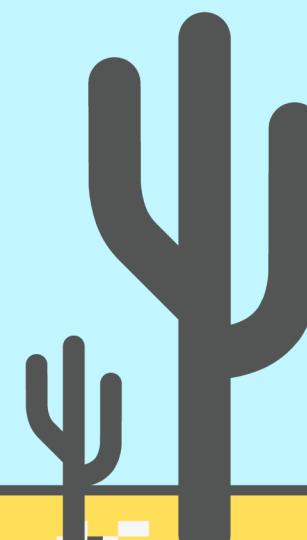
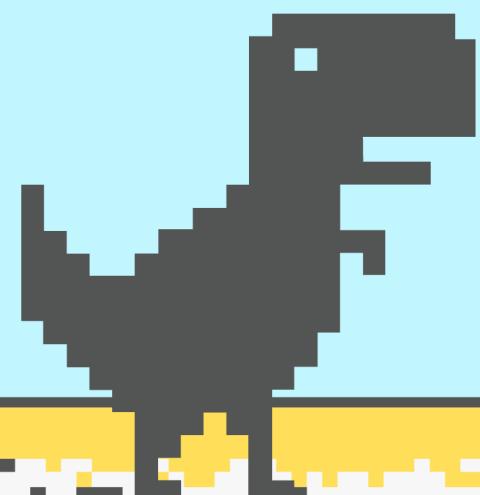


CHEAT KEY

START



INDEX

1. 프로젝트 개요

2. 주요 요구 사항

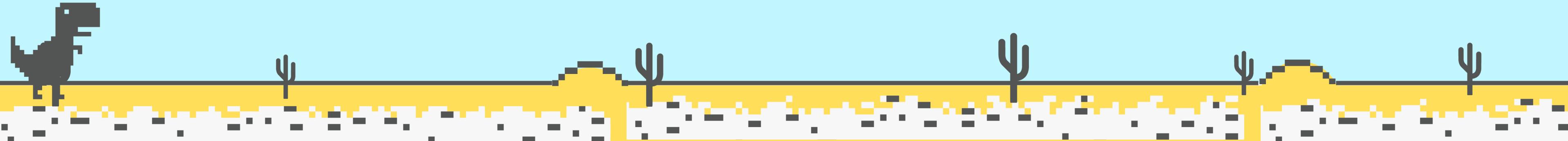
3. 서비스 구성

4. 개발 & 서비스 시연

5. 회고

6. 느낀 점

7. Q&A

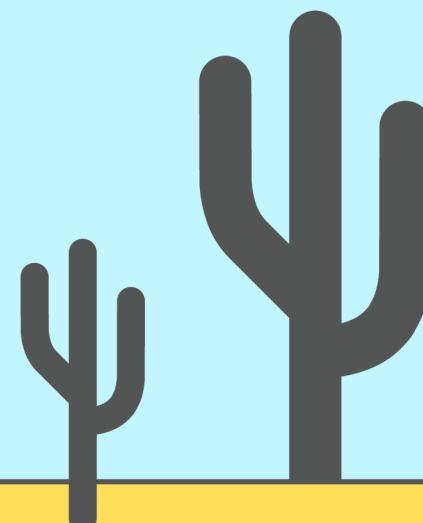


1. 프로젝트 개요 - 주제



AZURE KUBERNETES

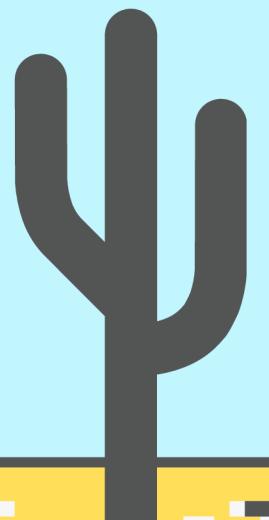
기반 실시간 테트리스 매칭 플랫폼 구축



1. 프로젝트 개요 - 목표

단순한 게임 개발을 넘어

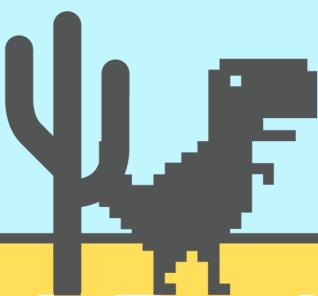
클라우드 기술(AKS, CI/CD)을 활용한
안정적이고 확장 가능한 게임 서비스구축



1. 프로젝트 개요 - 서비스 내용

실시간으로 진행가능한
1 VS 1 테트리스 게임

게임 결과를 저장하고
개인의 실력에 따른 매치메이킹



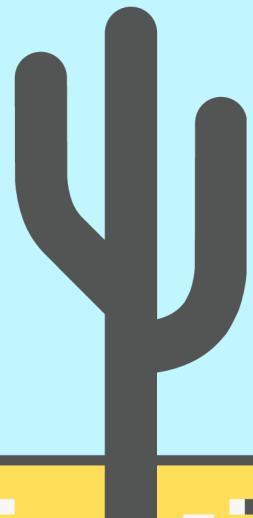
1. 프로젝트 개요 - 팀원 (R & R)

유하민(팀장): API 개발, DB 연동, 가상화 구현

이지후: API 개발, DB 연동, 발표자료 제작

박상우: FRONT-END 템플릿 구성, 테트리스 게임 설계

박한비: AZURE 인프라 구성, CI/CD(쿠버네티스 배포)



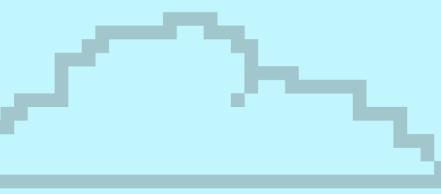
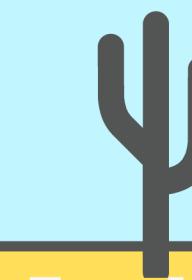
2. 주요 요구 사항 - 세이스

A. 회원가입/로그인

B. 매칭시스템

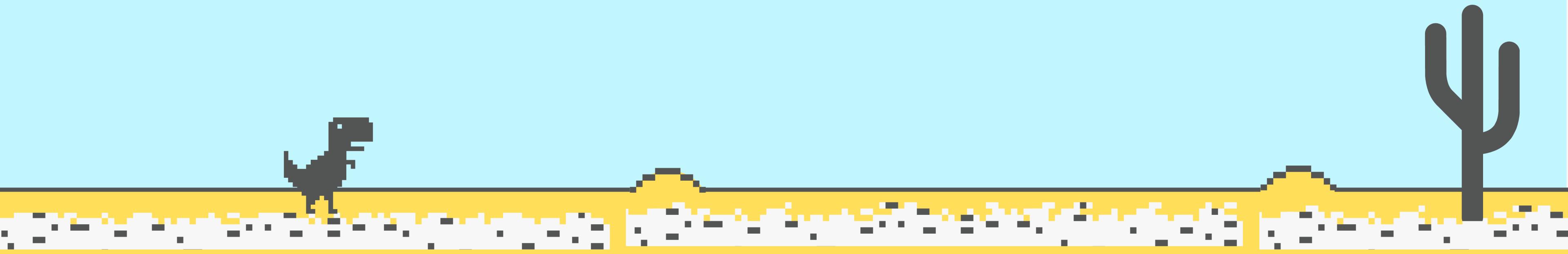
C. 실시간 매칭/결과기록

D. 가상화

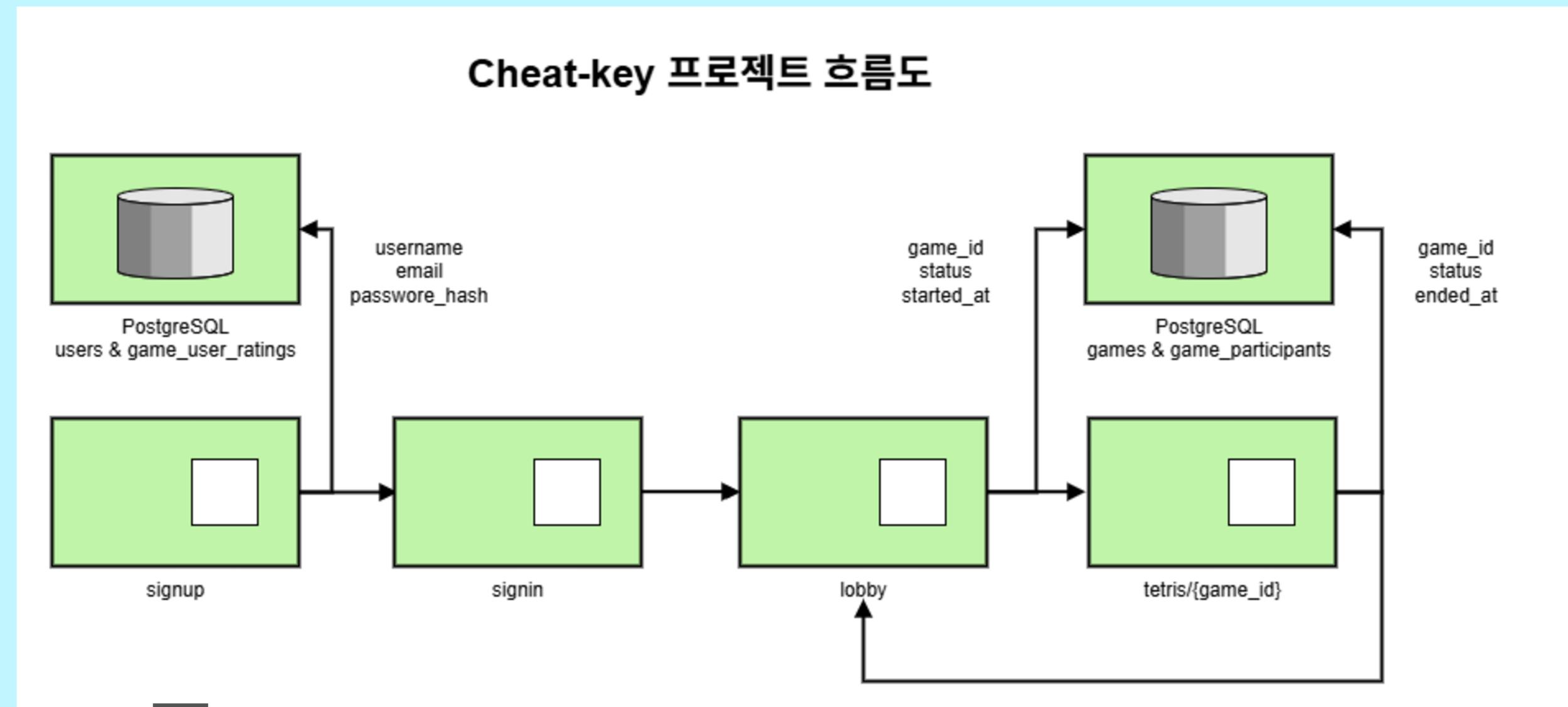


2. 주요 요구 사항 - 기술

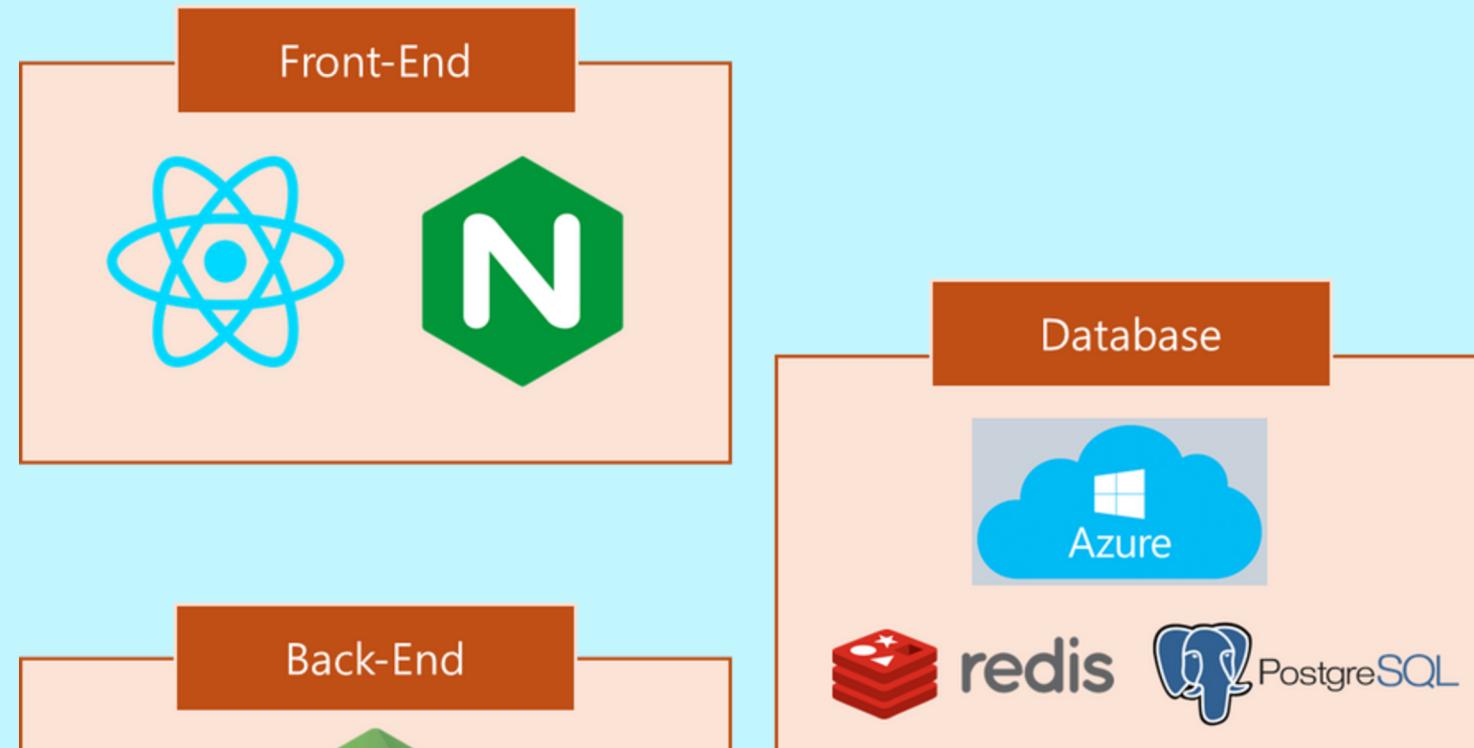
1. 안정적인 실시간 통신 [품질]
2. 사용자 데이터의 안전한 저장 [보안]
3. 트래픽 증가에 대한 대처 [확장성]



