쉘 스크립트 기반 GitLab CI/CD

FastAPI + JS(Frontend) 자동 배포 매뉴얼

- GitLab repo:

  👉 https://lab.ssafy.com/s13-final/S13P31S303.git

- 배포: EC2에 user 권한(`ubuntu`, `ec2-user`)만 있음
- 인프라: **no docker / no jenkins**
- 동작: EC2에서 git pull후 bash형태로 배포 관리 → 변경 있으면 FE/BE 재배포

사전 준비

(1) GitLab 접근 토큰 등록 (PAT)

read_repository권한으로 Settings-Access tokens에서 토큰 발급 (Role reporter 이상)

```
cat >> ~/.netrc <<'EOF'
machine lab.ssafy.com
  login <token_name>
  password <token_password>   # read_repository 권한만 필요
EOF
chmod 600 ~/.netrc
```

---

(2) 폴더 구조 만들기

```
mkdir -p ~/apps/{repo,backend,frontend}
mkdir -p ~/apps/repo/work
mkdir -p ~/apps/backend/{releases,shared}
mkdir -p ~/apps/frontend/{releases,shared}
```

FastAPI systemd 서비스 유닛 만들기

```
sudo nano /etc/systemd/system/fastapi.service

[Unit]
Description=FastAPI Backend Service
After=network.target

[Service]
# 실행 계정
User=ubuntu
Group=ubuntu

# 코드가 있는 작업 디렉토리(항상 현재 릴리스 심볼릭 링크를 가리킴)
WorkingDirectory=/home/ubuntu/apps/backend/current

# 가상환경의 uvicorn 직접 실행 (main.py 안의 app 기준)
ExecStart=/home/ubuntu/apps/backend/shared/venv/bin/uvicorn main:app
--host 127.0.0.1 --port 8000 --workers 2

# 장애 복구
Restart=always
RestartSec=3

# 환경변수
Environment="PORT=8000"

[Install]
WantedBy=multi-user.target
```

적용/기동:

```
sudo systemctl daemon-reload
sudo systemctl enable fastapi
sudo systemctl start fastapi
sudo systemctl status fastapi
```

로그 확인:

```
sudo journalctl -u fastapi -f
```

배포 스크립트

> Repo 구조
>
> - BE 코드 경로: `S13P31S303/BE`
> - FE 코드 경로: `S13P31S303/FE`

---

(1) 백엔드 배포

```
sudo nano ~/apps/backend_deploy.sh
==========================================================
==========
#!/usr/bin/env bash
set -euo pipefail

REPO_WORK="$HOME/apps/repo/work"
SRC="$REPO_WORK/BE"                # ← BE 소스 경로
APP_DIR="$HOME/apps/backend"
RELEASES="$APP_DIR/releases"
SHARED="$APP_DIR/shared"
TS="$(date +%F_%H-%M-%S)"
NEW="$RELEASES/$TS"

# Python 3.11 명시
PYBIN="/usr/bin/python3.11"

# 소스 확인
[ -d "$SRC" ] || { echo "❌ BE source not found: $SRC"; exit 1; }
[ -x "$PYBIN" ] || { echo "❌ $PYBIN not found.  Install it with: sudo apt inst
all python3.11 python3.11-venv"; exit 1; }

# 새 릴리스 생성
mkdir -p "$NEW" "$SHARED"
rsync -a --delete "$SRC/" "$NEW/"
```

```
# venv & requirements (항상 3.11 기반)
if [ ! -d "$SHARED/venv" ]; then
  echo "▶ Creating venv using $PYBIN ..."
  "$PYBIN" -m venv "$SHARED/venv" || { echo "❌ venv creation failed"; e
xit 1; }
fi

"$SHARED/venv/bin/python" -m ensurepip --upgrade || true
"$SHARED/venv/bin/python" -m pip install --upgrade pip setuptools wheel
>/dev/null

if [ -f "$NEW/requirements.txt" ]; then
  echo "▶ Installing requirements.txt ..."
  "$SHARED/venv/bin/pip" install -r "$NEW/requirements.txt"
else
  echo "ℹ️  No requirements.txt found, skipping install."
fi

# .env 심볼릭 링크 연결 (shared/.env → NEW/.env)
if [ -f "$SHARED/.env" ]; then
  ln -sfn "$SHARED/.env" "$NEW/.env"
  echo "▶ Linked .env from shared to new release."
else
  echo "⚠️  No .env found in $SHARED — service may fail if required."
fi

# 릴리스 전환
ln -sfn "$NEW" "$APP_DIR/current"

# FastAPI 재시작 (systemd)
echo "▶ Restarting fastapi.service ..."
sudo systemctl restart fastapi
sudo systemctl status fastapi --no-pager -l | sed -n '1,10p' || true

# 오래된 릴리스 정리(최신 5개만 유지)
( cd "$RELEASES" && ls -1tr | head -n -5 | xargs -r rm -rf ) || true
```

```
echo "✅ BE deployed at $TS"
==========
sudo chmod +x ~/apps/backend_deploy.sh
```

