IITP DABT Platform

배포 설치 가이드(자립 허브)

문서 버전: 1.0.0
작성일: 2025-11-03

(주)스위트케이

문서 History

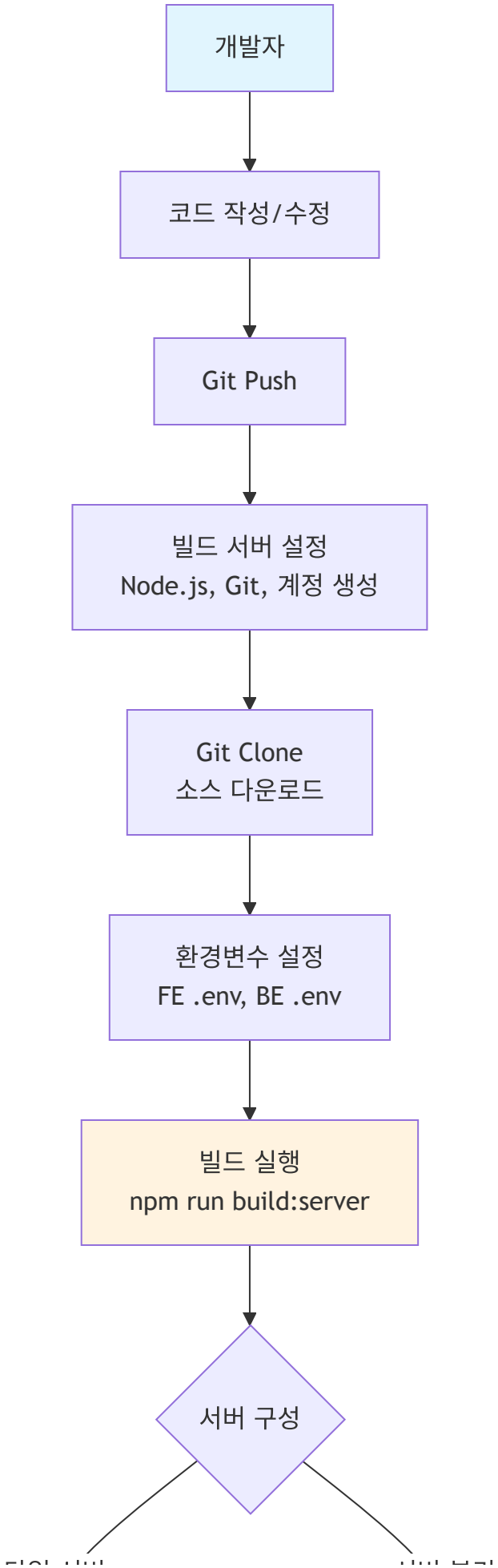
버전	날짜	변경 내용	작성자
v1.0.0	2025-11-03	최초 작성	(주)스위트케이

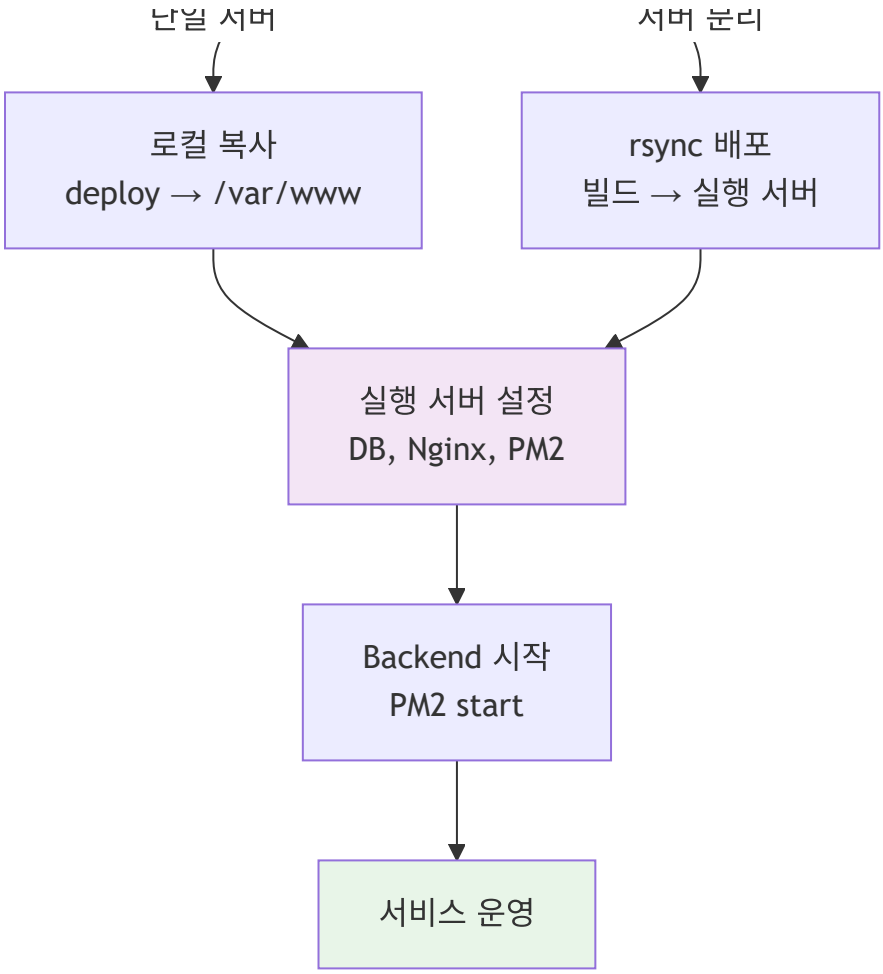
목차

0. 개요

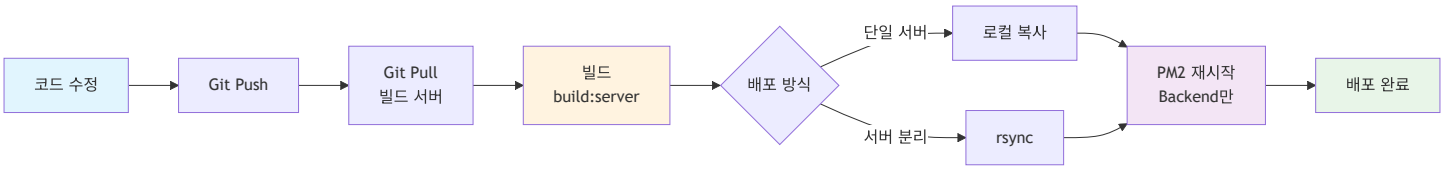
0.1 배포 Flow

전체 Flow (초기 설치)

