아카리(Akari) 프로젝트 기술 명세서
(Technical Documentation)

최종 수정일: 2025-12-27
프로젝트 상태: 배포 완료 (Production Ready) / 클라우드 네이티브

1. 프로젝트 개요 (Overview)

- **프로젝트명:** 아카리 (Akari / あかり)
- **유형:** Discord 인터랙티브 봇 (Interactive Bot)
- **목적:** 디스코드 서버 내 음성 지원(TTS), 사용자 관리, 상호작용 기능 제공.

특징:

- 24/7 무중단 운영: 로컬 PC가 아닌 클라우드 서버에서 상시 가동.
- 데이터 영속성: 서버 재부팅 후에도 데이터가 유지되는 외부 데이터베이스 연동.
- 자동 배포: 코드 수정 시 자동으로 서버에 반영되는 CI/CD 파이프라인 구축.

2. 기술 스택 (Tech Stack)

2.1. 언어 및 프레임워크

- **Language:** Python 3.10.x
- **Core Library:**discord.py (Discord API 비동기 래퍼)
- **Web Server:**Flask (헬스 체크용 경량 웹 서버)
- **Voice Engine:** FFmpeg (오디오 스트리밍 처리)

2.2. 데이터베이스 &스토리지

- **DBMS:** PostgreSQL 15+ (Supabase Cloud)
- **Driver:**psycopg2-binary
- **Connection Mode:** Session Pooler (IPv4 호환성 확보)

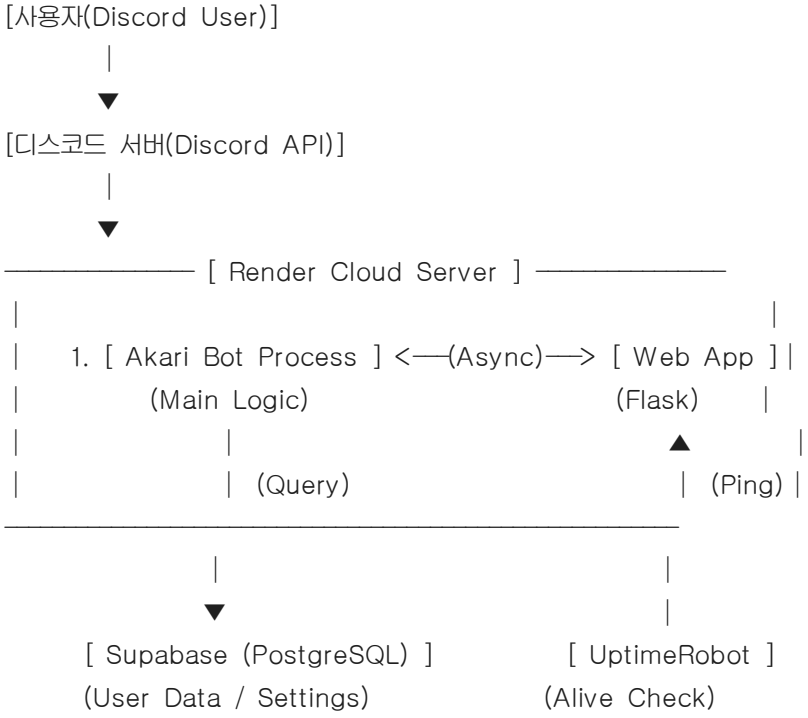
2.3. 인프라 및 배포

- **Hosting:** Render (Cloud Application Hosting - Linux Environment)
- **Monitoring:** UptimeRobot (5분 주기 HTTP 핑)
- **VCS:** GitHub (소스 코드 버전 관리)

3. 시스템 아키텍처 (System Architecture)

전체 시스템은 크게 **봇(Bot), 웹 서버(Web), 데이터베이스(DB)** 세 부분으로 구성됩니다.

코드 스니펫



1. **Akari Bot:** 디스코드 이벤트를 수신하고 로직을 처리.
2. **Web App:** 외부(UptimeRobot)에서 보내는 접속 요청을 받아 "서버가 살아있음"을 증명.
3. **Supabase:** 봇이 재시작되어도 사라지지 않는 영구적인 데이터를 저장.

4. 디렉토리 구조 (Directory Structure)

불필요한 로컬 DB 파일(db/, data/)이 제거된 **클린 아키텍처** 상태입니다.