---

(2) 프론트엔드 배포

```
sudo nano ~/apps/frontend_deploy.sh
==========
#!/usr/bin/env bash
set -euo pipefail

REPO_WORK="$HOME/apps/repo/work"
SRC="$REPO_WORK/FE"
APP_DIR="$HOME/apps/frontend"
RELEASES="$APP_DIR/releases"
TS="$(date +%F_%H-%M-%S)"
NEW="$RELEASES/$TS"

[ -d "$SRC" ] || { echo "FE source not found: $SRC"; exit 1; }

mkdir -p "$NEW"
rsync -a --delete "$SRC/" "$NEW/"
ln -sfn "$NEW" "$APP_DIR/current"

test -f "$APP_DIR/current/index.html" && echo "✅ FE deployed at $TS" ||
echo "⚠️ FE no index.html"
( cd "$RELEASES" && ls -1tr | head -n -5 | xargs -r rm -rf ) || true
==========
sudo chmod +x ~/apps/frontend_deploy.sh
```

.env 환경변수 주입

```
sudo nano ~/apps/backend/shared/.env

# 읽기 권한 최소화
sudo chmod 600 ~/apps/backend/shared/.env
```

폴링 스크립트 (develop 브랜치 자동 감지)

```
sudo nano ~/apps/poll_and_deploy.sh
==========================================================
==========
#!/usr/bin/env bash
set -euo pipefail

REPO_URL="https://lab.ssafy.com/s13-final/S13P31S303.git"
BRANCH="develop"
WORK="$HOME/apps/repo/work"

# 최초 클론
if [ ! -d "$WORK/.git" ]; then
  git clone --branch "$BRANCH" --depth 1 "$REPO_URL" "$WORK"
  changed=1
else
  BEFORE=$(git -C "$WORK" rev-parse HEAD)
  git -C "$WORK" fetch --depth 1 origin "$BRANCH"
  git -C "$WORK" switch -C "$BRANCH" origin/"$BRANCH"
  AFTER=$(git -C "$WORK" rev-parse HEAD)
  if [ "$BEFORE" != "$AFTER" ]; then changed=1; else changed=0; fi
fi

# 변경 시 FE/BE 동시 배포
if [ "$changed" -eq 1 ]; then
  echo "Changes detected on $BRANCH. Deploying..."
  "$HOME/apps/frontend_deploy.sh"
  "$HOME/apps/backend_deploy.sh"   # 내부에서 systemctl restart fastapi 수
행
```

```
else
  echo "No changes."
fi
============================================================
==========
sudo chmod +x ~/apps/poll_and_deploy.sh
```

> 나중에 main 브랜치로 전환 시:
>
> `BRANCH="develop"` → `main` 으로 바꿔주면 끝.

크론 등록 (옵션: 자동 주기 실행)

```
crontab -e

0 9 * * * /home/ubuntu/apps/poll_and_deploy.sh >> /home/ubuntu/apps/re
po/poll.log 2>&1
```

> 매일 아침 9시에 dev 브랜치에 변경사항이 있다면 자동 업데이트

수동 실행

```
# 최초 강제 배포, 가상환경을 만들기 위한 venv 설치하기
# 최신 버전 받기 위해 deadsnakes PPA 추가
sudo apt-get update
sudo apt-get install -y software-properties-common
sudo add-apt-repository -y ppa:deadsnakes/ppa
sudo apt-get update
# 3.11.9 본체 + venv + dev 헤더
sudo apt-get install -y python3.11 python3.11-venv python3.11-dev
~/apps/poll_and_deploy.sh

# FastAPI 로그 확인
sudo journalctl -u fastapi -f

# Frontend 확인 (test 임시용)
```

```
sudo ufw allow 9000
python3.11 -m http.server 9000 --directory ~/apps/frontend/current

# 브라우저: http://<EC2_IP>:9000
```