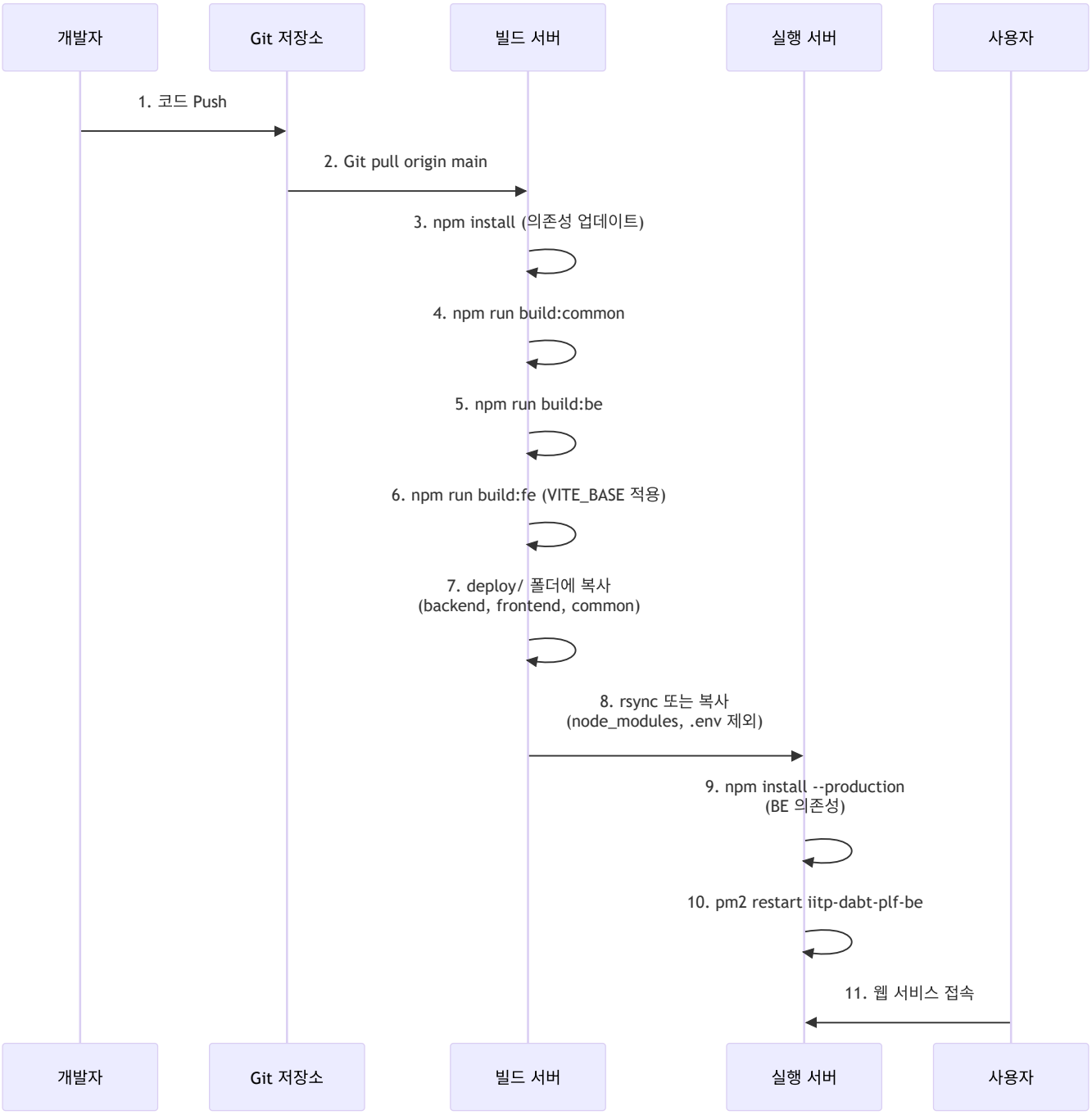

개발자
코드 작성/수정
Git Push
빌드 서버 설정
Node.js, Git, 계정 생성
Git Clone
소스 다운로드
환경변수 설정
FE .env, BE .env
빌드 실행
npm run build:server
서버 구성



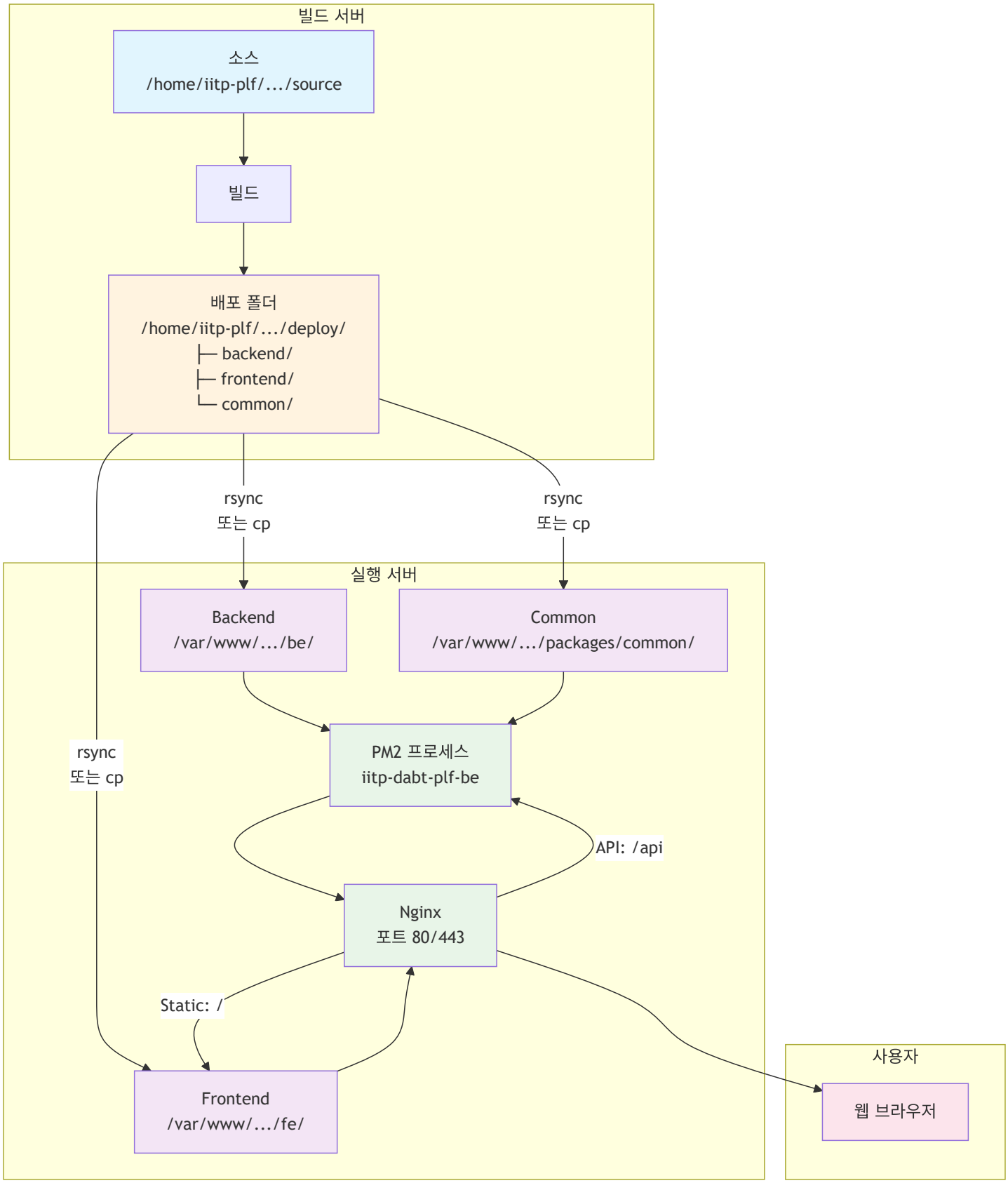
업데이트 배포 Flow (일상 운영)



상세 빌드 및 배포 과정


개발자
Git 저장소
빌드 서버
실행 서버
사용자
1. 코드 Push
2. Git pull origin main
3. npm install (의존성 업데이트)
4. npm run build:common
5. npm run build:be
6. npm run build:fe (VITE_BASE 적용)
7. deploy/ 폴더에 복사
(backend, frontend, common)
8. rsync 또는 복사
(node_modules, .env 제외)
9. npm install --production
(BE 의존성)
10. pm2 restart iitp-dabt-plf-be
11. 웹 서비스 접속
개발자
Git 저장소
빌드 서버
실행 서버
사용자

