

[포팅 메뉴얼]

날짜	@2024년 11월 19일
----	----------------

1) 프로젝트 및 환경 정보

- 백엔드
- 프론트
- 공통
- 포트 정보
- 공통 사항

2) 파일 시스템 구조

3) 클론 후 빌드 및 실행 방법

- 프론트엔드 빌드
- 백엔드 빌드 및 실행 방법

3) 시연 시나리오

1) 프로젝트 및 환경 정보

백엔드

- Python 3.11.9
- Fast API 0.33.1
- Cuda 11.8
- Pytorch

프론트

- React
- TypeScript
- Emotion
- Zustand
- Axios
- TanStack Query
- yarn berry

공통

- Docker 27.3.1

포트 정보

Container Name	Port Number
fast api (main)	5000
fast api (deploy)	8088
jenkins	9090
랜딩페이지	3000

공통 사항

- URL
 - Docker hub
 - <https://hub.docker.com/r/aiblockeditor/ai-block-editor/tags>
 - 랜딩 페이지
 - <http://k11a305.p.ssafy.io:3000/>
- Docker
 - 27.3.1

2) 파일 시스템 구조

```
/able
|
└─ /{version}
    |
    └─ /blocks                                # 블록 정보 저장 폴더
        └─ /layer                            # 블록 타입별 폴더
            └─ block1.json                  # 블록 메타데이터
            └─ block2.json
```

```

└─ /data                                     # 사용자 데이터 저장 폴더
  └─ /deploy                                # 배포 관리 폴더
    │ metadata.json                         # 배포 메타데이터
    └─ {path_name}.json                    # API 정보
  │
  └─ /devices                               # 사용자 디바이스 관리 폴더
    │ device_A.json                        # 사용자 디바이스_A 파일
    │
    └─ /projects                            # 프로젝트 관리 폴더
      └─ /project_A                         # 프로젝트 A 데이터 폴더
        │ metadata.json                     # 프로젝트 메타데이터
        │ thumbnail.jpg                     # 썸네일 이미지
        │ block_graph.json                  # 현재 작성중인 모델 구성 요소 정보
        └─ /train_results                   # 학습 결과 관리 폴더
          └─ /result_1                      # 학습 결과 1
            │ metadata.json                 # 학습결과 메타데이터(데이터 경로, 상태)
            │ hyper_parameter.json          # 하이퍼 파라미터
            │ block_graph.json              # 모델 구성 요소 정보
            │ performance_metrics.json      # 성능 지표 테이블
            │                               │ f1_score.json                # f1-score
            │                               │ confusion_matrix.jpg         # 혼동 행렬 이미지
            │                               └─ transform_pipeline.pickle    # 데이터 전처리
          └─ /checkpoints                    # 에포크 관리 폴더
            └─ /train_best                    # train_best의 모델, valid_best, final도
              │ model.pth
              │ training_loss.json           # 학습 손실 정보
              │ └─ validation_loss.json      # 검증 손실 정보
              │ heatmap.jpg                 # 히트맵 이미지
              │ original.jpg
              │ analysis_result.json         # 분석 결과: 상위 클래스 3개와 점수
              └─ /feature_maps               # 피쳐맵 이미지 폴더
                │ layers.Ablock.jpg          # A 블록 피쳐맵 이미지
                └─ layers.Bblock.jpg         # B 블록 피쳐맵 이미지
              └─ /valid_best
                │ ... # train_best의 모델과 동일한 구조
              └─ /final
                │ ... # train_best의 모델과 동일한 구조
          └─ /epoch_10                       # 에포크 10
            └─ ... # train_best의 모델과 동일한 구조

```

3) 클론 후 빌드 및 실행 방법

프론트엔드 빌드

```

# 프론트엔드 코드를 빌드
cd frontend/
yarn build

# 빌드 후 생성된 dist 디렉터리 내의 파일을 백엔드 디렉터리로 복사
cp -r dist/. ../backend/able/static/.
```

백엔드 빌드 및 실행 방법

정보

- 포트 번호 : 5000

Dockerfile

```
FROM python:3.11.9

ENV APPLICATION_NAME=able
ENV VERSION=v1

ENV HOME=/app
ENV PORT=5000

WORKDIR $HOME

RUN apt-get update && apt-get install -y libgl1-mesa-glx && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt && rm requirements.txt
RUN pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

COPY setup ./setup
RUN python3 setup/install.py

COPY src ./src
COPY deploy_server ./deploy_server
COPY static ./static

EXPOSE $PORT


CMD ["uvicorn", "src.main:app", "--host", "0.0.0.0", "--port", "5000", "--workers", "3"]
```

[requirements.txt](#)

Docker 이미지 빌드 및 실행

```
docker build -t able -f Dockerfile
docker run -d --name able
  -p 5000:5000 \
  -p 8088:8088 \
  -v /home/ubuntu/able/v1/blocks:/app/able/v1/blocks \
  -v /home/ubuntu/able/v1/data/projects:/app/able/v1/data/projects \
  -e TZ=Asia/Seoul \
  -e PYTHONPATH=/app \
  --gpus all \
  able
```

3) 시연 시나리오

 [사용 가이드](#)