3. 서비스 구성 - 흐름도



3. 서비스 구성 - 시스템



1. FRONT-END

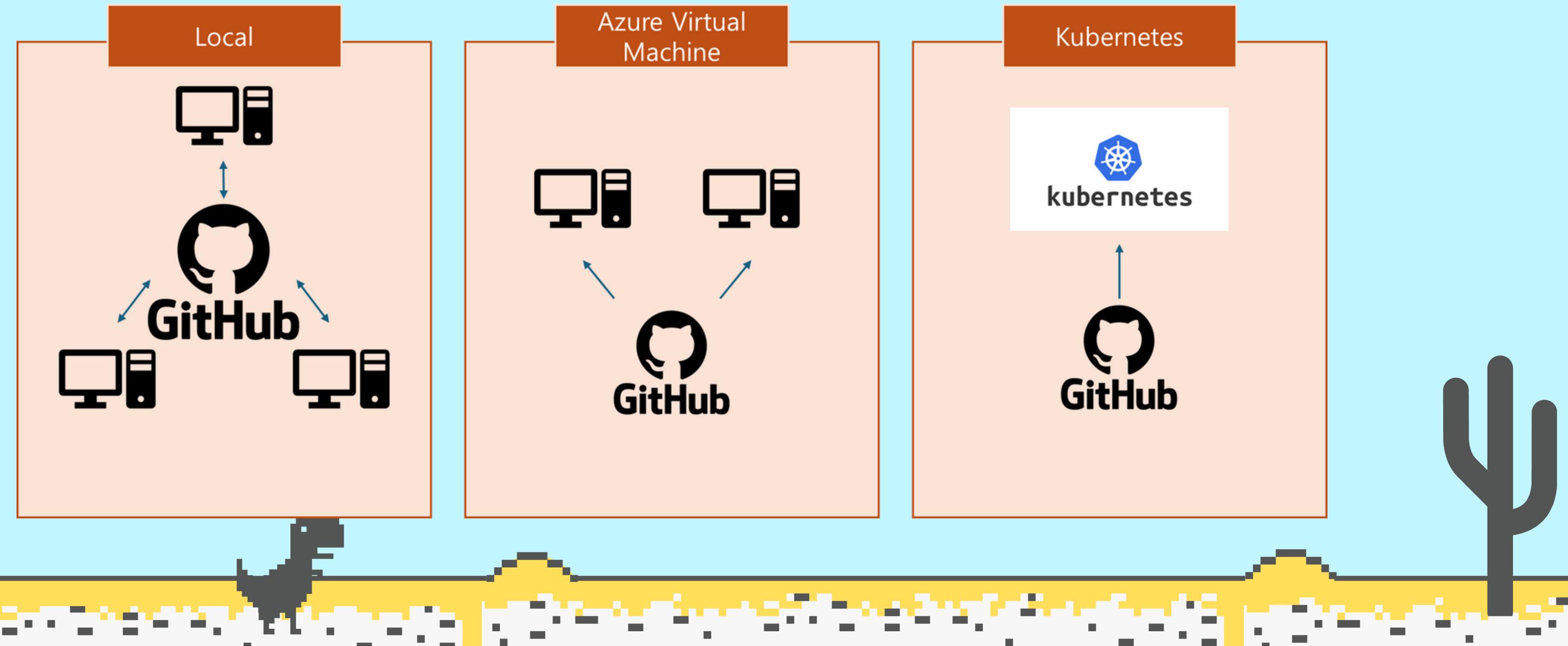
- API:** 서버 통신 담당
- ASSETS:** 이미지, 사운드, 폰트 등 정적 파일
- COMPONENTS:** 재사용 가능한 작은 UI 조각들
- CONTEXT:** 전역 상태 관리
- PAGES:** 실제 화면을 구성하는 페이지들

2. BACK-END

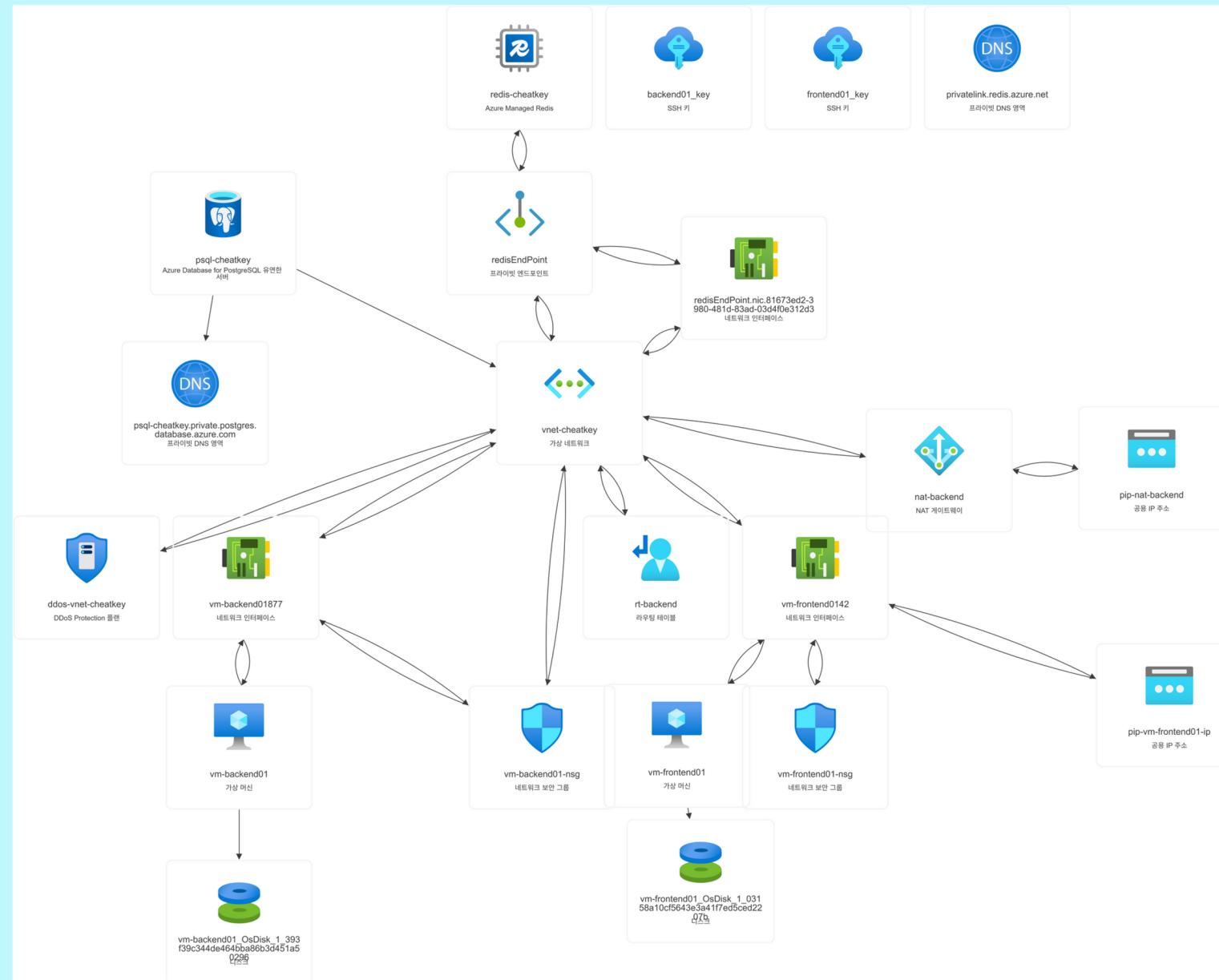
- API:** 엔드포인트 (요청의 첫 관문)
- CONFIG:** 데이터베이스 등 외부 서비스 연결 설정
- MIDDLEWARE:** 요청 중간 처리 (보안, 로깅 등)
- MIGRATIONS:** 데이터베이스 스키마 관리 (DB 설계도)
- SERVICES:** 핵심 비즈니스 로직 처리



3. 서비스 구성 - 인프라

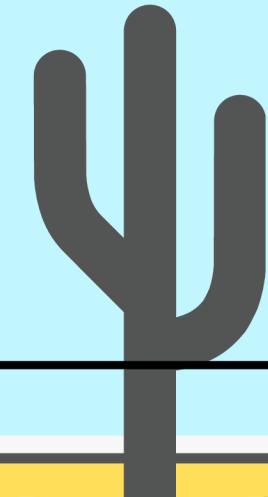


3. 서비스 구성 - AZURE 리소스

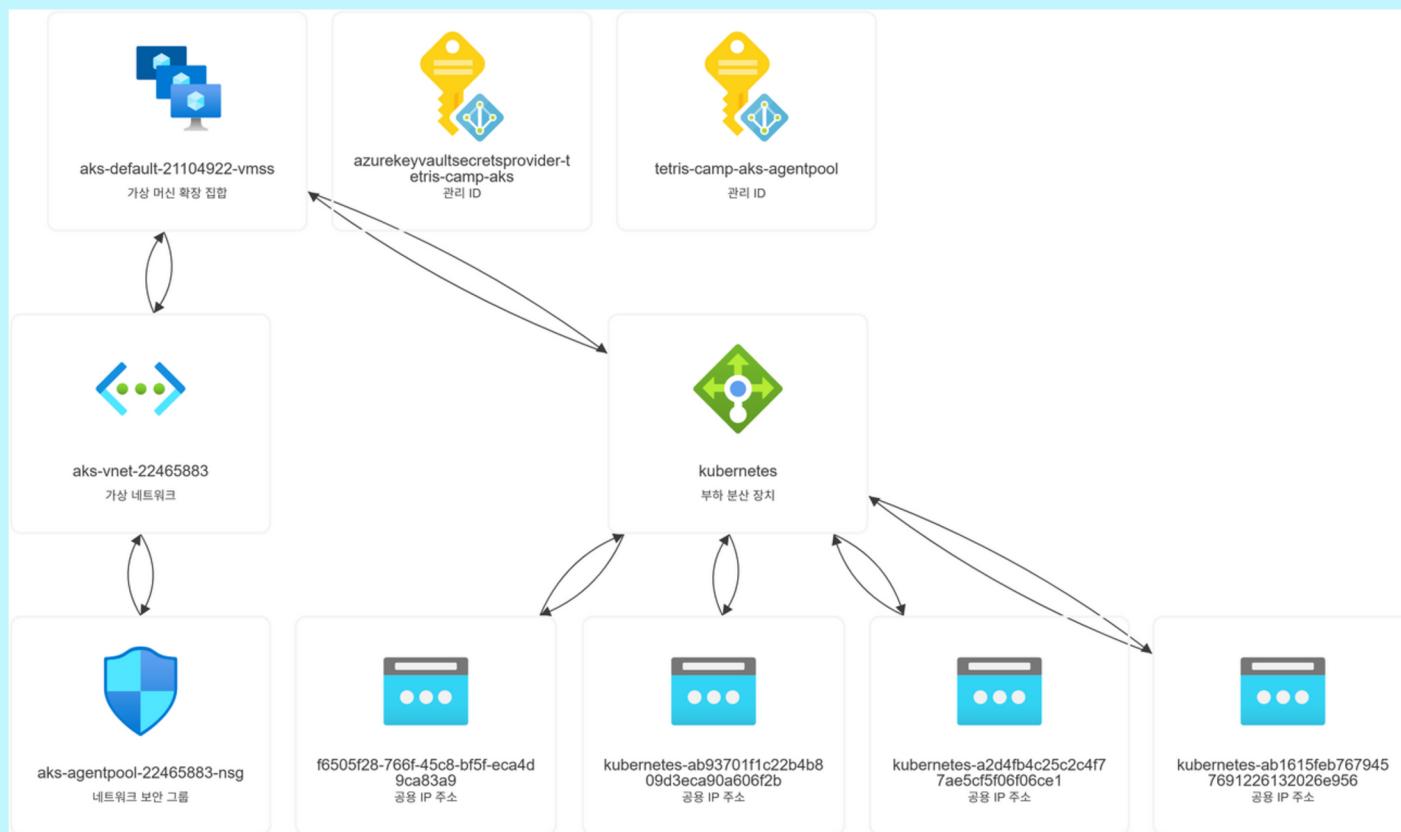


1. 단일 가상 머신

- 리눅스 가상 머신 (FE | BE)
- NAT 게이트 웨이, 라우팅 테이블
- AZURE POSTGRESQL 서버
- AZURE REDIS FOR CACHE