디렉토리 구조 및 배포 경로



0.2 서버 환경 선택

단일 서버 환경 (섹션 1):

- 빌드 서버 = 실행 서버 (같은 서버에서 빌드와 실행)

서버 분리 환경 (섹션 2):

- 빌드 서버 ≠ 실행 서버 (서버 분리)

0.3 시스템 요구사항

공통 요구사항:

- OS: Ubuntu 22.04+ (또는 CentOS 7+, Debian 10+)
- Node.js: 22.x 이상
- npm: 9.x 이상
- PostgreSQL: 16+ 이상
- Nginx: 1.18 이상
- PM2: 최신 버전
- Git: 2.x 이상

하드웨어 권장사항:

- CPU: 2 Core 이상
- RAM: 4GB 이상
- Disk: 20GB 이상 여유 공간
-

1. 초기 설치 - 단일 서버 환경

> **전제조건**: 빌드 서버 = 실행 서버 (같은 서버에서 모든 작업 수행)

1.0 서버 기본 세팅

> 👤 **실행 계정**: `root` 또는 sudo 권한이 있는 계정

```
# Ubuntu 20.04+ 기준
sudo apt update && sudo apt upgrade -y

# 필수 패키지 설치
sudo apt install -y git curl unzip jq build-essential nginx
```

1.0.1 Node.js 설치 (아래 중 하나 선택)

> **프로덕션 서버 권장: NodeSource (방법 1)**
>
> - 시스템 레벨 설치로 운영 안정성 우수
> - 여러 관리자가 동일한 환경 사용 가능
> - PM2, sudo 등 관리 도구와 호환성 최고

방법 1: NodeSource 사용 (프로덕션 권장)

```
# 비대화형 모드로 설치 (대화형 프롬프트 방지)
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo DEBIAN_FRONTEND=noninteractive bash -
sudo DEBIAN_FRONTEND=noninteractive apt install -y nodejs

# 버전 확인
node -v  # v22.x.x
npm -v   # 9.x.x 이상
```

> **Tip**: `DEBIAN_FRONTEND=noninteractive` 는 설치 중 대화형 화면을 완전히 비활성화합니다. 자동화 스크립트에 필수입니다.

방법 2: snap 사용 (간단, 자동 업데이트)

```
sudo snap install node --classic --channel=22

# 버전 확인
node -v
npm -v
```

방법 3: nvm 사용 (개발 환경 또는 버전 관리 필요 시)

```
# nvm 설치
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash
source ~/.bashrc

# Node.js 22 설치
nvm install 22
nvm use 22
nvm alias default 22

# 버전 확인
node -v  # v22.x.x
npm -v   # 9.x.x 이상
```

> **nvm에서 다른 방법으로 전환하려면**
>
> nvm으로 설치했다가 NodeSource 등으로 전환하려면 nvm을 완전히 제거해야 합니다:

```
# 1. nvm 비활성화 및 Node.js 삭제
nvm deactivate
nvm uninstall 22  # 설치된 모든 버전 삭제

# 2. nvm 디렉토리 삭제
rm -rf ~/.nvm

# 3. 모든 셸 설정 파일에서 nvm 관련 라인 삭제
# nvm 설정이 있는 파일 찾기
grep -l "NVM_DIR" ~/.bashrc ~/.profile ~/.bash_profile ~/.zshrc 2>/dev/null

# 발견된 각 파일 편집 (예시)
vi ~/.bashrc
vi ~/.profile
# 아래 라인들 삭제:
# export NVM_DIR="$HOME/.nvm"
# [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
# [ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"

# 4. 완전히 로그아웃 후 재로그인 (필수)
exit
# SSH 재접속 또는 콘솔 재로그인

# 5. 완전히 제거 확인
nvm --version  # command not found
which node  # 아무것도 나오지 않아야 함

# 6. 이제 NodeSource 등 다른 방법으로 설치
```

설치 확인 방법 (설치 방식별):

```
# 공통 확인
node -v  # v22.x.x
npm -v   # 9.x.x 이상

# nvm으로 설치한 경우
nvm current  # v22.x.x
nvm list     # 설치된 버전 목록

# snap으로 설치한 경우
snap list | grep node

# NodeSource로 설치한 경우
apt list --installed | grep nodejs
```

1.0.2 PM2 및 PostgreSQL 설치

PM2 설치 (Node.js 설치 방법에 따라 다름):

```
# NodeSource 또는 snap으로 설치한 경우 (권장)
sudo npm install -g pm2

# nvm으로 설치한 경우
npm install -g pm2

# 설치 확인
pm2 --version
which pm2
```

PM2 시작 스크립트 등록 (부팅 시 자동 시작):

```
# PM2 startup 명령 실행 (사용자에 맞는 명령어 출력됨)
pm2 startup

# 출력되는 명령어를 복사해서 실행 (예시)
# sudo env PATH=$PATH:/home/iitp-plf/.nvm/versions/node/v22.x.x/bin ...
# 위 명령어가 자동으로 올바른 경로를 포함하므로 그대로 실행하면 됩니다.
```

PostgreSQL 및 Nginx 확인:

```
# PostgreSQL 설치 (이미 설치되어 있으면 생략)
sudo apt install -y postgresql postgresql-contrib

# Nginx 상태 확인
sudo systemctl status nginx
sudo nginx -t
```

1.1 운영 계정 및 디렉토리 구조 생성

> **실행 계정**: `root` 또는 sudo 권한이 있는 계정

```
# iitp-plf 사용자 생성 , 이미 서비스 계정이 있으면 스킵
sudo useradd -m -s /bin/bash iitp-plf
sudo passwd iitp-plf

# 빌드/소스 디렉토리 생성
sudo mkdir -p /home/iitp-plf/iitp-dabt-platform/source
sudo mkdir -p /home/iitp-plf/iitp-dabt-platform/deploy
sudo chown -R iitp-plf:iitp-plf /home/iitp-plf/iitp-dabt-platform

# 실행 디렉토리 생성
sudo mkdir -p /var/www/iitp-dabt-platform/be
sudo mkdir -p /var/www/iitp-dabt-platform/fe
sudo mkdir -p /var/www/iitp-dabt-platform/script
sudo mkdir -p /var/www/iitp-dabt-platform/packages/common
sudo chown -R iitp-plf:iitp-plf /var/www/iitp-dabt-platform

# 디렉토리 구조 확인
tree -L 2 /home/iitp-plf/iitp-dabt-platform
tree -L 2 /var/www/iitp-dabt-platform
```

디렉토리 설명:

- `/home/iitp-plf/iitp-dabt-platform/source` : Git 소스 코드
- `/home/iitp-plf/iitp-dabt-platform/deploy` : 빌드 결과물
- `/var/www/iitp-dabt-platform/be` : Backend 실행 디렉토리
- `/var/www/iitp-dabt-platform/fe` : Frontend 정적 파일
- `/var/www/iitp-dabt-platform/packages/common` : 공통 패키지 (BE에서 참조)

1.2 데이터베이스 설정

```
# PostgreSQL 접속
sudo -u postgres psql

# 데이터베이스 생성
CREATE DATABASE iitp_dabt;

# Platform용 사용자 생성
CREATE USER iitp_platform_user WITH PASSWORD 'your_secure_password';

# 권한 부여
GRANT ALL PRIVILEGES ON DATABASE iitp_dabt TO iitp_platform_user;

# 종료
\q
```

보안 강화 (선택사항):

```
# PostgreSQL 외부 접속 차단 (로컬만 허용)
sudo vi /etc/postgresql/*/main/pg_hba.conf
# local   all   iitp_platform_user   md5

sudo systemctl restart postgresql
```