Plaintext

```
akari/
├── .env                    # 환경변수 설정 파일 (보안 주의: API 키, DB 주소)
├── .gitignore              # Git 업로드 제외 목록 (venv, .env 등)
├── build.sh                # Render 배포용 빌드 스크립트 (FFmpeg 설치 등)
├── main.py                 # 프로그램 진입점 (Entry Point)
├── requirements.txt        # 파이썬 의존성 패키지 목록
├── README.md               # 프로젝트 설명서
│
├── bot/                    # 봇 핵심 로직 디렉토리
│   ├── commands/           # (확장 가능) 슬래시 커맨드 모듈
│   ├── client.py           # Discord Bot 클래스 정의 및 이벤트 핸들러
│   ├── config.py           # 환경변수 로드 및 설정 관리
│   └── database.py         # Supabase(PostgreSQL) 연결 및 쿼리 관리
│
├── features/               # 기능별 모듈 디렉토리
│   └── voice_service.py    # 음성 채널 접속 및 TTS 재생 로직
│
├── web/                    # 헬스 체크용 웹 서버 디렉토리
│   └── app.py              # Flask 앱 정의 및 실행 로직
│
└── docs/                   # 프로젝트 관련 문서 보관
```

5. 핵심 모듈 상세 명세

A. 메인 실행부 (main.py)

- **역할:** 프로그램의 오케스트라 지휘자.
- **기능:**
  1. database.py의 init_db()를 호출하여 DB 테이블 존재 여부 확인 및 생성.
  2. 별도 스레드(Thread)를 생성하여 Flask 웹 서버 실행.
  3. discord.py 클라이언트를 실행하여 봇 온라인 전환.

B. 데이터베이스 모듈 (bot/database.py)

- **변경 사항:** 기존 SQLite(sqlite3)에서 PostgreSQL(psycopg2)로 마이그레이션 완료.
- **기능:**
  1. 환경변수 DATABASE_URL을 통해 보안 접속.
  2. SSL 모드(sslmode='require')를 사용하여 데이터 전송 암호화.
  3. 테이블 초기화 및 CRUD(데이터 생성, 읽기, 수정, 삭제) 기능 제공.

C. 웹 서버 모듈 (web/app.py)

- **역할:** Render 서버의 절전 모드(Sleep) 방지.
- **기능:**
  1. 0.0.0.0 포트 8080으로 HTTP 서버 개방.
  2. 루트 경로(/) 접속 시 200 OK 응답 반환.
  3. UptimeRobot이 이 주소를 주기적으로 호출하여 서버를 깨어있는 상태로 유지.

D. 음성 서비스 (features/voice_service.py)

- **역할:** 음성 채널 상호작용.
- **기능:**
  1. 사용자가 있는 음성 채널 감지 및 접속.
  2. FFmpegPCMAudio를 이용한 오디오 스트림 재생.
  3. (참고) 호환성 문제로 audioop 라이브러리 의존성 제거됨.

E. 빌드 스크립트 (build.sh)

- **역할:** Render 리눅스 서버 환경 설정.
- **내용:**
  1. OS 레벨의 패키지(ffmpeg) 다운로드 및 설치.

2. Python 패키지(pip install -r requirements.txt) 설치.

6. 환경 변수 (Environment Variables)

보안 경고: 이 내용은 .env 파일과 Render 대시보드의 Environment 탭에서만 관리되어야 합니다. 외부 유출 시 해킹 위험이 있습니다.

키(Key)	설명	관리 위치
DISCORD_TOKEN	디스코드 봇 인증 토큰	Render Env / Local .env
DATABASE_URL	Supabase PostgreSQL 접속 주소 (Session Pooler)	Render Env / Local .env
FLASK_PORT	(선택) 웹 서버 포트 (기본값: 8080)	코드 내 기본값 사용

7. 배포 및 유지보수 프로세스

1. **로컬 개발:** VS Code에서 기능 개발 및 python main.py로 테스트.
2. **커밋 및 푸시:**git push 명령어로 GitHub에 코드 업로드.
3. **자동 배포:** Render가 GitHub의 변경사항을 감지하여 자동으로:
   - 기존 프로세스 종료.
   - build.sh 실행 (환경 재설정).
   - 새 버전의 봇 실행.
4. **모니터링:** Render Logs 및 UptimeRobot을 통해 상태 확인.

[기록자 코멘트]

현재 프로젝트는 확장성과 안정성을 모두 갖춘 상태입니다. 로컬 파일 시스템에 의존하지 않으므로, 언제든지 서버를 옮기거나 확장해도 데이터 손실이 발생하지 않는 구조입니다.