3. 서비스 구성 - AZURE 리소스



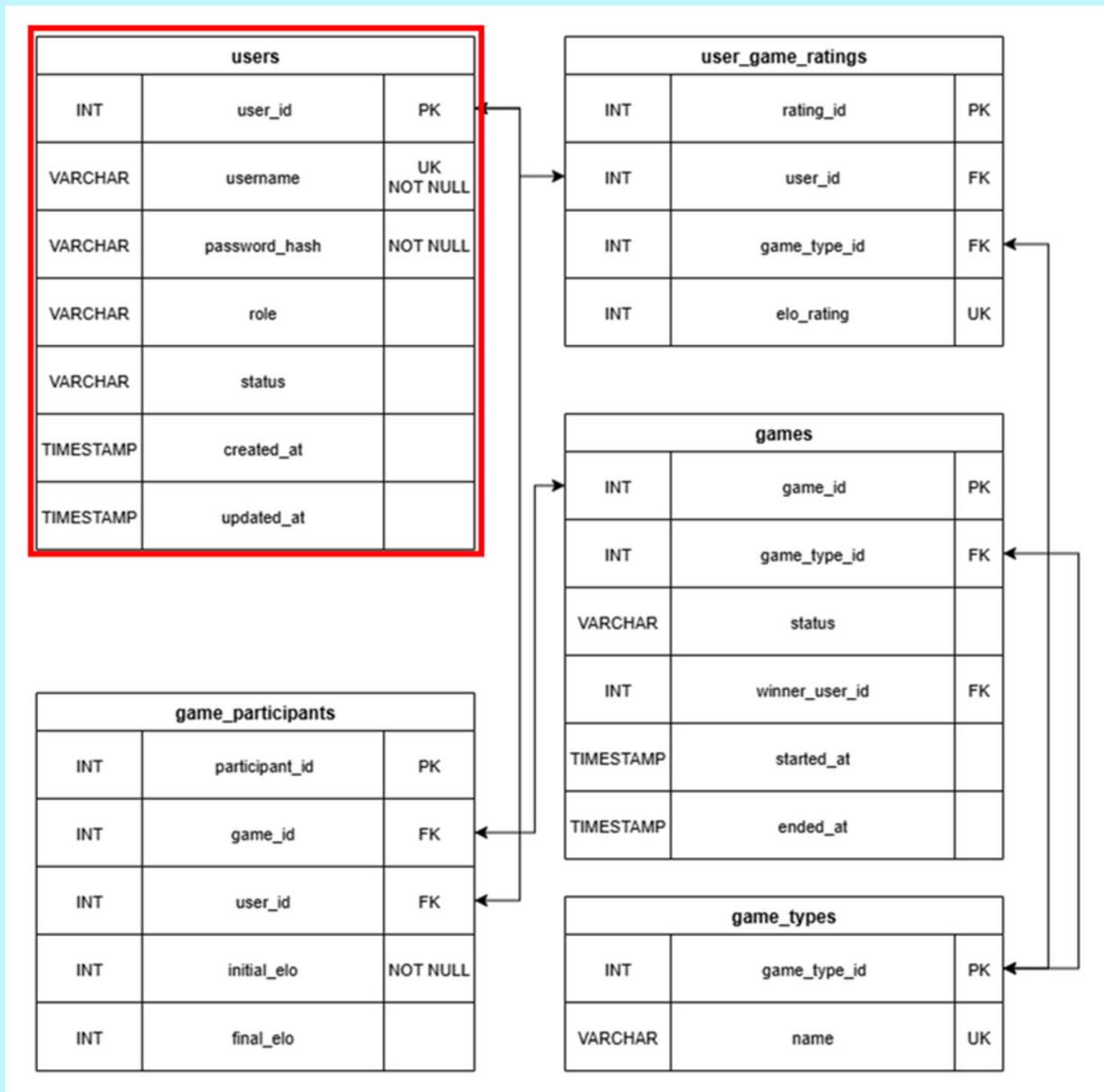
2. 쿠버네티스 CI/CD 사용

- ACR(AZURE CONTAINER REGISTRY)
- AKS.TF – AKS (AZURE KUBERNETES)
- AZURE POSTGRESQL 서버
- AZURE REDIS FOR CACHE
- AZURE KEY VAULT
- CI/CD용 관리 ID

- AKS 생성 시, 클러스터 구동에 필수적인 핵심 기반 시설을 자동으로 생성



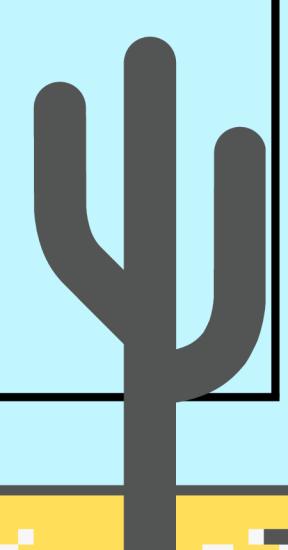
3. 서비스 구성 - 게임 서비스



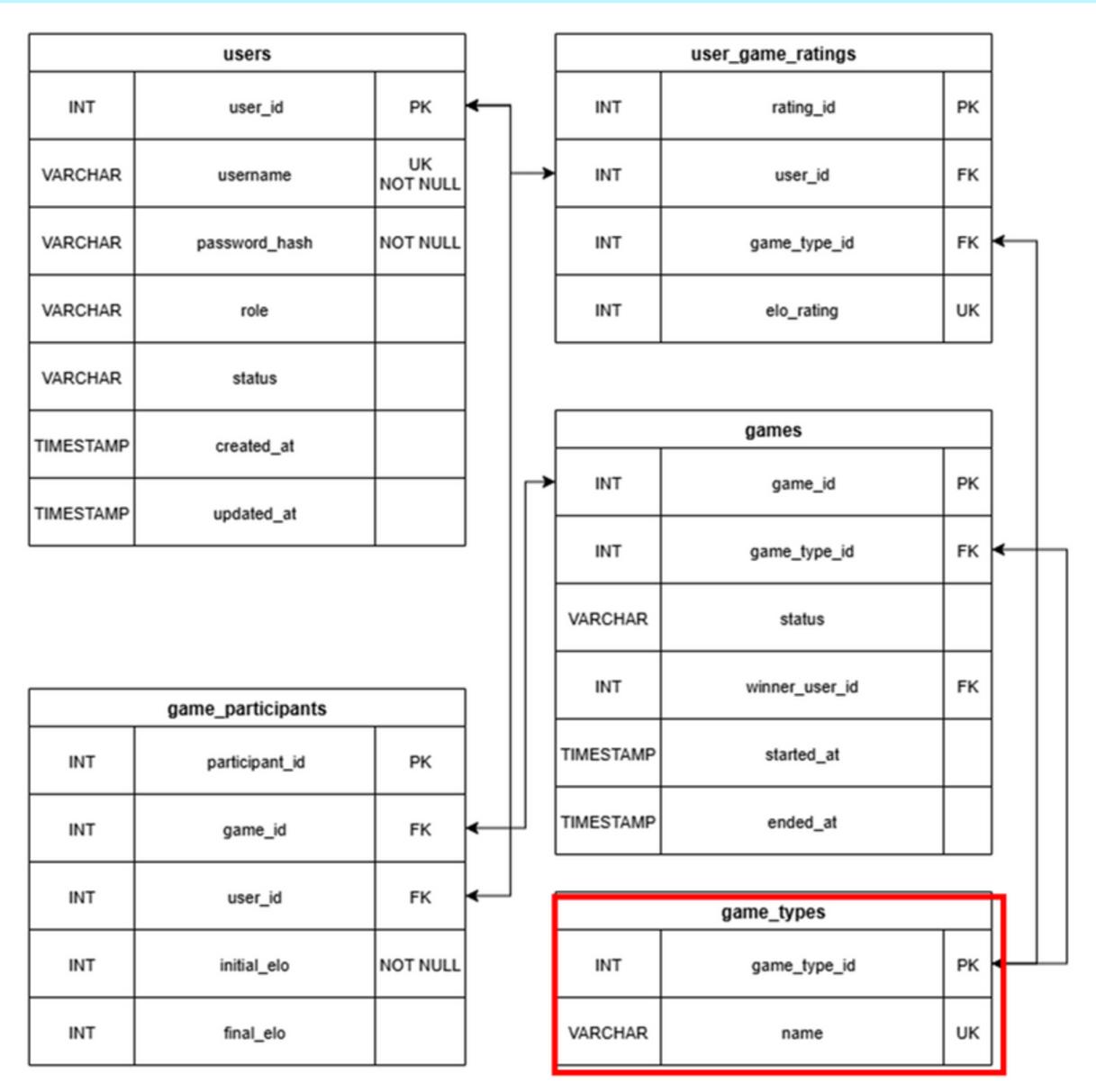
1. USERS

사용자의 계정 정보를 관리합니다.

- USER_ID**: 유저 고유 번호 (PK)
- USERNAME**: 아이디 (UNIQUE, NOTNULL)
- PASSWORD_HASH**: BCRYPT 해시화 된 비밀번호
- ROLE**: 유저의 역할 (유저 | 관리자)
- STATUS**: 계정의 상태 (활성 | 비활성)
- CREATED_AT**: 생성 시간
- UPDATED_AT**: 변경 시간