1.3 프로젝트 클론 및 초기 설정

> **실행 계정**: `iitp-plf` (운영 계정)

```
# iitp-plf 사용자로 전환 (root 또는 sudo 권한 계정에서)
sudo -iu iitp-plf

# Git에서 소스 다운로드
cd /home/iitp-plf/iitp-dabt-platform/source
git clone https://github.com/sweetk-dev/06-IITP-DABT-Platform.git .

# 브랜치 확인 (main 브랜치 사용)
git branch
git status

# 전체 패키지 설치
npm install

# 설치 확인
ls -la node_modules/
ls -la packages/common/node_modules/
ls -la be/node_modules/
ls -la fe/node_modules/
```

1.4 환경변수 설정

> **실행 계정**: `root` 또는 sudo 권한이 있는 계정 (서버 디렉토리 접근용)

1.4.1 Backend 환경변수 (실행 서버용)

Backend는 실행 시에만 환경변수 필요 (빌드 시 불필요)

```
# 실행 서버 디렉토리에 .env 생성 (최초 1회)
sudo vi /var/www/iitp-dabt-platform/be/.env
```

내용:

```
# 서버 설정
NODE_ENV=production
PORT=33000

# 데이터베이스 설정
DB_HOST=localhost
DB_PORT=5432
DB_NAME=iitp_dabt
DB_USER=iitp_platform_user
DB_PASSWORD=your_secure_password
DB_SSL=false

# CORS 설정 (실제 서버 주소로 변경)
CORS_ORIGINS=http://your-server-ip-or-domain

# OpenAPI 서버 설정 (실제 값으로 변경)
OPEN_API_SERVER_URL=https://api.example.com
OPEN_API_AUTH_KEY=your_api_key_here
OPEN_API_AUTH_SECRET=your_api_secret_here
OPEN_API_PAGE_SIZE=100
OPEN_API_TIMEOUT=30000

# 로깅 설정
LOG_LEVEL=info
LOG_DIR=./logs

# 보안 설정
ENC_SECRET=your_encryption_secret_key_here

# 기타 설정
API_RATE_LIMIT=100
REQUEST_TIMEOUT=30000
```

권한 설정:

```
sudo chown iitp-plf:iitp-plf /var/www/iitp-dabt-platform/be/.env
sudo chmod 600 /var/www/iitp-dabt-platform/be/.env
```

1.4.2 Frontend 빌드 환경변수 (빌드 서버용)

Frontend는 빌드 시에만 환경변수 필요 (실행 시 불필요)

방법 1: .env 파일 사용 (권장)

```
# 빌드 서버 소스 디렉토리에 .env 생성
cd /home/iitp-plf/iitp-dabt-platform/source/fe
cp env.sample .env
vi .env
```

내용 (단독 설치 기준):

```
# 프로덕션 빌드용 설정 (단독 설치)
VITE_PORT=5173
VITE_BASE=/
VITE_API_BASE_URL=http://your-server-ip-or-domain
VITE_API_TIMEOUT=10000
VITE_API_DATA_PREVIEW_LIMIT=10
VITE_VISUAL_TOOL=http://your-server-ip:visual-tool-port/
VITE_EMPLOYMENT_SITE_URL=https://www.ablejob.co.kr/
VITE_OPEN_API_CENTER_URL=http://your-server-ip/admin/
VITE_OPEN_API_CENTER_ABOUT_URL=http://your-server-ip/admin/about
```

중요 설정 설명:

- `VITE_BASE=/` : 단독 설치는 루트 경로 사용
- `VITE_API_BASE_URL=http://your-server-ip-or-domain` : API 서버 주소 (프록시 없이 직접 호출 시)
  - Nginx 프록시 사용 시: `VITE_API_BASE_URL=` (빈 값 또는 상대 경로)
- FE 코드가 자동으로 `/api/v1/...` 을 추가하므로 `/api` 포함 금지

방법 2: shell 환경변수 export (대안)

```
export VITE_PORT=5173
export VITE_BASE=/
export VITE_API_BASE_URL=http://your-server-ip
export VITE_API_TIMEOUT=10000
export VITE_API_DATA_PREVIEW_LIMIT=10
export VITE_VISUAL_TOOL=http://your-server-ip:visual-tool-port/
export VITE_EMPLOYMENT_SITE_URL=https://www.ablejob.co.kr/
export VITE_OPEN_API_CENTER_URL=http://your-server-ip/admin/
export VITE_OPEN_API_CENTER_ABOUT_URL=http://your-server-ip/admin/about
```

1.4.3 빌드 스크립트 환경변수 (선택사항)

자동화 스크립트 사용 시:

```
cd /home/iitp-plf/iitp-dabt-platform/source
cp script/server/env.sample.build-server script/server/.env
vi script/server/.env
```

내용:

```
# Git 설정
GIT_REPO_URL=https://github.com/sweetk-dev/06-IITP-DABT-Platform.git
GIT_BRANCH=main

# 경로 설정
SOURCE_PATH=/home/iitp-plf/iitp-dabt-platform/source
DEPLOY_PATH=/home/iitp-plf/iitp-dabt-platform/deploy

# 실행 서버 경로 (단일 서버이므로 localhost 또는 생략)
PROD_BE_PATH=/var/www/iitp-dabt-platform/be
PROD_FE_PATH=/var/www/iitp-dabt-platform/fe
OPS_SCRIPT_PATH=/var/www/iitp-dabt-platform/script

# 빌드 설정
NODE_ENV=production
NPM_CONFIG_PRODUCTION=true
```

1.5 빌드

> 👤 **실행 계정**: `iitp-plf` (운영 계정)

1.5.1 전체 빌드 (기본, 권장)

```
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 빌드 (common → be → fe 순서로 자동 빌드)
npm run build:server
```

빌드 과정:

1. Git pull (최신 코드)

2. npm install (의존성 업데이트)
3. Common 빌드
4. Backend 빌드 (빌드 정보 자동 생성)
5. Frontend 빌드 (환경변수 적용)
6. deploy/ 폴더로 복사

빌드 확인:

```
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/backend/dist/
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/frontend/
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/common/dist/

# 빌드 정보 확인
cat /home/iitp-plf/iitp-dabt-platform/deploy/backend/build-info.json
```

1.5.2 개별 빌드 (옵션)

언제 사용하나?

- 특정 부분만 수정했을 때
- 빠른 빌드가 필요할 때
- 전체 빌드는 과할 때

주의: 의존성이 있으므로 순서 중요!

