8조 CI/CD 구현 계획

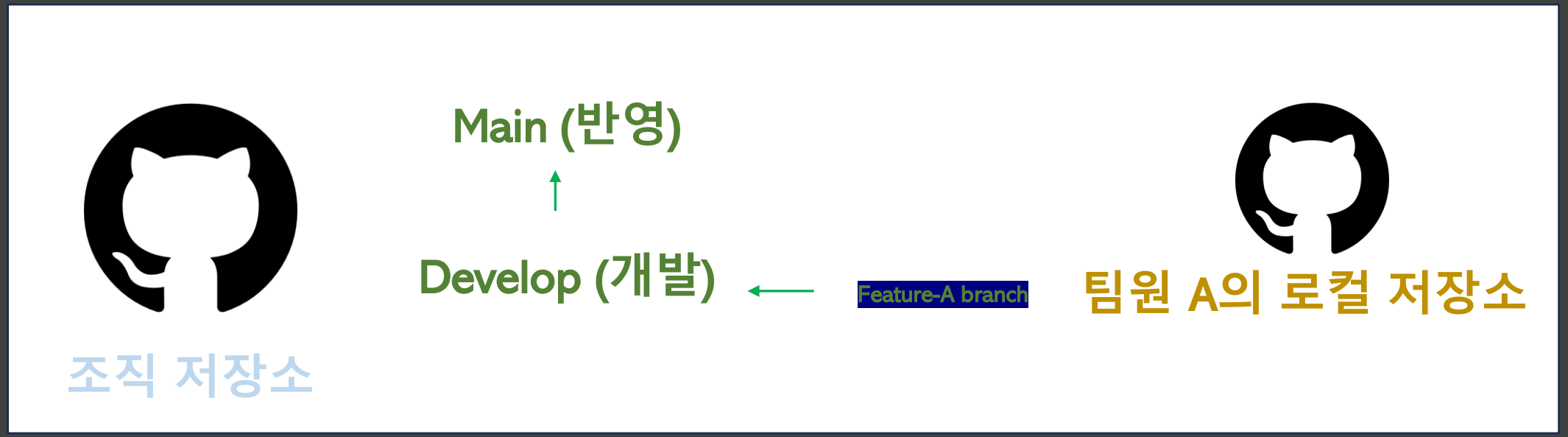
(구현사항 미비로 계획서 제출로 요구사항 조정)

1. CI 파트 구현 계획

1.버전 관리 전략:

[원격 저장소]
(1) MAIN, (2) DEVELOP,

[로컬 저장소]
(2)를 클론 받은 뒤 feature 단위로 작업후 DEVELOP에 PR



1. CI 파트 구현 계획

2. 자동 테스트, 코드 품질 검사:

- ktlint를 이용한 코드 스타일 검사
- gradle을 이용한 테스트 (run: ./gradlew -Dorg.gradle.internal.launcher.welcomeMessageEnabled=false --no-daemon --no-parallel --continue test –stacktrace)
- testCodeTest (향후 예정)

spring-boot-clean / .githooks / pre-commit

hubtwork [Common] Init project (team-hlab#1)

Code Blame Executable File · 26 lines (22 loc) · 1.02 KB Code 55% faster with GitHub Copilot

```
#!/bin/sh
readonly REQUIRED_KTLINT_VERSION=0.50.0
readonly KTLINT_INSTALL_GUIDE_LINK=https://pinterest.github.io/ktlint/$REQUIRED_KTLINT_VERSION/install/cli/#package-managers

# check ktlint installation
if ! [ -x "$(command -v ktlint)" ]; then
  printf 'ktlint is not installed !!\n\nPlease check below links and install ktlint.\n%s\n' $KTLINT_INSTALL_GUIDE_LINK
  exit 1
fi

# check ktlint version
local_ktlint_version="$(printf '%s\n' "$REQUIRED_KTLINT_VERSION" "$(ktlint --version)" | sort -V | head -n1)"
if [ "$local_ktlint_version" != "$REQUIRED_KTLINT_VERSION" ]; then
  printf 'ktlint version update needed!\n\nRequired: %s\nCurrent: %s\n' "$REQUIRED_KTLINT_VERSION" "$local_ktlint_version"
  printf 'please upgrade version over required. (ex: brew upgrade ktlint)\n'
  exit 1
fi

# execute ktlint
if ! git diff --name-only --cached --relative \
  | grep --color=never '\.kt[s"]\?$' \
  | xargs ktlint --code-style=ktlint_official --relative; then
    printf '\nktlint check failed.\nPlease check your code before commit!\n'
  exit 1
fi
```

KTLINT 스타일 검사 예시

1. CI 파트 구현 계획

2. 자동 테스트, 코드 품질 검사:

- ktlint를 이용한 코드 스타일 검사
- gradle을 이용한 테스트 (run: ./gradlew -Dorg.gradle.internal.launcher.welcomeMessageEnabled=false --no-daemon --no-parallel --continue test –stacktrace)
- testCodeTest (향후 예정)

spring-boot-clean / .githooks / pre-commit

hubtwork [Common] Init project (team-hlab#1)

Code Blame Executable File · 26 lines (22 loc) · 1.02 KB Code 55% faster with GitHub Copilot

```
#!/bin/sh
readonly REQUIRED_KTLINT_VERSION=0.50.0
readonly KTLINT_INSTALL_GUIDE_LINK=https://pinterest.github.io/ktlint/$REQUIRED_KTLINT_VERSION/install/cli/#package-managers

# check ktlint installation
if ! [ -x "$(command -v ktlint)" ]; then
  printf 'ktlint is not installed !!\n\nPlease check below links and install ktlint.\n%s\n' $KTLINT_INSTALL_GUIDE_LINK
  exit 1
fi

# check ktlint version
local_ktlint_version="$(printf '%s\n' "$REQUIRED_KTLINT_VERSION" "$(ktlint --version)" | sort -V | head -n1)"
if [ "$local_ktlint_version" != "$REQUIRED_KTLINT_VERSION" ]; then
  printf 'ktlint version update needed!\n\nRequired: %s\nCurrent: %s\n' "$REQUIRED_KTLINT_VERSION" "$local_ktlint_version"
  printf 'please upgrade version over required. (ex: brew upgrade ktlint)\n'
  exit 1
fi

# execute ktlint
if ! git diff --name-only --cached --relative \
  | grep --color=never '\.kt[s"]\?$' \
  | xargs ktlint --code-style=ktlint_official --relative; then
    printf '\nktlint check failed.\nPlease check your code before commit!\n'
  exit 1
fi
```

KTLINT 스타일 검사 예시

1. CI 파트 구현 계획

이미지 PUSH:

[메인에 병합시]
ECR 푸시, ECS 에서 이미지 실행

Feature-A branch
(when main merge)

Main (반영)

Develop (개발)

Feature-A branch

조직 저장소

팀원 A의 로컬 저장소

2. CD 파트 구현 계획

1.버전 관리 전략:

[원격 저장소]
(1) MAIN, (2) DEVELOP,

[로컬 저장소]
(2)를 클론 받은 뒤 feature 단위로 작업후 DEVELOP에 PR

(1) VPC → Public Subnet에서 실행

(2) ECR 이미지 → Task Definition 생성 (fargate) → Target Group 지정.

(3) Load Balancer → target group과 연결

(4) ECS → Load Balancer, Target Group, Auto Scaling, Task Definition 등록

(5) Cloud Watch에서 server container 모니터링