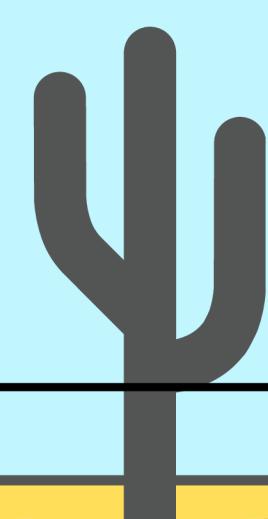
3. 서비스 구성 - 데이터베이스



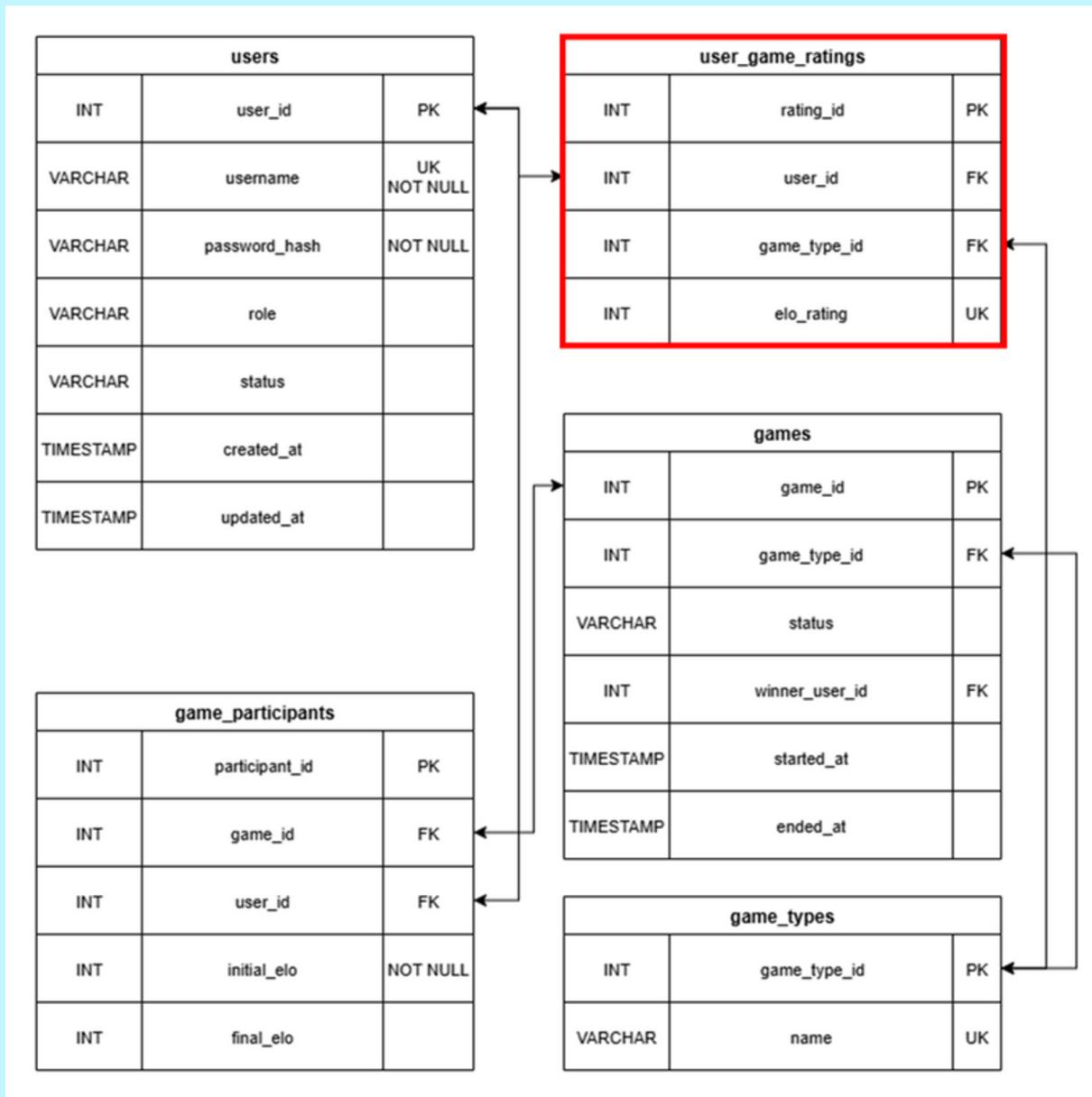
2. GAME_TYPES

제공하는 게임의 종류를 정의합니다

- GAME_TYPE_ID**: 게임 고유 번호 (PK)
- NAME**: 게임 이름



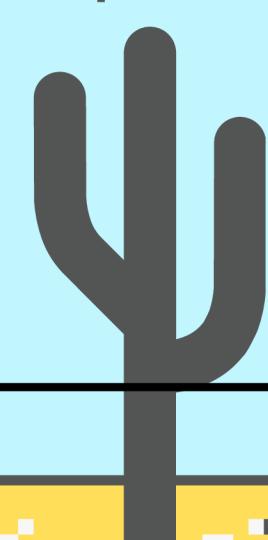
3. 서비스 구성 - RATING 서비스



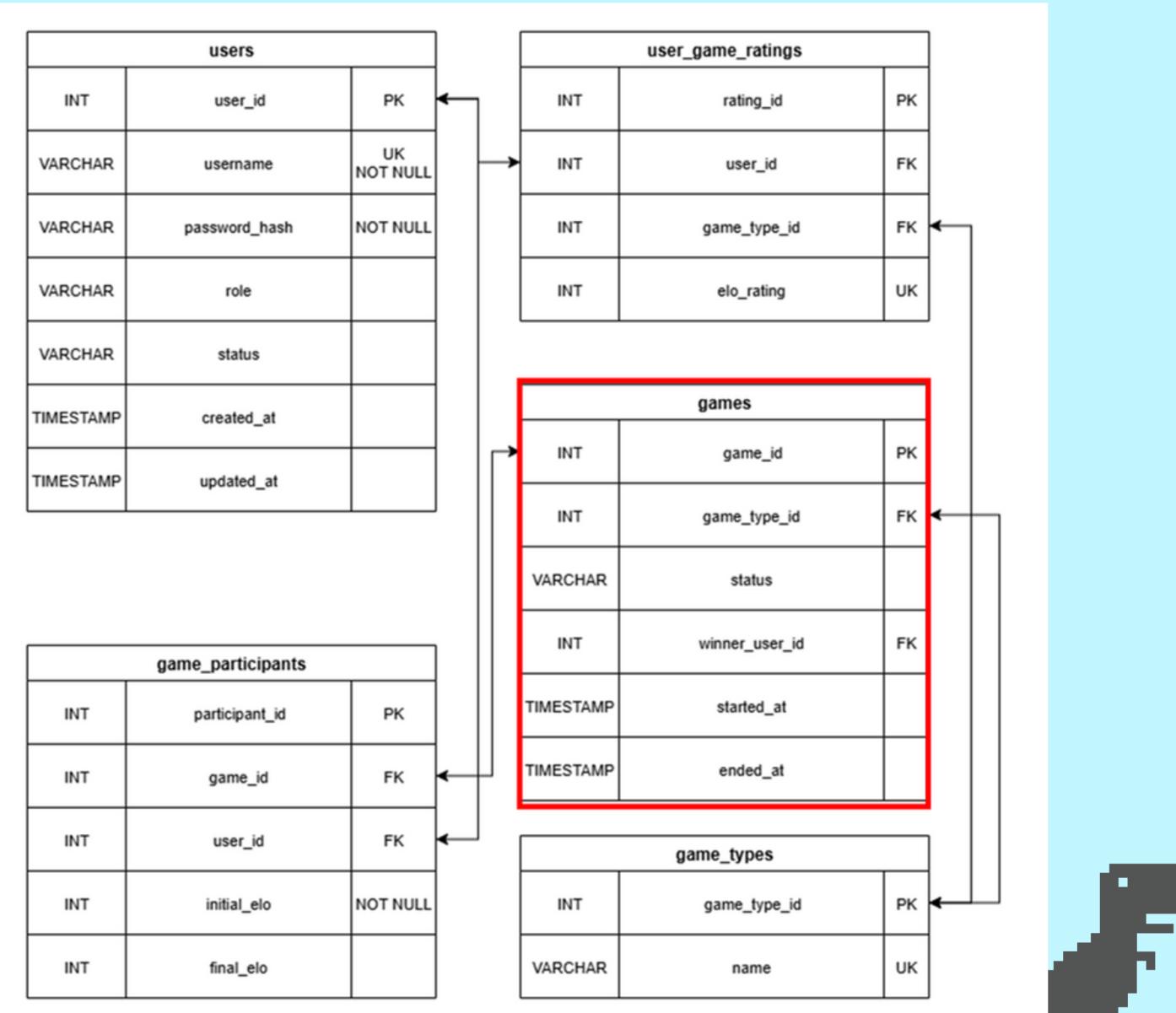
3. RATINGS

게임 별 ELO 점수를 관리합니다.

- RATING_ID**: 레이팅 기록 고유 번호 (PK)
- USER_ID**: 사용자 고유 번호 (FK)
- GAME_TYPE_ID**: 게임 고유 번호 (FK)
- ELO_RATING**: 게임 별 사용자 ELO 점수 (UNIQUE)



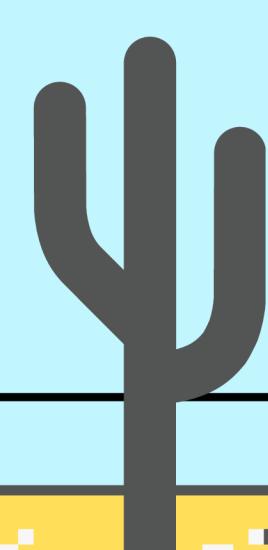
3. 서비스 구성 - 게임 서비스



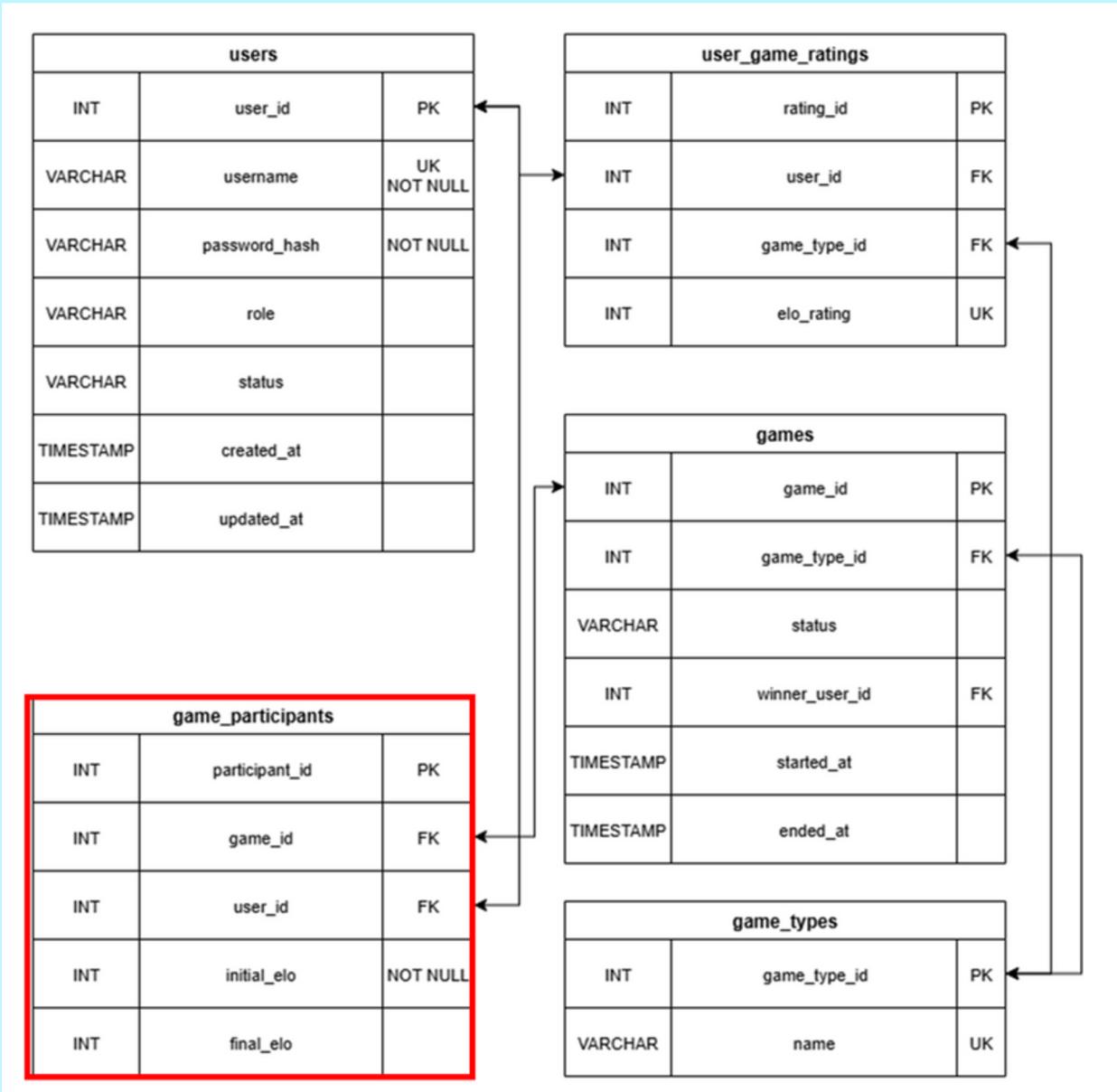
4. GAMES

게임 세션 정보를 기록합니다.

- **GAME_ID**: 게임 세션 고유 번호 (PK)
- **GAME_TYPE_ID**: 게임 고유 번호 (FK)
- **STATUS**: 게임 상태 (IN_PROGRESS | COMPLETED)
- **WINNER_USER_ID**: 승리한 사용자 고유 번호
- **STARTED_AT**: 게임 시작 시각
- **ENDED_AT**: 게임 종료 시각



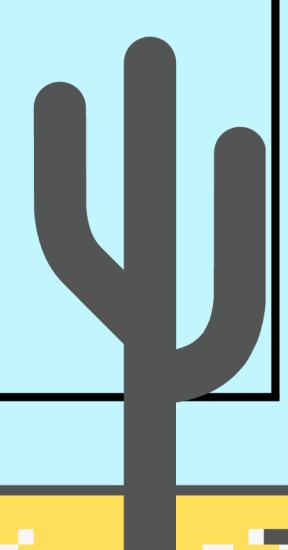
3. 서비스 구성 - 게임 서비스



5. PARTICIPANTS

게임 세션에 참여한 정보를 기록합니다.

- PARTICIPANT_ID**: 참여 기록 고유 번호 (PK)
- GAME_ID**: 게임 세션 고유 번호 (FK)
- USER_ID**: 참여한 사용자 고유 번호 (FK)
- INITIAL_ELO**: 게임 시작 시점의 ELO 점수
- FINAL_ELO**: 게임 종료 후 변동된 ELO 점수



4. 개발 & 서비스 시연 - FE / BE

회원가입

닉네임

이메일

비밀번호 (8자 이상)

비밀번호 확인

가입하기

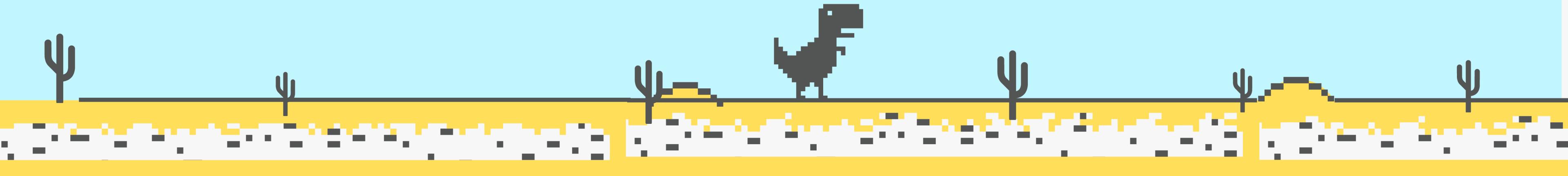
메인 화면

이미 계정이 있으신가요? [로그인](#)