Common만 빌드:

```
npm run build:server:common

# 확인
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/common/dist/
```

Backend만 빌드:

```
# Common이 먼저 빌드되어 있어야 함!
npm run build:server:be

# 확인
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/backend/dist/
cat /home/iitp-plf/iitp-dabt-platform/deploy/backend/build-info.json
```

Frontend만 빌드:

```
# 환경변수 확인 필수! (fe/.env 또는 export)
# Common이 먼저 빌드되어 있어야 함!
npm run build:server:fe

# 확인
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/frontend/
```

개별 빌드 시나리오 예시:

변경 내용	빌드 명령	이유
BE 로직만 수정	`build:server:be`	FE/Common 변경 없음
FE UI만 수정	`build:server:fe`	BE/Common 변경 없음
Common만 수정	`build:server:common`	빠른 빌드
Common + BE	`build:server:common` → `build:server:be`	순서 준수
전체 수정	`build:server`	안전한 방법

1.6 배포 (단일 서버)

> **실행 계정**: `iitp-plf` (운영 계정)

1.6.1 전체 배포 (기본, 권장)

배포 스크립트를 사용하면 모든 것이 자동으로 처리됩니다!

```
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 배포 (Common + Backend + Frontend)
npm run deploy:server
```

1) 스크립트에서 자동으로 처리:

- 단일 서버 환경 자동 감지 (localhost)
- rsync로 안전한 배포 (.env, node_modules, logs 자동 제외)
- 파일/디렉토리 권한 자동 설정 (755/644)
- Common → `/var/www/iitp-dabt-platform/packages/common/`

- Backend → `/var/www/iitp-dabt-platform/be/`
- Frontend → `/var/www/iitp-dabt-platform/fe/`
- 버전 정보 자동 출력

배포 후 자동으로 표시되는 정보:

```
서버용 전체 배포 완료!
배포된 서비스:
  Backend: localhost:/var/www/iitp-dabt-platform/be
  Frontend: localhost:/var/www/iitp-dabt-platform/fe
```

> **참고**:
>
> - Backend `.env` 파일은 스크립트가 자동으로 보존 (덮어쓰지 않음)
> - `node_modules/` 는 실행 서버에서 별도 설치 필요

2) 운영 스크립트 배포 (최초 1회 필수):

```
# 운영 관리 스크립트 배포
npm run deploy:server:ops
```

배포되는 항목:

- `start-server-be.js` : Backend PM2 시작
- `restart-server-be.js` : Backend PM2 재시작
- `restart-server-fe.js` : Nginx reload (Frontend 재배포 후)
- `stop-server-be.js` : Backend PM2 중지
- `package.json` : npm run 별칭 제공

배포 위치:

- 스크립트: `/var/www/iitp-dabt-platform/script/`
- package.json: `/var/www/iitp-dabt-platform/package.json`
- **최초 설치 시 (1회 필수)**
- start/restart/stop 스크립트가 수정되었을 때
- 일반적인 코드 배포 시에는 **불필요**

배포 후 사용 예:

```
cd /var/www/iitp-dabt-platform

# npm run 별칭 사용 (편리함)
npm run restart:server:be
npm run restart:server:fe

# 또는 직접 실행
node script/restart-server-be.js
node script/restart-server-fe.js
```

> **참고**: `npm run deploy:server` 는 BE/FE/Common 코드만 배포하며, 운영 스크립트는 포함하지 않습니다.

1.6.2 개별 배포 (옵션)

1) Common만 배포: 권장

```
cd /home/iitp-plf/iitp-dabt-platform/source

# 스크립트 사용 (단일 서버 + 서버 분리 모두 지원)
npm run deploy:server:common

# 후속 조치:
npm run restart:server:be  # BE 재시작 필수
# FE는 경우에 따라 재빌드 (아래 표 참조)
```

2) Backend만 배포: 권장

```
cd /home/iitp-plf/iitp-dabt-platform/source

# 스크립트 사용 (단일 서버 + 서버 분리 모두 지원)
npm run deploy:server:be

# 후속 조치: Backend 재시작 (자동으로 의존성 설치됨)
npm run restart:server:be
# → 자동으로 npm install --omit=dev 실행
# → 자동으로 PM2 restart 실행
```

의존성 설치 관련:

- **자동 방식 (권장)**: `npm run restart:server:be` 실행 시 자동으로 의존성 설치

- **수동 방식 (필요 시)**:

```
cd /var/www/iitp-dabt-platform/be
npm install --production  # 또는 --omit=dev
pm2 restart iitp-dabt-plf-be
```

3) Frontend만 배포: 권장

```
cd /home/iitp-plf/iitp-dabt-platform/source

# 스크립트 사용 (단일 서버 + 서버 분리 모두 지원)
npm run deploy:server:fe

# 후속 조치: Frontend 재시작 (Nginx reload만)
npm run restart:server:fe
# → Nginx 설정 테스트
# → Nginx reload (무중단 재시작)
```

> **Frontend는 의존성 설치 불필요**: 빌드된 정적 파일만 배포되므로 `npm install` 이 필요 없습니다.

개별 배포 시나리오 예시:

변경 내용	빌드	배포	후속 조치
BE API만 수정	`build:server:be`	BE만 배포	`restart:server:be`
FE UI만 수정	`build:server:fe`	FE만 배포	`restart:server:fe`
Common 타입만	`build:server:common`	Common만 배포	`restart:server:be`
Common 값 변경	`build:server:common` → `build:server:fe`	전체 배포	전체 재시작
전체 수정	`build:server`	전체 배포	전체 재시작

1.7 Backend 실행 환경 설정

1.7.1 자동 방식 (권장)

Backend 시작/재시작 스크립트가 **자동으로 의존성을 설치**합니다:

```
# 방법 1: npm run 사용 (최초 배포 후)
cd /var/www/iitp-dabt-platform
npm run start:server:be
# → 자동으로 npm install --omit=dev 실행

# 방법 2: 직접 실행
cd /var/www/iitp-dabt-platform
node script/start-server-be.js
# → 자동으로 npm install --omit=dev 실행
```

스크립트가 자동으로 처리하는 것들:

- 버전 정보 출력 (Backend, Common, 빌드 시간)
- 의존성 자동 설치 (`npm install --omit=dev`)
- PM2로 서버 시작

1.7.2 수동 방식 (필요 시)

스크립트를 사용하지 않고 직접 설치하려면:

```
# Backend 디렉토리로 이동
cd /var/www/iitp-dabt-platform/be

# 의존성 설치 (프로덕션 모드)
npm install --production
# 또는
npm install --omit=dev

# @iitp-dabt-platform/common 심볼릭 링크 확인
ls -la node_modules/@iitp-dabt-platform/common
# → ../../../packages/common을 가리켜야 함

# .env 파일 존재 확인
ls -la .env

# 로그 디렉토리 생성
mkdir -p logs

# PM2로 직접 시작
pm2 start dist/server.js --name iitp-dabt-plf-be
```

> **Frontend는 의존성 설치 불필요**: Frontend는 빌드된 정적 파일(HTML/CSS/JS)이므로 `npm install` 이 필요 없습니다. Nginx가 직접 제공합니다.

1.8 Nginx 설정 (루트 경로)

> **실행 계정**: `root` 또는 sudo 권한이 있는 계정

1.8.1 Step 1: Nginx 설정 파일 생성

```
# conf.d에 설정 파일 생성 (.conf 확장자 필수)
sudo vi /etc/nginx/conf.d/iitp-dabt-platform.conf
```

내용:

```
# Backend upstream
upstream iitp_dabt_platform_backend {
    server 127.0.0.1:33000;
    keepalive 32;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;  # 실제 도메인으로 변경 가능

    root /var/www/iitp-dabt-platform/fe/dist;
    index index.html;

    # ========================
    # [1] API 프록시
    # ========================
    location /api/ {
        proxy_pass http://iitp_dabt_platform_backend/api/;
        proxy_http_version 1.1;
        proxy_read_timeout 120s;
        proxy_send_timeout 120s;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        client_max_body_size 20m;
    }

    # ========================
    # [2] Frontend 정적 자산
    # ========================
    location /assets/ {
        alias /var/www/iitp-dabt-platform/fe/dist/assets/;
        try_files $uri =404;
        expires 7d;
        add_header Cache-Control "public, max-age=604800";
    }

    # ========================
    # [3] SPA Fallback
    # ========================
    location / {
```

```
        try_files $uri $uri/ /index.html;
    }

    # ========================
    # [4] 보안 헤더
    # ========================
    add_header X-Frame-Options SAMEORIGIN always;
    add_header X-Content-Type-Options nosniff always;
    add_header Referrer-Policy strict-origin-when-cross-origin always;
}
```

