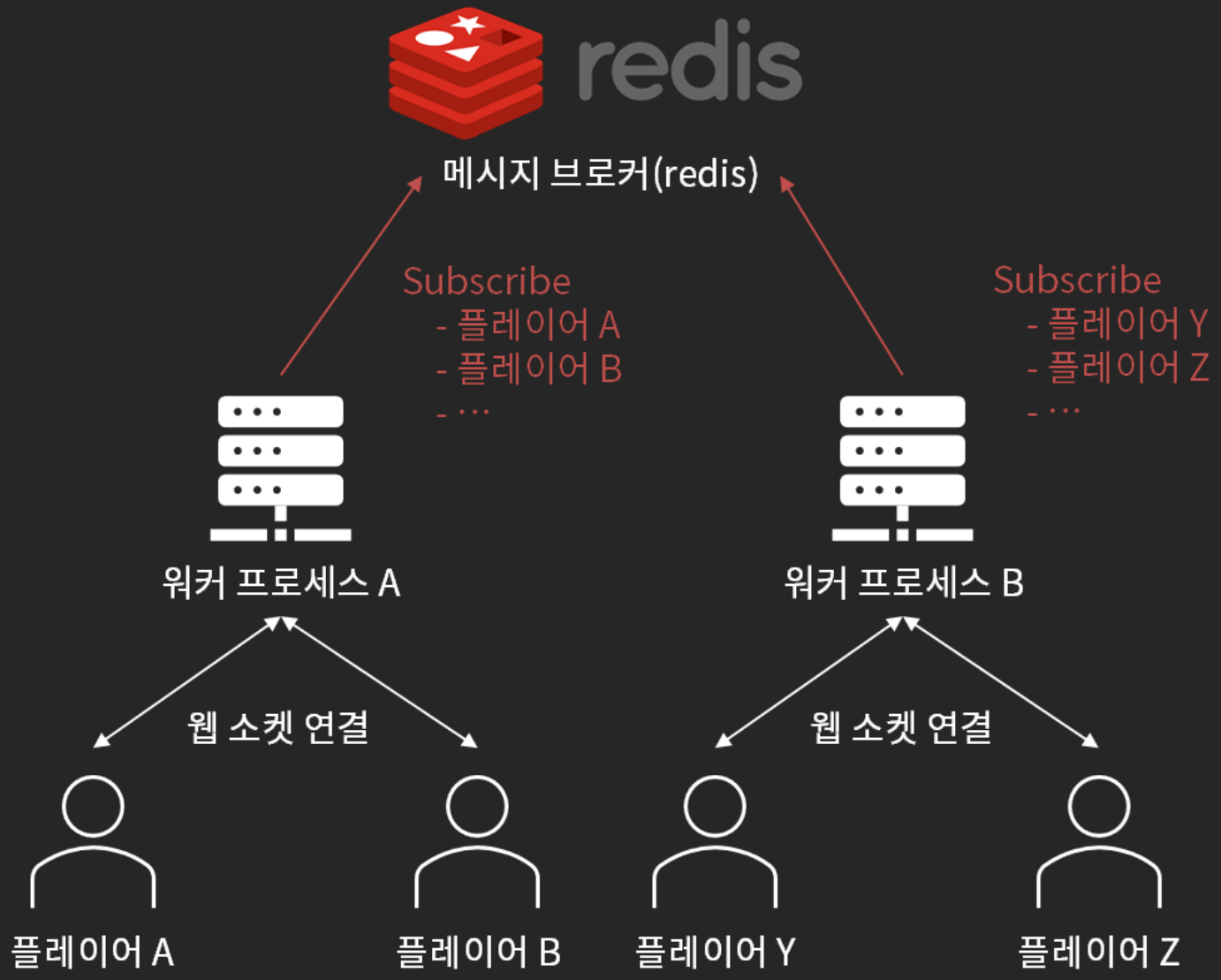
MySQL

MySQL RDS 인스턴스

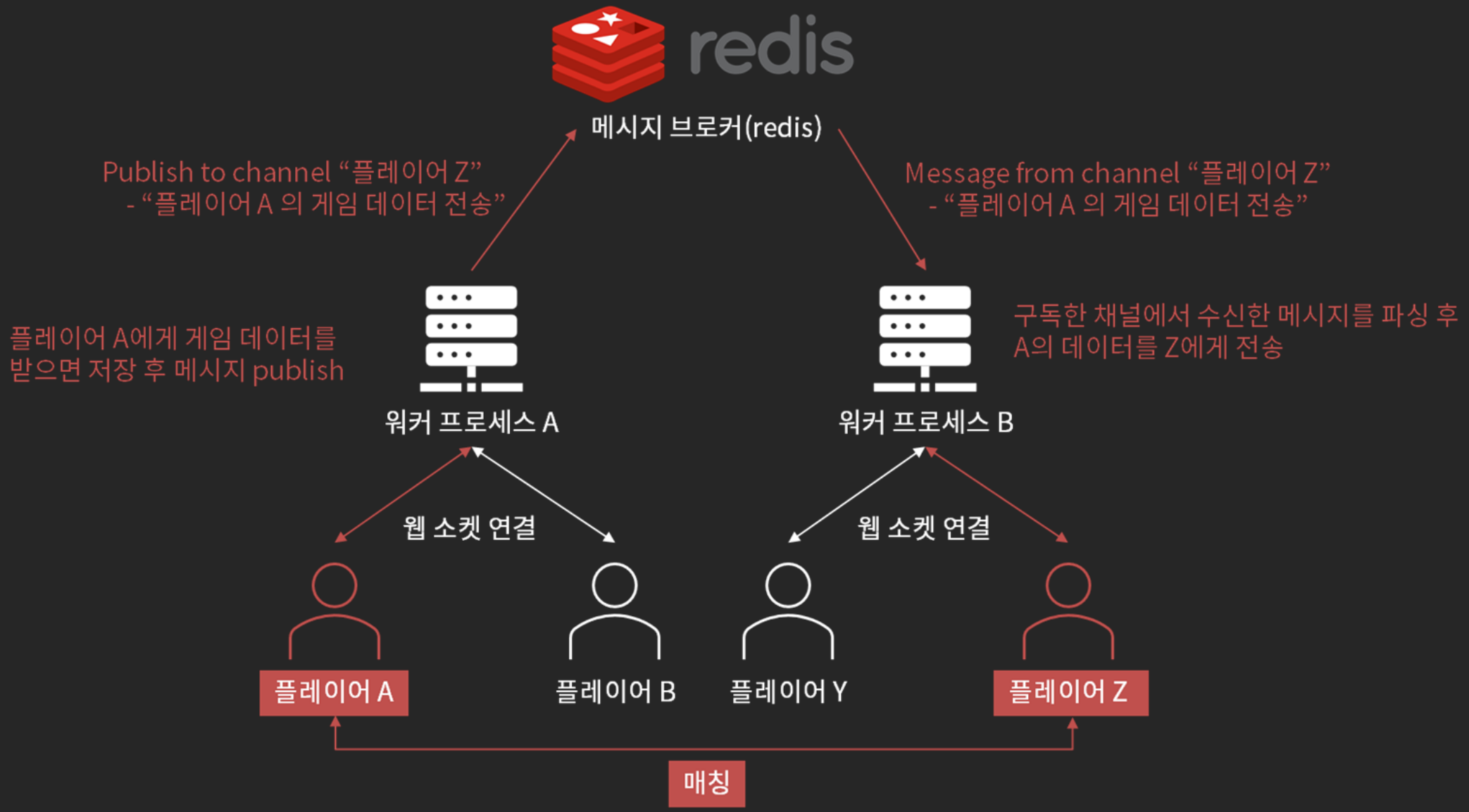
중계 서버 세부 구성


메시지 브로커
유저
명령 코드
요청 코드 & 데이터
main.py
(FastAPI 워커 프로세스)
특정 데이터를 전달하도록
하는 메시지 publish
UserMsgExecutor
ServerMsgExecutor
유저에게 받은 메시지에 따
라 데이터 저장,
유저가 게임의 host일 경우
게임 상황(게임 종료, 승패
판정)도 담당
Set
Set
Redis session DB
Redis waiting DB
Get
Get
명령 코드로 받은 정보를
Redis에서 Get, 이후 유저
(channel 이름)에게 전송

메시지 브로커 구조


redis
메시지 브로커(redis)
Subscribe
- 플레이어 A
- 플레이어 B
- …
Subscribe
- 플레이어 Y
- 플레이어 Z
- …
워커 프로세스 A
워커 프로세스 B
웹 소켓 연결
웹 소켓 연결
플레이어 A
플레이어 B
플레이어 Y
플레이어 Z

메시지 브로커 구조


redis
메시지 브로커(redis)
Publish to channel “플레이어 Z”
- “플레이어 A 의 게임 데이터 전송”
Message from channel “플레이어 Z”
- “플레이어 A 의 게임 데이터 전송”
플레이어 A에게 게임 데이터를
받으면 저장 후 메시지 publish
구독한 채널에서 수신한 메시지를 파싱 후
A의 데이터를 Z에게 전송
워커 프로세스 A
워커 프로세스 B
웹 소켓 연결
웹 소켓 연결
플레이어 A
플레이어 B
플레이어 Y
플레이어 Z
매칭