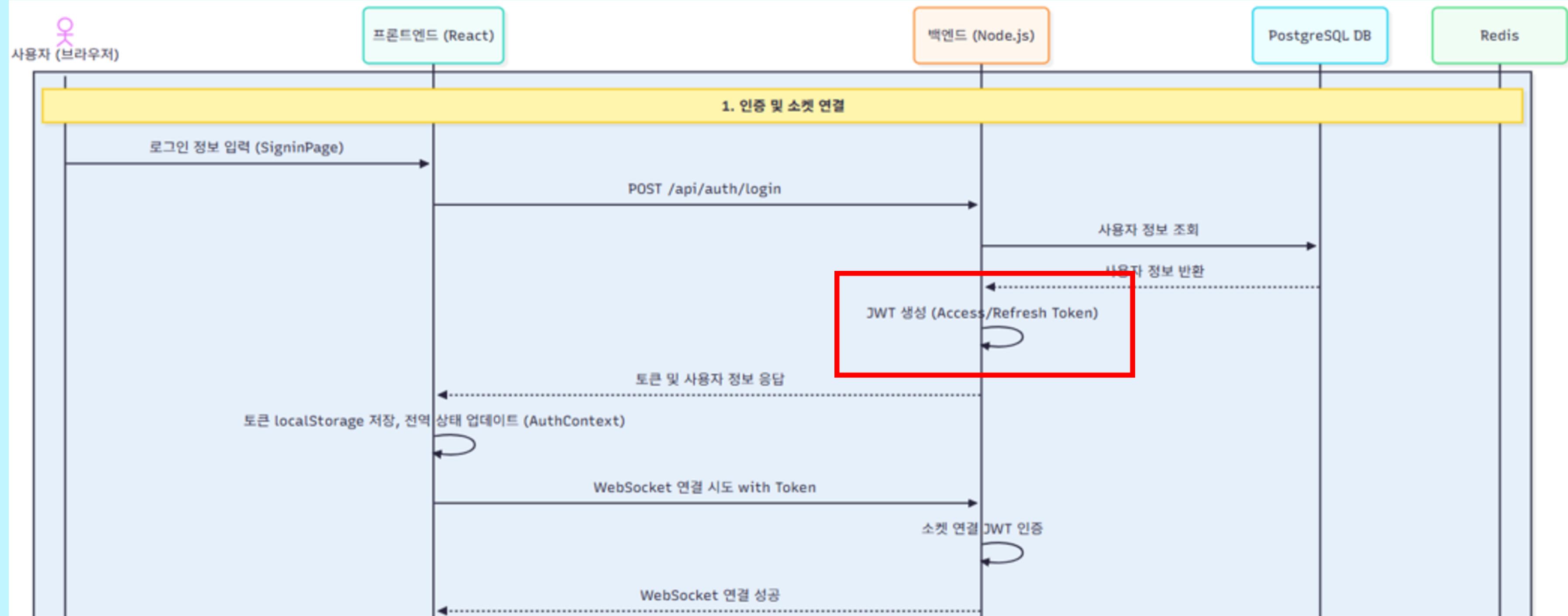
1. ✓SIGNUP

사용자 회원가입 페이지

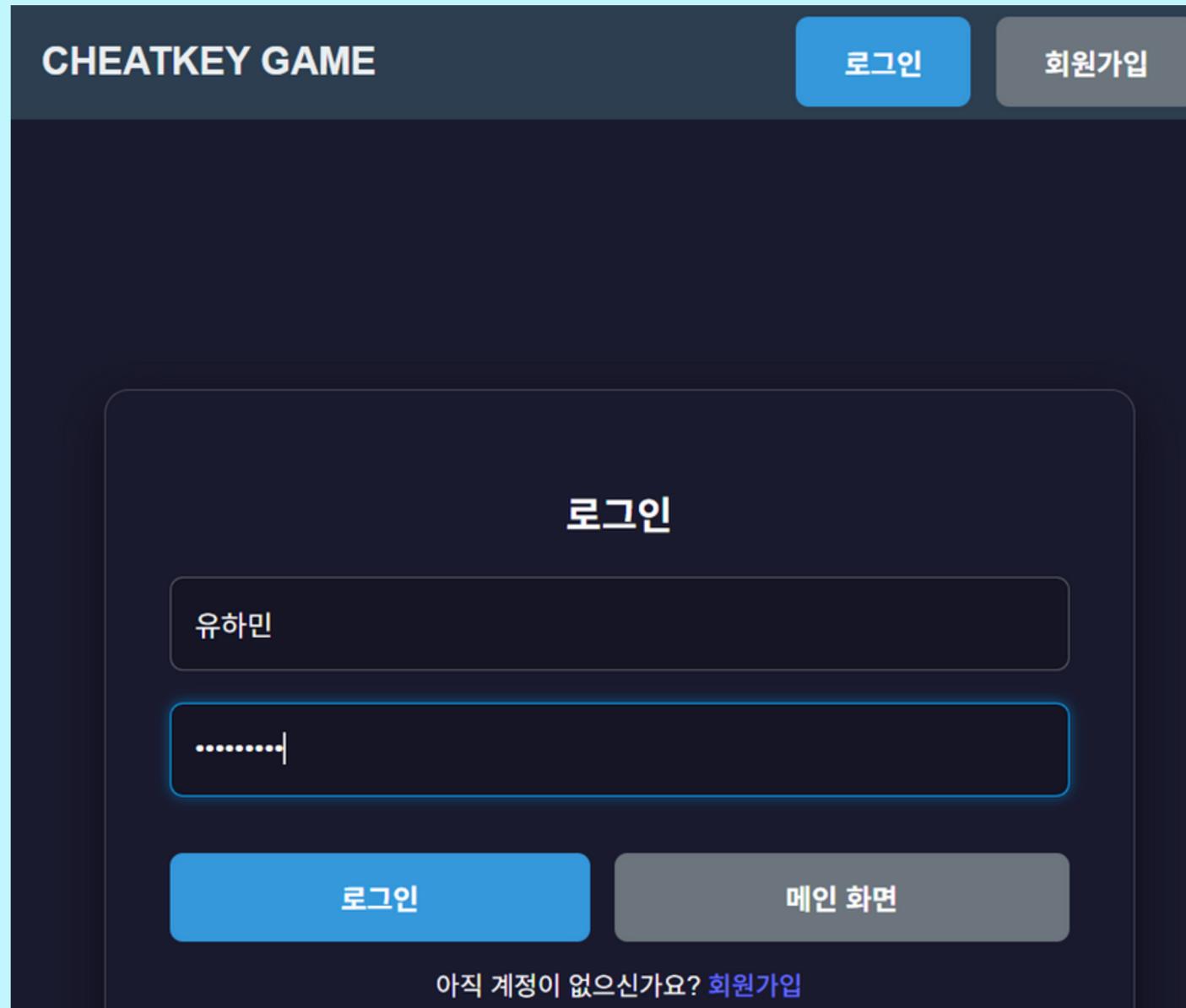
- 입력
닉네임, 이메일, 비밀번호
- 저장
고유번호, 아이디, 이메일, 비밀번호,
- 기능
비밀 번호 확인 기능



4. 개발 & 서비스 시연 - FE / BE



4. 개발 & 서비스 시연 - FE / BE



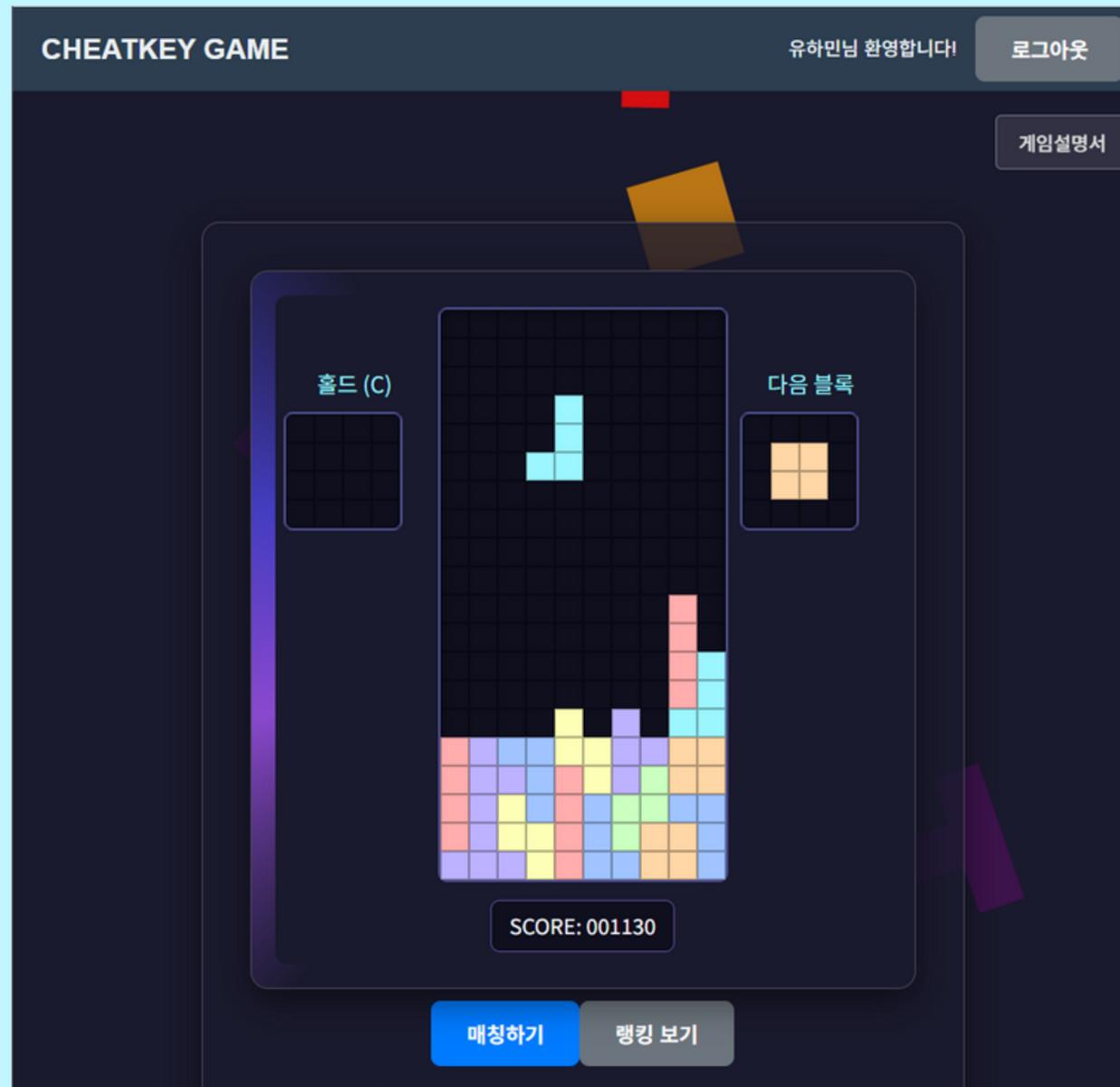
2. SIGNIN

로그인 페이지

- 입력
닉네임 또는 이메일 선택
- 비교
비밀번호를 **BCRYPT**로 해시화 한 값을 **USERS** 테이블과 비교
- 기능
식별자 (아이디 **OR** 이메일) 를 통한 로그인
로그인 성공 시 **JWT** 토큰 생성 (**ACCESS | REFRESH**)