1.8.2 Step 2: 설정 검증 및 적용

```
# 기존 default 설정 처리
# nginx.conf가 sites-enabled/default를 직접 참조하는 경우가 있어 확인 필요
sudo cat /etc/nginx/nginx.conf | grep -n "sites-enabled"

# Case 1: 파일이 존재하고 충돌하는 경우
ls -la /etc/nginx/sites-enabled/default
sudo rm -f /etc/nginx/sites-enabled/default

# Case 2: 파일이 없는데 nginx.conf가 직접 참조하는 경우
# 에러: open() "/etc/nginx/sites-enabled/default" failed (2: No such file or directory)
# 해결: 빈 파일 생성 또는 nginx.conf 수정
sudo touch /etc/nginx/sites-enabled/default
# 또는
# sudo vi /etc/nginx/nginx.conf
# → include /etc/nginx/sites-enabled/default; 라인을 주석 처리 또는 삭제

# 설정 파일 문법 테스트
sudo nginx -t

# 성공하면 실제 충돌하는 파일 제거
sudo rm -f /etc/nginx/sites-enabled/default

# 재확인
sudo nginx -t

# Nginx 재시작
sudo systemctl restart nginx

# 상태 확인
sudo systemctl status nginx

# 재부팅 후 자동 시작 확인
sudo systemctl is-enabled nginx
# → enabled 출력되어야 함
```

> **conf.d 방식의 장점**:
>
> - 파일 생성만으로 자동 적용 (심볼릭 링크 불필요)
> - `/etc/nginx/nginx.conf` 에서 `include /etc/nginx/conf.d/*.conf;` 로 자동 로드
> - 재부팅 후에도 자동으로 유지됨

> **sites-enabled/default 관련 주의사항**:
>
> - Nginx는 `conf.d/*.conf` 와 `sites-enabled/*` **둘 다** 로드합니다
> - 기본 설치 시 `sites-enabled/default` 도 `listen 80 default_server;` 를 사용합니다
> - **우리 설정과 충돌하므로 제거 필요합니다**
>
> **만약 "No such file or directory" 에러가 발생하면**:
>
> - 일부 시스템의 `nginx.conf` 는 특정 파일을 직접 참조합니다
> - `sudo cat /etc/nginx/nginx.conf | grep -n default` 실행
> - `include /etc/nginx/sites-enabled/default;` 같은 라인이 있으면:
>   - 해당 라인을 주석 처리: `# include /etc/nginx/sites-enabled/default;`
>   - 또는 와일드카드로 변경: `include /etc/nginx/sites-enabled/*;`

1.9 서비스 시작 및 자동 시작 설정

> **실행 계정**: `iitp-plf` (운영 계정)

1.9.1 Backend 시작 (PM2)

Step 1: PM2로 Backend 시작

```
cd /var/www/iitp-dabt-platform/be

# PM2로 시작
pm2 start dist/server.js --name iitp-dabt-plf-be

# 상태 확인
pm2 list
pm2 logs iitp-dabt-plf-be --lines 50

# 헬스체크
curl http://localhost:33000/api/v1/health
```

Step 2: 재부팅 후 자동 시작 설정 (필수)

```
# 1. PM2 startup 스크립트 생성 (iitp-plf 계정에서 실행)
pm2 startup

# 2. 출력된 명령어를 복사하여 실행 (sudo 포함)
# 예시 출력:
# sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u iitp-plf --h
# → 위 명령어를 그대로 복사해서 실행

# 3. 현재 PM2 프로세스 목록 저장
pm2 save

# 4. systemd 서비스 상태 확인
sudo systemctl status pm2-iitp-plf
# → active (exited) 상태여야 함

# 5. 재부팅 후 자동 시작 확인
sudo systemctl is-enabled pm2-iitp-plf
# → enabled 출력되어야 함
```

> **중요**:
>
> - `pm2 startup` 은 **iitp-plf 계정**에서 실행해야 합니다
> - 출력된 sudo 명령어를 반드시 실행해야 systemd 서비스가 등록됩니다
> - `pm2 save` 를 실행해야 현재 프로세스가 저장됩니다
> - 이 설정 후에는 서버 재부팅 시 자동으로 Backend가 시작됩니다

기대 출력:

```
{"status":"healthy","timestamp":"2024-01-01T00:00:00.000Z"}
```

1.9.2 Frontend 확인

```
# 정적 파일 확인
ls -la /var/www/iitp-dabt-platform/fe/

# 브라우저 접속 테스트
curl -I http://localhost/
```

1.10 검증

> **실행 계정**: `iitp-plf` (운영 계정) 또는 모든 계정

1.10.1 기본 검증

```
# Backend 헬스체크
curl http://localhost:33000/api/v1/health
curl http://localhost/api/v1/health  # Nginx 경유

# Backend 버전 확인
curl http://localhost:33000/api/v1/version

# Frontend 접속
curl -I http://localhost/

# PM2 상태
pm2 list

# Nginx 상태
sudo systemctl status nginx

# Backend 로그
pm2 logs iitp-dabt-plf-be --lines 50
```

성공 확인:

- Backend 헬스체크: HTTP 200, `{"status":"healthy"}`
- Frontend: HTTP 200, HTML 응답
- PM2: `iitp-dabt-plf-be` 상태 `online`
- Nginx: `active (running)` 상태

1.10.2 재부팅 후 자동 시작 검증 (필수)

```
# 시스템 재부팅
sudo reboot

# 재부팅 후 로그인하여 확인

# 1. Nginx 자동 시작 확인
sudo systemctl status nginx
# → active (running) 상태여야 함

# 2. PM2 자동 시작 확인
pm2 list
# → iitp-dabt-plf-be가 online 상태여야 함

# 3. Backend 헬스체크
curl http://localhost:33000/api/v1/health
curl http://localhost/api/v1/health

# 4. Frontend 접속
curl -I http://localhost/
```

재부팅 후 성공 확인:

- Nginx: `active (running)` - 자동 시작됨
- PM2: `iitp-dabt-plf-be` 상태 `online` - 자동 시작됨
- Backend 헬스체크: HTTP 200
- Frontend: HTTP 200

> **재부팅 후 자동 시작이 안 되면**:
>
> - Nginx: `sudo systemctl enable nginx` 실행
> - PM2: 섹션 1.9 Step 2의 `pm2 startup` + `pm2 save` 재실행

2. 초기 설치 - 서버 분리 환경

> **전제조건**: 빌드 서버 ≠ 실행 서버 (서버가 물리적으로 분리)

2.1 빌드 서버 설정

2.1.1 빌드 서버 기본 세팅

```
# Ubuntu 20.04+ 기준 (빌드 서버)
sudo apt update && sudo apt upgrade -y

# 필수 패키지 설치
sudo apt install -y git curl build-essential rsync
```

Node.js 설치 (아래 중 하나 선택):

섹션 1.0의 Node.js 설치 방법 참조 (**프로덕션: NodeSource 권장**, snap, nvm 중 선택)

```
# 설치 후 버전 확인
node -v  # v22.x.x
npm -v   # 9.x.x 이상
```

2.1.2 계정 및 디렉토리 생성

```
# iitp-plf 사용자 생성
sudo useradd -m -s /bin/bash iitp-plf
sudo passwd iitp-plf

# 디렉토리 생성
sudo mkdir -p /home/iitp-plf/iitp-dabt-platform/source
sudo mkdir -p /home/iitp-plf/iitp-dabt-platform/deploy
sudo chown -R iitp-plf:iitp-plf /home/iitp-plf/iitp-dabt-platform
```

