1. Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

1) 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전(IDE버전 포함) 기재

build.gradle에 자바 21로 되어있습니다.

나머지는 싸피에서 제공하는 모든 것들을 사용했습니다.

2) 빌드 시 사용되는 환경 변수 등의 내용 상세 기재

fe, be, ai 폴더에서 .env.example을 참고해 환경 변수를 주입하면 됩니다.

3) 배포 시 특이사항 기재

프론트, ai, 백엔드 환경변수 등록 후 Jenkins Execute Shell에 각각 명령어를 입력하면 됩니다.

fe

```
#!/bin/bash
set -xeuo pipefail
WORKSPACE=/var/jenkins_home/workspace/fe-dev
cd "$WORKSPACE"

DEV_CONTAINER_NAME="react-vite-app-dev"
DEV_IMAGE_NAME="react-vite-app:dev"
DEV_PORT="3001"

# 1) Jenkins Credentials에서 .env 파일 사용
ENV_FILE="$WORKSPACE/fe/.env"
mkdir -p "$WORKSPACE/fe"

echo "=== DEBUG: Jenkins Credentials ==="
echo "FE variable points to: $FE"
ls -la "$FE" || echo "Credential file not found"

# Jenkins secret file을 작업 디렉터리로 복사
if [ -f "$FE" ]; then
    cp "$FE" "$ENV_FILE"
    echo "Copied Jenkins credential file to $ENV_FILE"
    chmod 600 "$ENV_FILE"
    echo "Generated .env file content:"
    sed 's/=.*/=***/' "$ENV_FILE"  # 보안상 값은 마스킹해서 출력
else
    echo "Warning: Jenkins credential file not found, creating empty .env"
```

```
    touch "$ENV_FILE"
    chmod 600 "$ENV_FILE"
fi

# 2) 현재 상태 출력 (detached HEAD 대비)
echo "PWD: $(pwd)"
echo "Branch: $(git branch --show-current || true)"

# 3) 필수 폴더 체크
if [ ! -d "fe" ]; then
  echo "Error: fe directory not found!"
  exit 1
fi

# 4) (선택) vite.config.ts 패치
cp fe/vite.config.ts fe/vite.config.ts.backup || true
cat > fe/vite.config.ts << 'EOF'
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
export default defineConfig({
  plugins: [react()],
  preview: {
    host: '0.0.0.0',
    port: 4173,
    strictPort: true,
    allowedHosts: ['localhost','127.0.0.1','j13a409.p.ssafy.io','.p.ssafy.io']
  }
})
EOF

# 5) 이전 컨테이너 정리 (없어도 실패하지 않게)
docker stop $DEV_CONTAINER_NAME || true
docker rm $DEV_CONTAINER_NAME || true
docker rmi $DEV_IMAGE_NAME || true

# 6) Docker 빌드 (※ 이 시점에 fe/.env 가 존재해야 Vite가 빌드시 읽습니다)
docker build -t $DEV_IMAGE_NAME ./fe

# 7) 컨테이너 실행 (※ SPA라 --env-file 불필요)
```

```
docker run -d --name $DEV_CONTAINER_NAME -p $DEV_PORT:4173 $DEV_IMAGE_NAME

# 8) 확인
docker ps | grep $DEV_CONTAINER_NAME || true
docker logs $DEV_CONTAINER_NAME --tail 50 || true

echo "Dev application is running:"
echo "- Local:       http://localhost:$DEV_PORT"
echo "- External: http://j13a409.p.ssafy.io:$DEV_PORT"
```

ai

```
ENV_FILE=$WORKSPACE/ai/.env     # .env 실제 경로와 동일하게

cat "$AI" > "$ENV_FILE"
chmod 600 "$ENV_FILE"

cd ai

make re
```

be

```
#!/bin/bash

WORKSPACE=/var/jenkins_home/workspace/be
IMAGE_NAME=spring:latest
CONTAINER_NAME=spring
ENV_FILE=$WORKSPACE/be/.env     # .env 실제 경로와 동일하게

cd "$WORKSPACE"

# 🔎 Jenkins가 제공하는 이전/현재 커밋 비교
PREV_COMMIT=${GIT_PREVIOUS_SUCCESSFUL_COMMIT:-}
CUR_COMMIT=${GIT_COMMIT:-}

if [ -z "$PREV_COMMIT" ]; then
    echo "[WARN] 이전 빌드 커밋을 찾을 수 없음 → 첫 빌드로 간주하고 배포 진행"
else
    if git diff --quiet "$PREV_COMMIT" "$CUR_COMMIT" -- be/; then
```

```
            echo "[INFO] be 폴더에 변경 없음 → 배포 스킵"
            exit 0
        fi
fi

echo "[INFO] be 폴더에 변경 감지됨 → 빌드 & 배포 진행"

#⑴ Jenkins Secret File(BACK) 내용을 .env로 생성
if [ -z "$BACK" ]; then
        echo "ERROR: BACK environment variable is empty"
        exit 1
fi

# 실제 내용으로 복사
cat "$BACK" > "$ENV_FILE"
chmod 600 "$ENV_FILE"

#⑵ Docker 빌드 (Dockerfile은 be 폴더 안에 있음)
docker build -t $IMAGE_NAME $WORKSPACE/be || exit 1

#⑶ 기존 컨테이너 삭제
docker rm -f $CONTAINER_NAME || true

#⑷ 새 컨테이너 실행
docker run -d ₩
        --name $CONTAINER_NAME ₩
        --env-file "$ENV_FILE" ₩
        -p 8080:8080 ₩
        $IMAGE_NAME
```

2. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서

싸피 로그인, 금융망 API와 싸피 GMS를 사용했습니다.

3. DB 덤프 파일 최신본

dummy_card_transactions_18m.csv 참고

4. 시연 시나리오

1. 로그인 슈퍼앱에 대해 설명할 때 랜딩 전체 보여주기
2. 슈퍼앱 로그인
3. 메인 장독대 페이지로 들어감 (누수 없음)
4. 9월 장독대에서
4-1. 주거/통신 누수 (호버)
4-2. 경조사/회비 누수
4-3. 교통/차량 누수
4-4. 세 가지 누수 조절하며 물 양 달라지는거 보여주기
5. 콩쥐의 씀씀이 가서 라인 차트 (색 다른거) 설명할 때 호버
5-2. 표에서 호텔즈컴바인(9/14) -> 주거로 바꾸기
6. 두꺼비의 조언 9월 카드 보여준 뒤 (아까 장독대 3개 누수였음)
6-1. 8월 두꺼비의 조언 가서 교육, 마트, 카페 3개 보여주기
+7. 콩쥐의 곳간: 질문 들어오면 한 단계씩 보여주기