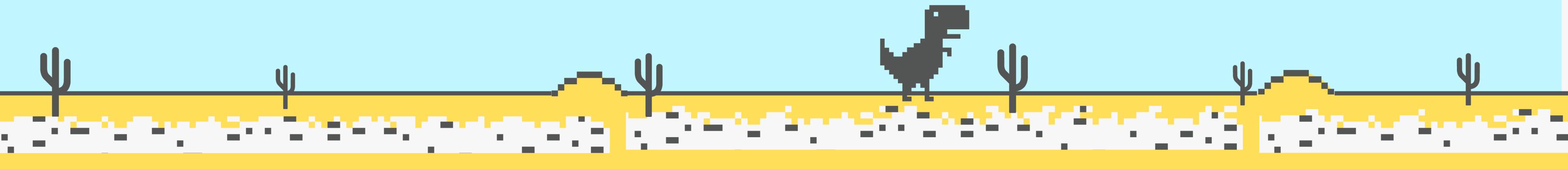


4. 개발 & 서비스 시연 - FE / BE

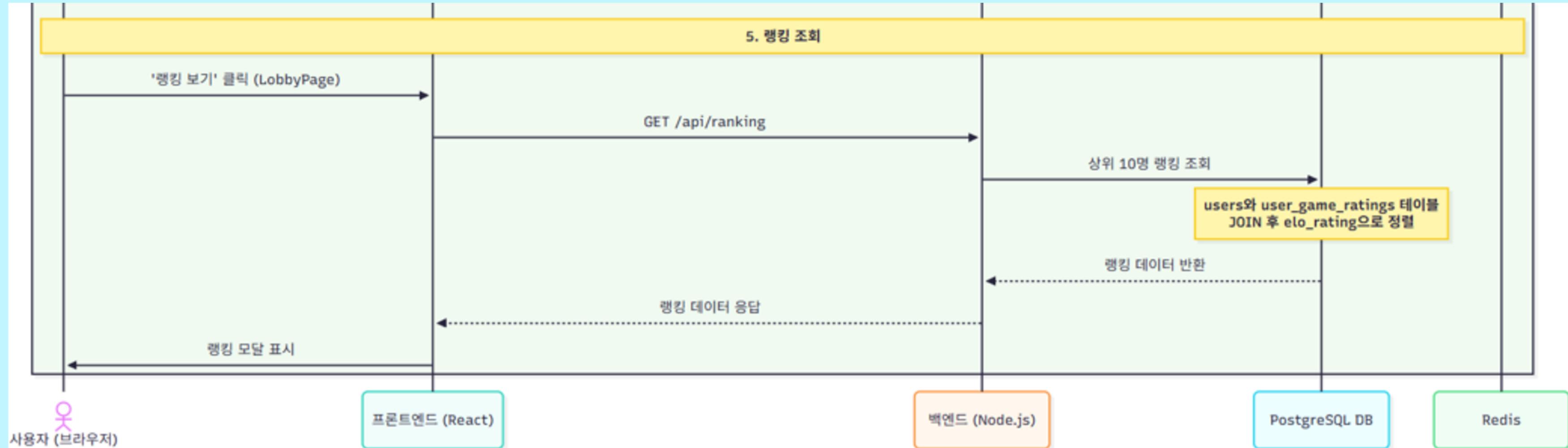


3. ↗ LOBBY 로비 페이지

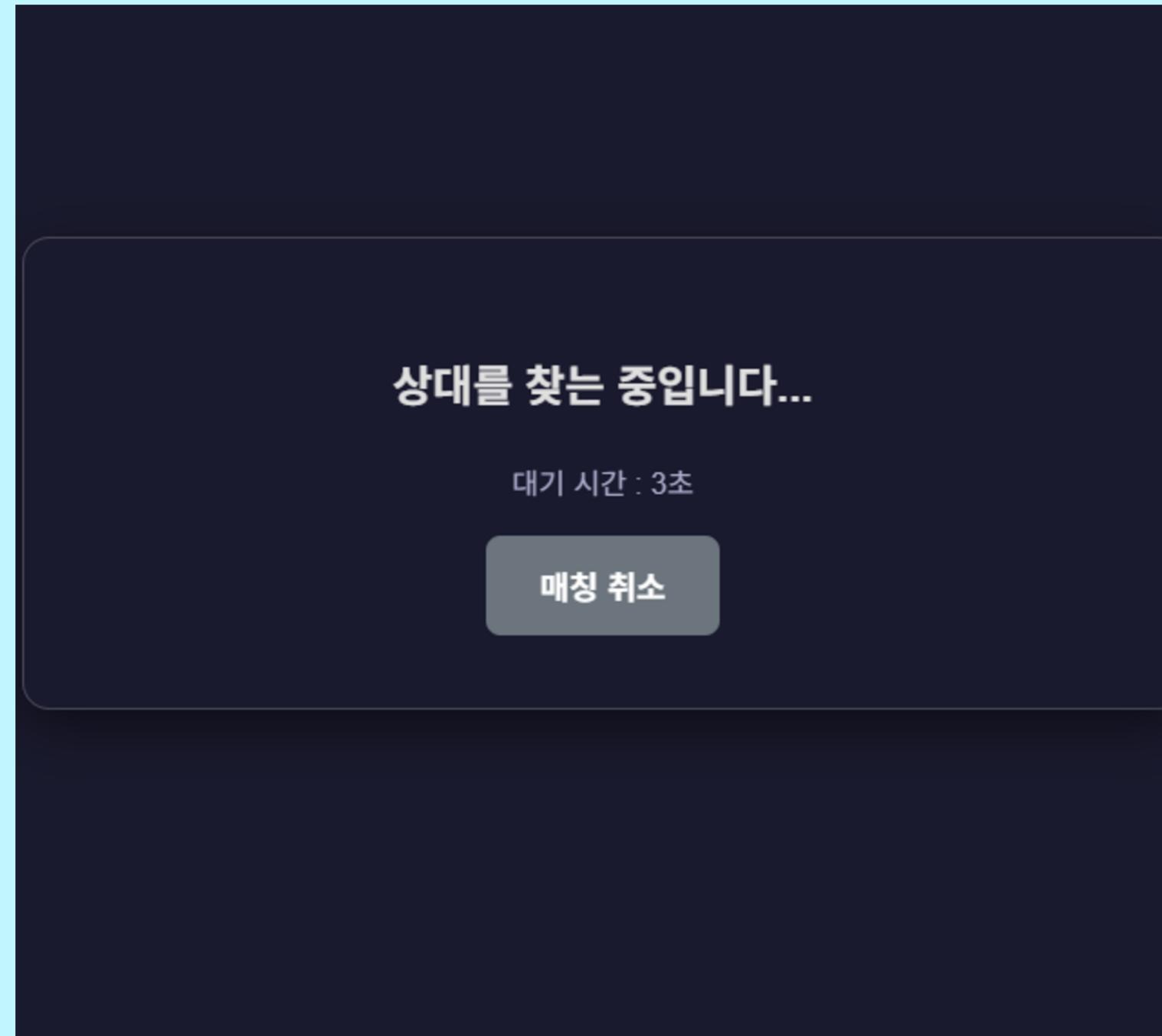
- 기능
 - 매칭하기 - **REDIS** 매칭 대기열 진입 기능
 - 랭킹보기 - 상위 10명의 점수 및 랭킹 출력 기능
 - 로그아웃 - 로그 아웃 기능
 - 게임설명서 - 게임 설명이 적힌 모달 컴포넌트 출력 기능



4. 개발 & 서비스 시연 - FE / BE



4. 개발 & 서비스 시연 - FE / BE



4. ↗TETRIS↗MATCH...

매칭 페이지

- 기능
레이팅 **150**점 이상 차이시 매칭 비성사
5초마다 매칭 상대 탐색
유령 매칭 방지



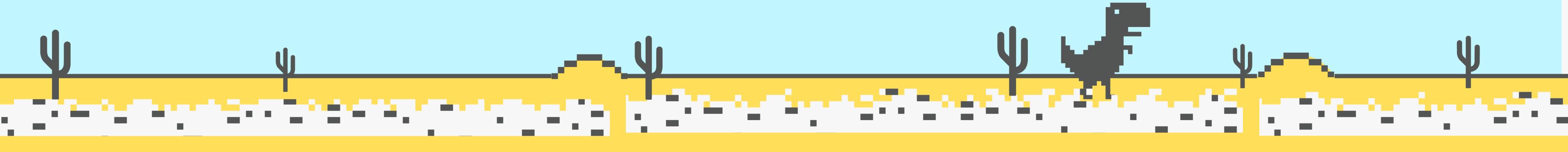
4. 개발 & 서비스 시연 - FE / BE



5. [/TETRIS/GAME_ID](#)

게임 페이지

- 기능
 - 고스트 블록 기능
 - 홀드키 – 원하는 블록 저장 기능
 - 스페이스키 – 하드 드롭 기능
 - 공격 기능 – 2줄이상 한 번에 클리어시 상대방에게 방해 블록 전송
 - 점수 출력 기능
 - 지진 효과
 - 화면 식별 기능 (상대방과 내 화면 구분)



4. 개발 & 서비스 시연 - FE / BE



6. ✓TETRIS✓RESULT

결과 페이지

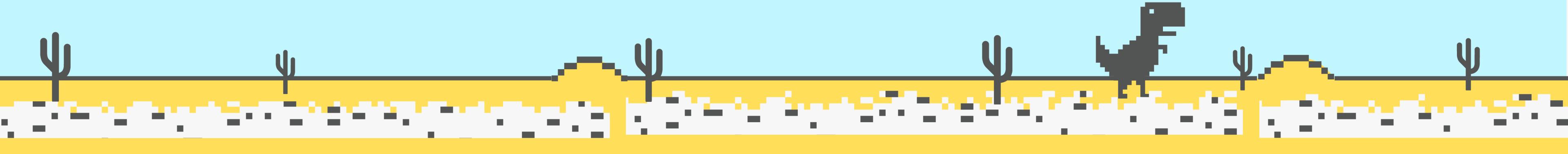
- 기능

레이팅 변동 안내 모달창 팝업

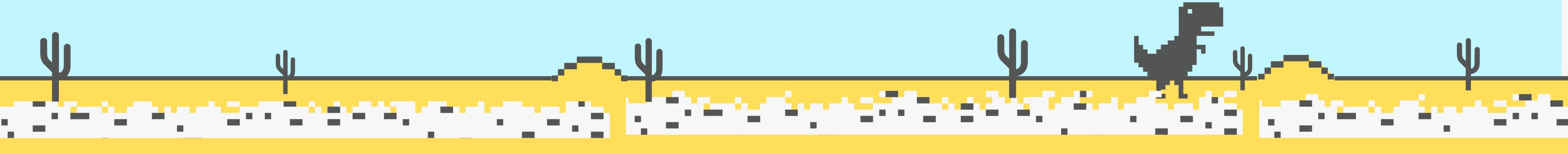
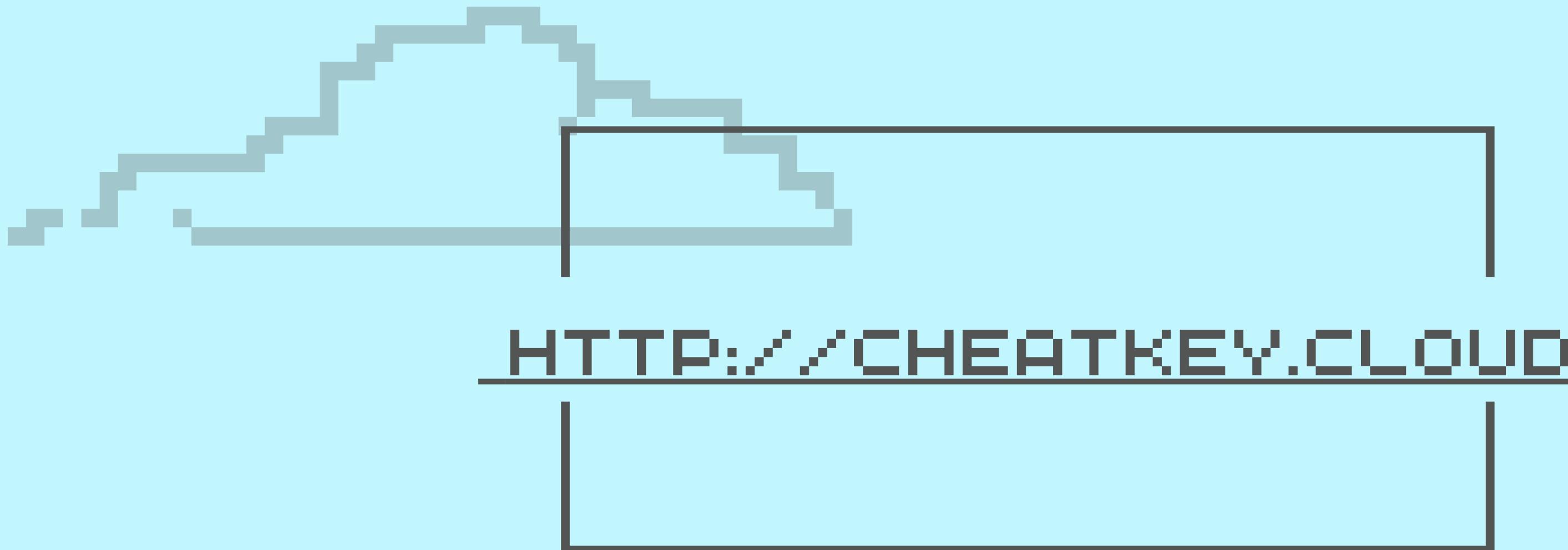
승자 : 레이팅 점수 상승

패자 : 레이팅 점수 감소

5초 후 자동 로비 페이지 이동

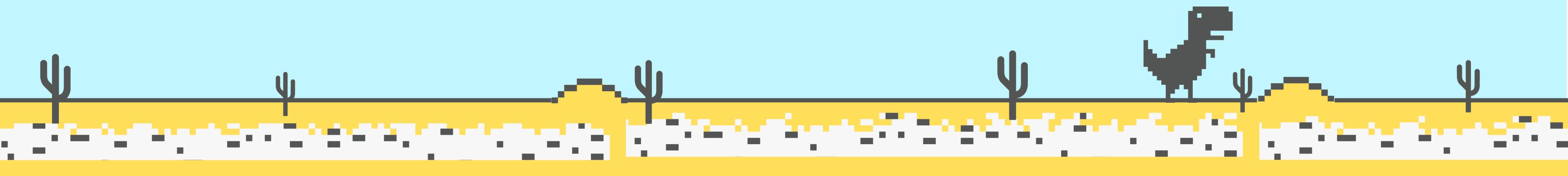
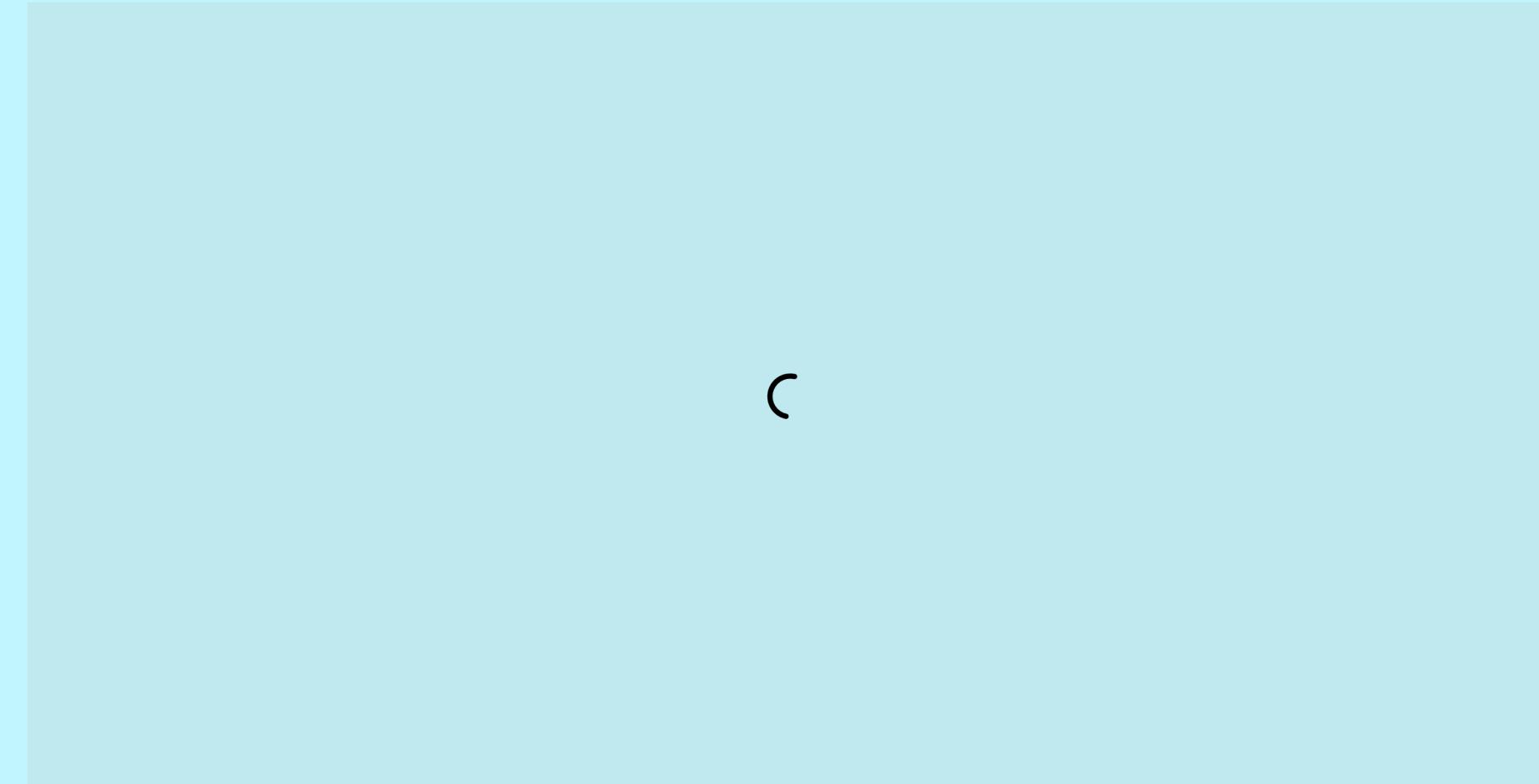