2.1.3 SSH 키 설정 (rsync용)

```
# iitp-plf 사용자로 전환
sudo -iu iitp-plf

# SSH 키 생성
ssh-keygen -t rsa -b 4096 -C "iitp-plf@build-server"
# Enter 3번 (비밀번호 없이)

# 공개키 확인
cat ~/.ssh/id_rsa.pub
```

실행 서버에 공개키 등록:

```
# 실행 서버에서 실행
# (빌드 서버의 공개키를 복사하여 실행 서버에 추가)
echo "ssh-rsa AAAA..." >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
chmod 700 ~/.ssh
```

연결 테스트:

```
# 빌드 서버에서 실행
ssh iitp-plf@실행서버IP
# 비밀번호 없이 접속되면 성공
exit
```

2.1.4 프로젝트 클론 및 설정

```
# iitp-plf 사용자로 (빌드 서버)
cd /home/iitp-plf/iitp-dabt-platform/source
git clone https://github.com/sweetk-dev/06-IITP-DABT-Platform.git .

# 전체 패키지 설치
npm install
```

2.1.5 환경변수 설정 (빌드 서버)

Frontend 빌드 환경변수:

```
cd /home/iitp-plf/iitp-dabt-platform/source/fe
cp env.sample .env
vi .env
```

내용 (단독 설치, 실행 서버 주소 사용):

```
VITE_PORT=5173
VITE_BASE=/
VITE_API_BASE_URL=http://실행서버IP
VITE_API_TIMEOUT=10000
VITE_API_DATA_PREVIEW_LIMIT=10
VITE_VISUAL_TOOL=http://실행서버IP:visual-tool-port/
VITE_EMPLOYMENT_SITE_URL=https://www.ablejob.co.kr/
VITE_OPEN_API_CENTER_URL=http://실행서버IP/admin/
VITE_OPEN_API_CENTER_ABOUT_URL=http://실행서버IP/admin/about
```

빌드 스크립트 환경변수:

```
cd /home/iitp-plf/iitp-dabt-platform/source
cp script/server/env.sample.build-server script/server/.env
vi script/server/.env
```

내용:

```
GIT_REPO_URL=https://github.com/sweetk-dev/06-IITP-DABT-Platform.git
GIT_BRANCH=main
SOURCE_PATH=/home/iitp-plf/iitp-dabt-platform/source
DEPLOY_PATH=/home/iitp-plf/iitp-dabt-platform/deploy

# 실행 서버 정보
PROD_SERVER_HOST=실행서버IP
PROD_SERVER_USER=iitp-plf
PROD_SERVER_PORT=22
PROD_BE_PATH=/var/www/iitp-dabt-platform/be
PROD_FE_PATH=/var/www/iitp-dabt-platform/fe
OPS_SCRIPT_PATH=/var/www/iitp-dabt-platform/script

NODE_ENV=production
NPM_CONFIG_PRODUCTION=true
```

2.2 실행 서버 설정

2.2.1 실행 서버 기본 세팅

```
# Ubuntu 20.04+ 기준 (실행 서버)
sudo apt update && sudo apt upgrade -y

# 필수 패키지 설치
sudo apt install -y curl nginx
```

Node.js 설치 (아래 중 하나 선택):

섹션 1.0의 Node.js 설치 방법 참조 (**프로덕션: NodeSource 권장**, snap, nvm 중 선택)

```
# PM2 글로벌 설치
sudo npm install -g pm2

# PM2 시작 스크립트 등록 (부팅 시 자동 시작)
pm2 startup
# 출력되는 명령어 실행 (sudo env PATH=... 형태)

# PostgreSQL 설치
sudo apt install -y postgresql postgresql-contrib

# 버전 확인
node -v  # v22.x.x
npm -v   # 9.x.x 이상
pm2 -v
psql --version
```

2.2.2 계정 및 디렉토리 생성

```
# iitp-plf 사용자 생성 (실행 서버)
sudo useradd -m -s /bin/bash iitp-plf
sudo passwd iitp-plf

# 실행 디렉토리 생성
sudo mkdir -p /var/www/iitp-dabt-platform/be
sudo mkdir -p /var/www/iitp-dabt-platform/fe
sudo mkdir -p /var/www/iitp-dabt-platform/script
sudo mkdir -p /var/www/iitp-dabt-platform/packages/common
sudo chown -R iitp-plf:iitp-plf /var/www/iitp-dabt-platform
```

2.2.3 데이터베이스 설정

섹션 1.2와 동일

2.2.4 Backend 환경변수 설정

```
# 실행 서버
sudo vi /var/www/iitp-dabt-platform/be/.env
```

내용: 섹션 1.4.1과 동일

2.3 빌드 및 배포

2.3.1 빌드 (빌드 서버)

전체 빌드 (기본, 권장):

```
# 빌드 서버에서 실행
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 빌드
npm run build:server

# 빌드 확인
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/backend/
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/frontend/
ls -la /home/iitp-plf/iitp-dabt-platform/deploy/common/
```

개별 빌드 (옵션):

- 섹션 1.5.2 개별 빌드 참조
- Common만, BE만, FE만 빌드 방법 및 시나리오

2.3.2 배포 (빌드 서버 → 실행 서버)

방법 1: 전체 배포 스크립트 (권장)

```
# 빌드 서버에서 실행
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 배포
npm run deploy:server

# 운영 스크립트 배포 (최초 1회)
npm run deploy:server:ops
```

> **운영 스크립트 배포 상세 설명**: 섹션 1.6.1 운영 스크립트 배포 참조

방법 2: 개별 배포 스크립트 (빠른 배포)

```
# BE만 배포
npm run deploy:server:be
# 후속: npm run restart:server:be

# FE만 배포
npm run deploy:server:fe
# 후속: npm run restart:server:fe

# Common만 배포
npm run deploy:server:common
# 후속: npm run restart:server:be (FE는 경우에 따라)
```

개별 배포 상세: 섹션 1.6.2 참조

방법 3: 수동 rsync (상세 제어 필요 시)

```
# 빌드 서버에서 실행

# Backend 배포
rsync -avz --delete \
  --exclude='node_modules' --exclude='.env' \
  /home/iitp-plf/iitp-dabt-platform/deploy/backend/ \
  iitp-plf@실행서버IP:/var/www/iitp-dabt-platform/be/

# Frontend 배포
rsync -avz --delete \
  /home/iitp-plf/iitp-dabt-platform/deploy/frontend/ \
  iitp-plf@실행서버IP:/var/www/iitp-dabt-platform/fe/

# Common 패키지 배포
rsync -avz --delete \
  /home/iitp-plf/iitp-dabt-platform/deploy/common/ \
  iitp-plf@실행서버IP:/var/www/iitp-dabt-platform/packages/common/

# 운영 스크립트 배포
rsync -avz \
  script/server/*.js script/server/.env \
  iitp-plf@실행서버IP:/var/www/iitp-dabt-platform/script/
```

개별 배포 옵션 및 후속 조치: 섹션 1.6.2 참조

2.4 실행 서버에서 Backend 실행 준비

```
# 실행 서버에서 실행
cd /var/www/iitp-dabt-platform/be

# 의존성 설치
npm install --production

# 심볼릭 링크 확인
ls -la node_modules/@iitp-dabt-platform/common

# .env 파일 확인
cat .env

# 로그 디렉토리 생성
mkdir -p logs
```

2.5 Nginx 설정

섹션 1.8과 동일 (실행 서버에서 설정)

2.6 서비스 시작

섹션 1.9와 동일 (실행 서버에서 실행)

2.7 검증

섹션 1.10과 동일 (실행 서버에서 확인)