4. 개발 & 서비스 시연 - FE / BE

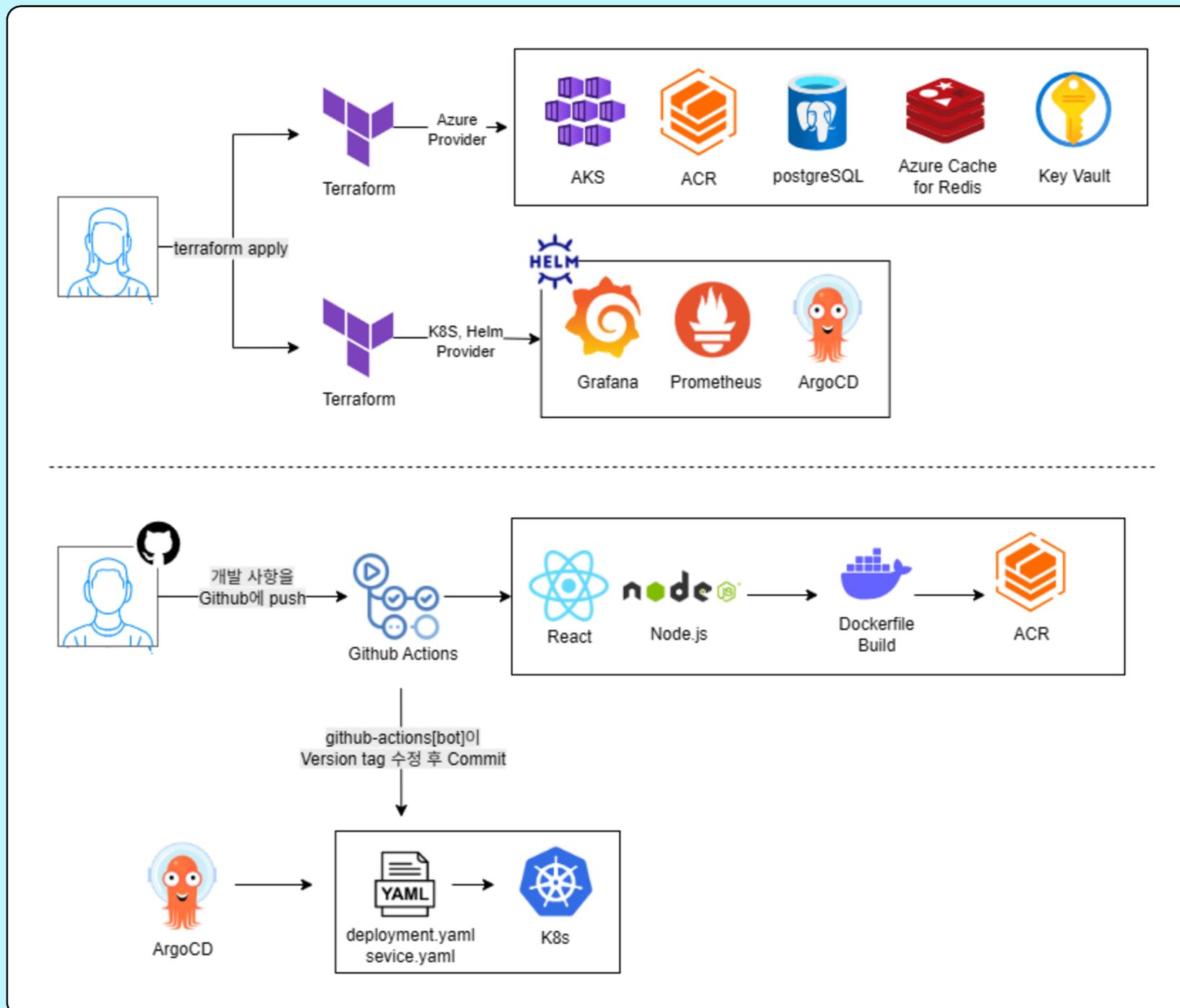


4. 개발 & 서비스 시연 - FE / BE

Front- End 개발기록



4. 개발 & 서비스 시연 - 인프라

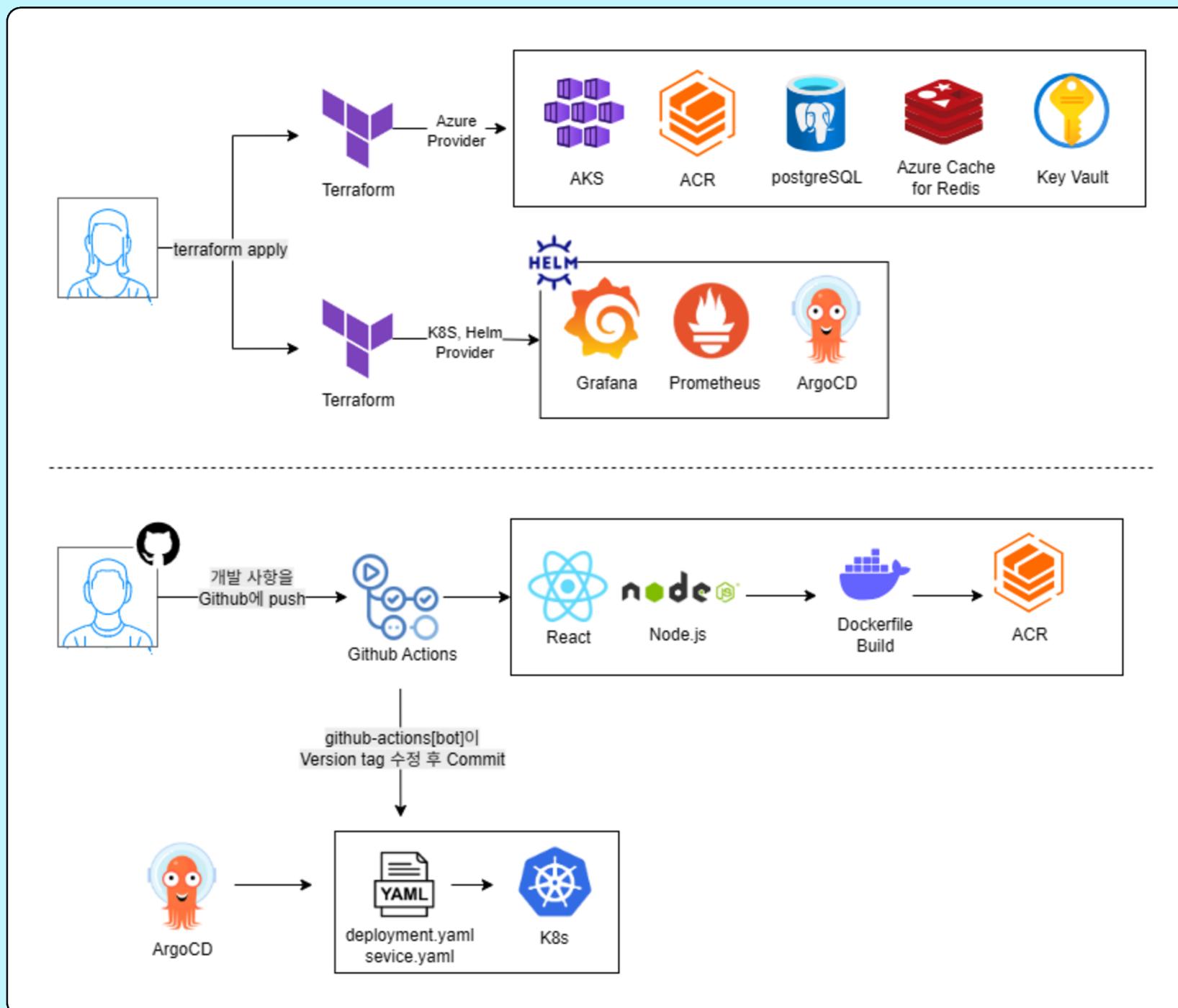


INFRA

- 리소스 생성
테라폼 **APPLY** 명령어로 **AZURE** 리소스, 쿠버네티스 리소스 오브젝트 생성

- **GITHUB ACTIONS**
개발 사항(프론트엔드, 백엔드) 수정을 통해 **GITHUB ACTIONS**에 커밋하면, **GITHUB ACTIONS** 워크플로우 작동

4. 개발 & 서비스 시연 - 인프라



INFRA

- DOCKER IMAGE BUILD
GITHUB ACTIONS가 DOCKER IMAGE BUILD 이후 ACR (AZURE CONTAINER REGISTRY)에 IMAGE PUSH
- DEPLOY
[GITHUB ACTIONS BOT]이 DEPLOYMENT.YAML 파일의 버전 태그를 수정하고 K8S 오브젝트와 이미지를 배포

4. 개발 & 서비스 시연 - TERRAFORM

terraform	
acr.tf	- ACR(Azure Container Registry)
aks.tf	- AKS (Azure Kubernetes Service)
cicd_identity.tf	- CI/CD용 Key Vault 관리 ID
database.tf	- PostgreSQL Server, Cache for Redis
group.tf	- 리소스 그룹
k8s-addons.tf	- K8s Add-on (추가 기능)
keyvault.tf	- Key Vault
main.tf	- 전반적인 구성의 정의
outputs.tf	- 생성된 리소스의 정보를 출력
terraform.tfvars	- 변수에 실제 값 할당 (.gitignore)
variables.tf	- 변수 선언 및 정의

```
hanbi 🐰 terraform terraform apply
data.azure_client_config.current: Reading...
data.azure_client_config.current: Read complete after 0s [id=Y2xpZW50Q29uZmlncy9jbGllbnRJZD0wNGIwNzc5NS04ZGRiLTQ2MWEtYmJlZS0wMmY5ZTFizjdINDY7b2JqZWN0SwQ9N2JjYjFmZmMtMje3ZC00NlwQwlwIxMtctMzc3ZTE4ZDk2ZjNh03N1YnNjcm1wdGlvbk1kPwMzOGQyZDI3LWU2MzEtNDRIiMC040Tk1LTY5N2JkMGVmNDJkNDt0ZW5hbnRJZD1iMGU20GI2Mi1hZmVhLTQxYWItYTRmMy1jNGIzNmV1NWQ5MTA=]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
<= read (data resources)

Terraform will perform the following actions:

# data.kubernetes_secret.argocd_initial_admin_secret will be read during apply
# (depends on a resource or a module with changes pending)
<= data "kubernetes_secret" "argocd_initial_admin_secret" {
    + data      = (sensitive value)
    + id       = (known after apply)
    + immutable = (known after apply)
```

형식 ↑↓
Kubernetes Service
관리 ID
키 자격 증명 모음
Azure Database for PostgreSQL 유연한 서버
Azure Cache for Redis
Container registry

리소스 그룹 > tetrisgame-rg > tetrisgame-aks

tetrisgame-aks | 워크로드

네임스페이스로 필터링

이름	네임스페이스	준비됨	기간 ↓	CPU	메모리
coredns	kube-system	✓ 2/2	14시간	● 메트릭 사용	
coredns-autoscaler	kube-system	✓ 1/1	14시간		
konnectivity-agent	kube-system	✓ 2/2	14시간		
konnectivity-agent-autoscaler	kube-system	✓ 1/1	14시간		
metrics-server	kube-system	✓ 2/2	14시간		
argocd-applicationset-controller	argocd	✓ 1/1	14시간		
argocd-dex-server	argocd	✓ 1/1	14시간		
argocd-notifications-controller	argocd	✓ 1/1	14시간		
argocd-redis	argocd	✓ 1/1	14시간		
argocd-repo-server	argocd	✓ 1/1	14시간		
argocd-server	argocd	✓ 1/1	14시간		
prometheus-stack-grafana	monitoring	✓ 1/1	14시간		
prometheus-stack-kube-prom-operator	monitoring	✓ 1/1	14시간		
prometheus-stack-kube-state-metrics	monitoring	✓ 1/1	14시간		

4. 개발 & 서비스 시연 - GITHUB ACTIONS

```
# Workflow 이름  
name: Build and Push Image to ACR (CI)  
  
# Github의 main 브랜치에 코드가 push될 때 워크플로우 실행  
on:  
  push:  
    branches: [ main ]  
  
# 워크플로우가 Git에 다시 Push할 수 있도록 권한 부여  
permissions:  
  contents: write  
  
# 환경변수 정의  
env:  
  ACR_NAME: tetricampacr  
  IMAGE_REPOSITORY: tetris-app  
  
jobs:  
  # ci라는 이름의 작업 정의  
  ci:  
    runs-on: ubuntu-latest  
  
    steps:  
      # 1. 소스 코드 체크아웃  
      # 2. 이미지 태그 설정  
      # 3. ACR에 로그인  
      # 4. Dockerfile로 이미지 build  
      # 5. 빌드된 Docker 이미지를 ACR로 push  
      # 6. Git 설정 및 YAML 파일 업데이트  
      # 7. 변경된 YAML 파일을 infra 브랜치에 다시 Push
```

The screenshot displays two main sections: the GitHub Actions interface and the Azure Container Registry (ACR) interface.

GitHub Actions Interface: The top part shows a workflow named "Build and Push Image to ACR (CI)" with a single job named "ci". The job status is "succeeded last week in 22s". The steps listed are: Set up job, Checkout source code, Set image tag, Login to Azure Container Registry, Build Docker image, Push Docker image to ACR, Update Kubernetes manifest, Commit and push manifest changes, Post Checkout source code, and Complete job.

Azure Container Registry (ACR) Interface: The bottom part shows the "tetris-backend" repository in ACR. It lists three tags: 1.2.6, 1.2.5, and 1.2.4. Each tag has a corresponding Docker image hash and timestamp.

GitHub Repository Interface: The bottom section shows the "cheat-key" repository. It displays recent pushes from the "infra" branch, showing a successful check for "Build and Push Image to ACR (CI) / ci (push)". The repository details show 0 stars, 0 forks, and 0 releases.

4. 개발 & 배포 시연 - ARGO CD

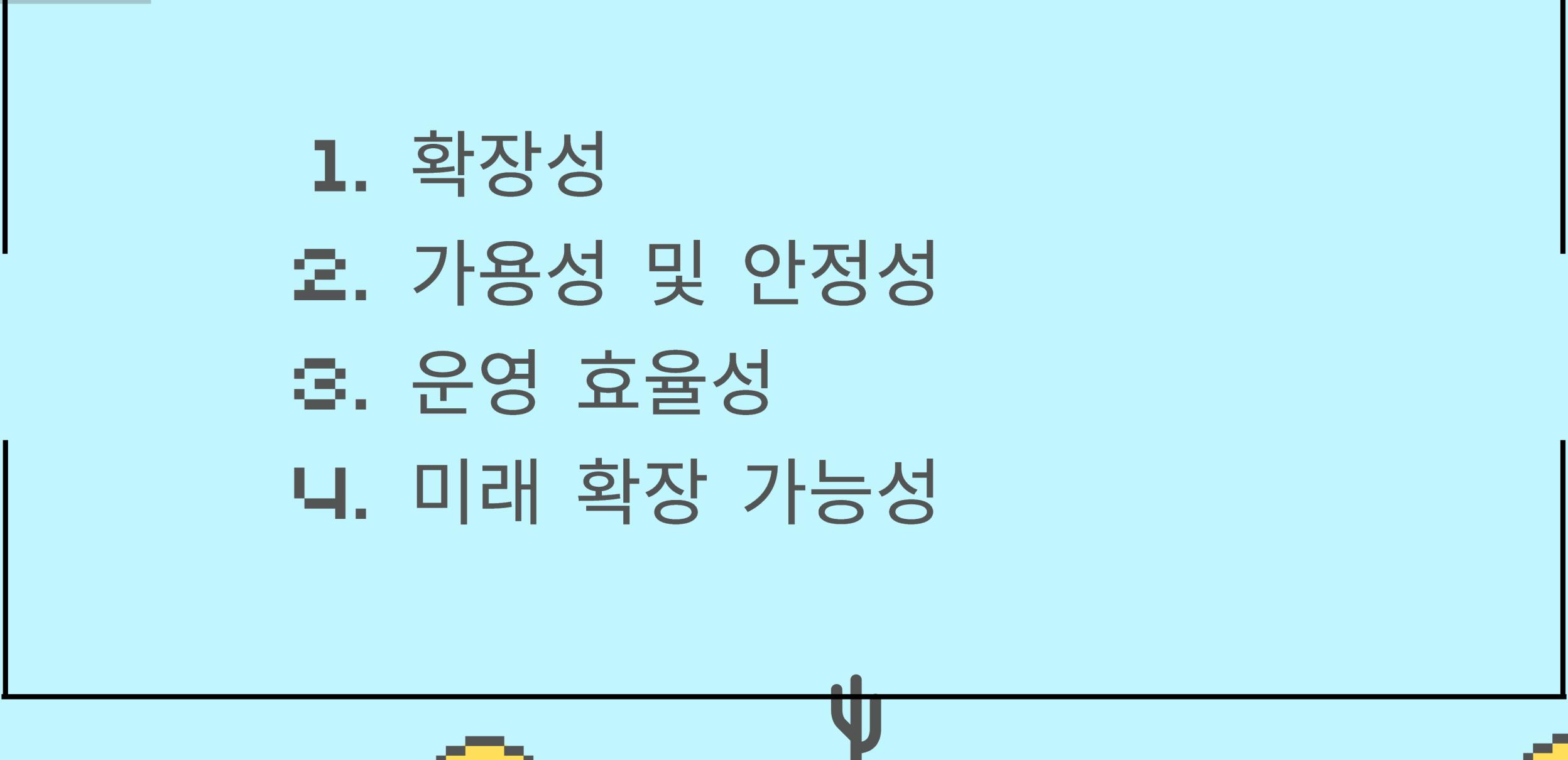
hanbi cheat-key kubectl get svc -n argocd

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
argocd-applicationset-controller	ClusterIP	10.0.150.2	<none>	7000/TCP	13h
argocd-dex-server	ClusterIP	10.0.89.247	<none>	5556/TCP,5557/TCP	13h
argocd-redis	ClusterIP	10.0.200.247	<none>	6379/TCP	13h
argocd-repo-server	ClusterIP	10.0.61.133	<none>	8081/TCP	13h
argocd-server	LoadBalancer	10.0.50.120	20.249.160.233	80:31704/TCP,443:31644/TCP	13h

20.249.160.233/applications/argocd/tetris-app?view=tree&resource=

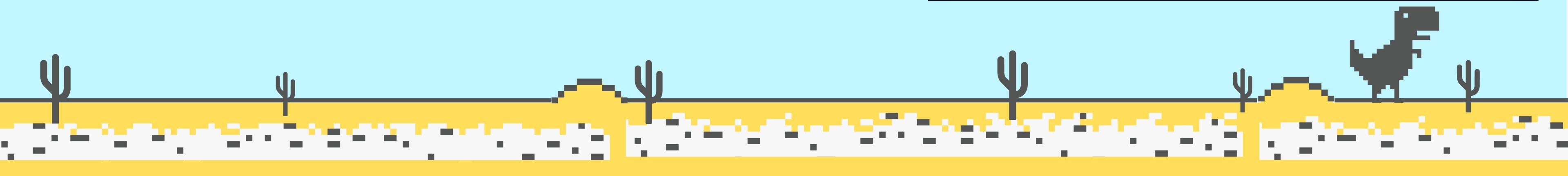
The screenshot shows the Argo CD web interface for the 'tetris-app'. The left sidebar has options for Applications, Settings, User Info, and Documentation. The main area shows the 'tetris-app' application details. The 'APP HEALTH' section indicates 'Progressing'. The 'SYNC STATUS' section shows 'Synced to main (de5befc)' with a green checkmark and 'Sync OK to de5befc' with a green checkmark. The 'APPLICATION DETAILS TREE' section displays a deployment graph with nodes like 'tetris-frontend-service', 'tetris-backend', 'tetris-frontend', and 'azur...'. A large black pixelated T-Rex is standing on the right side of the interface.

5. 회고 - 왜 클라우드?

- 
1. 확장성
 2. 가용성 및 안정성
 3. 운영 효율성
 4. 미래 확장 가능성

5. 호고 - 트러블 슈팅

1. Key vault 연결
2. Axios Error
3. 패자의 매칭 결과 값이 초기화됨
4. 라인 클리어시 상대 블록 삭제
5. ghost user 와 매칭



5. 회고 - 발전 가능성

1. 다양한 게임 서비스
2. 이메일 인증 기능
3. 쿠버네티스 배포



6. 느낀 점

유하민 :

R & R 을 구분하여 프로젝트를 진행했지만, 예상치 못한 이슈들을 함께 머리를 맞대고 해결해 나가는 과정이 즐거웠습니다

박한비 :

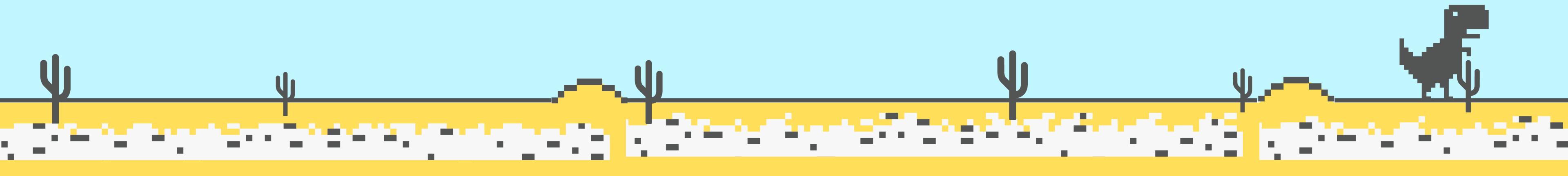
‘지속적인 배포’는 마무리짓지 못했지만, 그 과정에서 마주한 에러들을 해결하며 실제 운영 환경에 필요한 문제 해결 능력과 자신감을 얻었습니다.

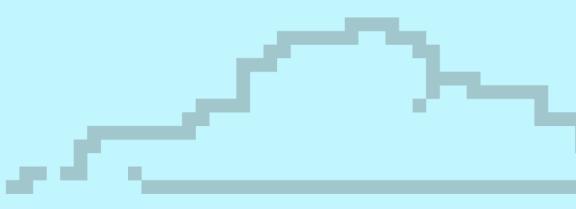
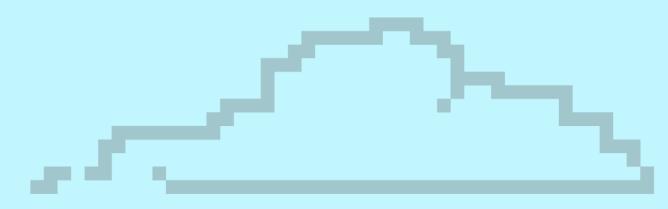
박상우 :

코드 시퀀스의 이해도가 더 올라가는 시간이었고, 디버깅에 성공했을 때의 달성감을 알게 되었습니다.

이지후 :

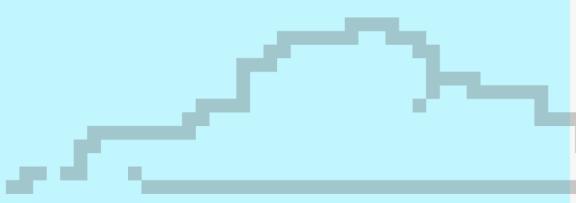
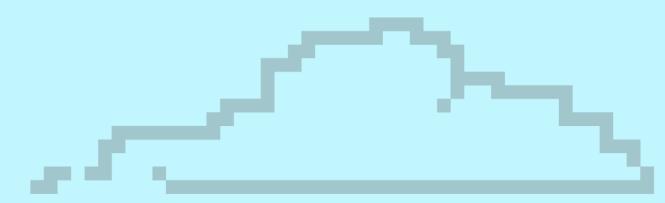
프로그램을 직접 실행하고 테스트하며 기능을 개선하고 추가하는 과정에서 완성도가 높아지는 것을 보며 큰 성취감을 얻었습니다





THANK YOU

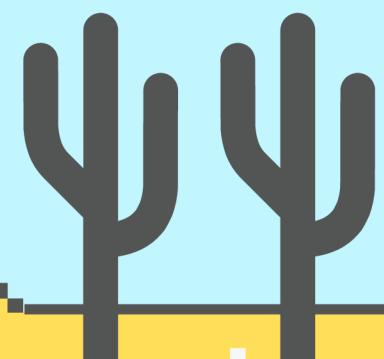




Q

S

A



STAGE CLEAR