3. 업데이트 배포 (일상 운영)

> **전제조건**: 섹션 1 또는 2의 초기 설치가 완료된 상태
>
> **실행 계정**: `iitp-plf` (운영 계정)

3.1 사전 확인 및 백업

```
# 현재 버전 확인
curl http://localhost:33000/api/v1/version

# 서비스 상태 확인
pm2 list
pm2 logs iitp-dabt-plf-be --lines 50

# 데이터베이스 백업 (선택사항)
pg_dump -U iitp_platform_user iitp_dabt > backup_$(date +%Y%m%d_%H%M%S).sql

# 현재 코드 백업 (선택사항)
cd /home/iitp-plf/iitp-dabt-platform
tar -czf source_backup_$(date +%Y%m%d_%H%M%S).tar.gz source/
```

3.2 소스 업데이트

단일 서버:

```
cd /home/iitp-plf/iitp-dabt-platform/source
git fetch origin
git pull origin main

# 변경사항 확인
git log -5 --oneline
git diff HEAD@{1} HEAD
```

빌드 서버 (서버 분리):

```
# 빌드 서버에서 실행
cd /home/iitp-plf/iitp-dabt-platform/source
git fetch origin
git pull origin main

# 변경사항 확인
git log -5 --oneline
```

3.3 의존성 업데이트 확인

```
# package.json 변경 확인
git diff HEAD@{1} HEAD -- package.json be/package.json fe/package.json packages/common/package.j

# 변경이 있으면 재설치
npm install
```

3.4 빌드

단일 서버:

```
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 빌드 (권장)
npm run build:server
```

개별 빌드 옵션:

- 섹션 1.5.2 개별 빌드 참조
- BE만, FE만, Common만 빌드 방법 및 주의사항

빌드 서버 (서버 분리):

```
# 빌드 서버에서 실행
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 빌드 (권장)
npm run build:server
```

개별 빌드 옵션: 섹션 1.5.2 참조

3.5 배포

> **실행 계정**: `iitp-plf` (운영 계정)

단일 서버:

```
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 배포 (권장)
npm run deploy:server

# 또는 개별 배포
npm run deploy:server:be      # Backend만
npm run deploy:server:fe      # Frontend만
npm run deploy:server:common  # Common만
```

스크립트가 자동으로 처리:

- rsync로 안전한 배포 (node_modules, .env, logs 자동 제외)
- 파일 권한 자동 설정
- 버전 정보 자동 출력

수동 배포 (비권장):

```
# Backend
rsync -av --delete \
  --exclude='node_modules' --exclude='.env' --exclude='logs' \
  /home/iitp-plf/iitp-dabt-platform/deploy/backend/ \
  /var/www/iitp-dabt-platform/be/

# Frontend
rsync -av --delete \
  /home/iitp-plf/iitp-dabt-platform/deploy/frontend/ \
  /var/www/iitp-dabt-platform/fe/

# Common
rsync -av --delete \
  /home/iitp-plf/iitp-dabt-platform/deploy/common/ \
  /var/www/iitp-dabt-platform/packages/common/
```

상세 옵션: 섹션 1.6.2 개별 배포 참조

서버 분리:

```
# 빌드 서버에서 실행
cd /home/iitp-plf/iitp-dabt-platform/source

# 전체 배포 (권장)
npm run deploy:server

# 또는 개별 배포
npm run deploy:server:be      # Backend만
npm run deploy:server:fe      # Frontend만
npm run deploy:server:common  # Common만
```

상세 옵션: 섹션 1.6.2 참조

3.6 서비스 재시작

> **전제 조건**:
>
> - **PM2 자동 시작이 이미 설정되어 있어야 합니다** (초기 설치 시 설정됨)
> - 설정 확인: `sudo systemctl status pm2-iitp-plf`
> - 미설정 시: 섹션 1.9 Step 2 참조하여 `pm2 startup` + `pm2 save` 실행

전체 재시작 (권장)

```
# Backend 재시작
npm run restart:server:be

# Frontend 재시작 (Nginx reload)
npm run restart:server:fe

# 로그 확인
pm2 logs iitp-dabt-plf-be --lines 50
```

개별 재시작

Backend만:

```
npm run restart:server:be
# 또는: pm2 restart iitp-dabt-plf-be
```

Frontend만:

```
npm run restart:server:fe
# 또는: sudo nginx -t && sudo systemctl reload nginx
```

재시작 필요 시나리오:

배포 내용	Backend 재시작	Frontend 재시작
BE만 배포	**필수**	불필요
FE만 배포	불필요	**필수(Nginx reload)**
Common만 배포	**필수**	조건부 (섹션 1.6.2 참조)
전체 배포	**필수**	**필수**

3.7 검증

```
# 버전 확인 (변경되었는지 확인)
curl http://localhost:33000/api/v1/version

# 헬스체크
curl http://localhost:33000/api/v1/health
curl http://localhost/api/v1/health

# PM2 상태
pm2 list
pm2 logs iitp-dabt-plf-be --lines 20

# Frontend 접속 테스트
curl -I http://localhost/

# 브라우저에서 실제 접속 테스트
```

3.8 롤백 (문제 발생 시)

```
# Git 롤백 (빌드 서버 또는 단일 서버)
cd /home/iitp-plf/iitp-dabt-platform/source
git log --oneline -10
git reset --hard <이전_커밋_해시>

# 재빌드
npm run build:server

# 재배포 (섹션 3.5 참조)

# 서비스 재시작
pm2 restart iitp-dabt-plf-be

# DB 롤백 (필요 시)
psql -U iitp_platform_user iitp_dabt < backup_YYYYMMDD_HHMMSS.sql
```

4. 서비스 운영 관리

4.1 서비스 관리

PM2 명령어

```
# 상태 확인
pm2 list

# 로그 확인
pm2 logs iitp-dabt-plf-be
pm2 logs iitp-dabt-plf-be --lines 100

# 재시작
pm2 restart iitp-dabt-plf-be

# 중지
pm2 stop iitp-dabt-plf-be

# 시작
pm2 start iitp-dabt-plf-be

# 삭제
pm2 delete iitp-dabt-plf-be

# 메모리/CPU 모니터링
pm2 monit
```

Nginx 관리

```
# 설정 테스트
sudo nginx -t

# 재시작
sudo systemctl restart nginx

# Reload (무중단)
sudo systemctl reload nginx

# 상태 확인
sudo systemctl status nginx

# 로그 확인
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log
```

4.2 로그 관리

```
# Backend 로그
pm2 logs iitp-dabt-plf-be

# Backend 파일 로그
tail -f /var/www/iitp-dabt-platform/be/logs/combined.log
tail -f /var/www/iitp-dabt-platform/be/logs/error.log

# Nginx 로그
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log

# PostgreSQL 로그
sudo tail -f /var/log/postgresql/postgresql-*-main.log
```

4.3 모니터링

```
# 시스템 리소스
htop
free -h
df -h

# 네트워크
sudo netstat -tlnp | grep 33000
sudo ss -tlnp | grep 33000

# 프로세스
ps aux | grep node
ps aux | grep nginx

# PM2 모니터링
pm2 monit
```

4.4 데이터베이스 관리

```
# 백업
pg_dump -U iitp_platform_user iitp_dabt > backup_$(date +%Y%m%d_%H%M%S).sql

# 복원
psql -U iitp_platform_user iitp_dabt < backup_YYYYMMDD_HHMMSS.sql

# 접속
psql -U iitp_platform_user -d iitp_dabt

# 연결 확인
psql -U iitp_platform_user -d iitp_dabt -c "SELECT 1;"
```