CI/CD의 코드는 아래와 같이 크게 4 부분으로 나뉜다.

1. clone\_repository
2. check\_latest\_commit
3. auto\_testing
4. merge\_branch
5. clone\_repository  
   clone\_repository는 repository-list.txt 파일에서 clone 할 repository의 url을 가져와 remote\_repositories/[ORGANIGATION\_NAME]-[REPOSITORY\_NAME].git의 위치에 cloning을 한다.
   1. $(<[FILE\_NAME])  
      repository의 url을 변수에 저장하기 위해 사용
   2. git clone [URL] [DIRECTORY]  
      github로부터 clone을 위해 사용
6. check\_latest\_commit  
   repository의 디렉토리에서 git 명령어로 fetch를 실시한다.  
   이 때, 새로운 commit이 있으면 merge를 수행한다.
   1. $(find ./ -name .git -type d)  
      [ORGANIGATION\_NAME]-[REPOSITORY\_NAME].git 디렉토리에 cloning을 하였으므로 해당 디렉토리의 위치를 찾아 변수에 저장하기 위해 사용
   2. $([Var]:[Start]:[End]  
      Start번째 문자열에서 End번째 문자열까지 제거한다  
      디렉토리로 이동하기 위한 디렉토리명과 repository의 organization\_name, name을 얻기 위한 전처리로 사용
   3. $(echo [Var] | cut -d '-' -f[Num])  
      -d 옵션을 통해 ‘-‘를 기준으로 문자열을 분리해 [Num]번째 문자열을 얻는다.  
      repository의 organization\_name과 name을 얻기 위해 사용
   4. $(git fetch origin HEAD:HEAD 2>&1)  
      git fetch시 새로운 commit이 있을 때와 없을 때의 출력되는 결과값을 저장하기 위해 사용  
      2>&1은 file discripter의 숫자로, 에러 발생 시 stderr의 형식을 stdout의 형식으로 바꾸어 변수에 저장하기 위해 사용
   5. if [ 0 != ${#Var} ]  
      git fetch로부터 얻은 결과값이 error이면 문자열의 수가 0, error가 아니면 결과물이 출력되어 문자열의 수가 0보다 큰 성질을 이용하여 새로운 commit이 있을 때만 merge를 하기 위해 사용
   6. git merge origin  
      git에 merge를 하기 위해 사용
7. auto\_testing  
   clone된 CICD 내부의 cicd-setting.txt에서 명령어들을 읽고 수행한다.
   1. $(<[FILE\_NAME])  
      cicd-setting.txt에 담겨있는 명령어들을 읽기 위해 사용
   2. $(echo [Var] | cut -d ',' -f[Num])  
      cicd-setting.txt에 ‘,’를 기준으로 구분되어져 있는 명령어들을 분리하여 각각의 변수에 넣기 위해 사용
   3. $(Var)  
      변수에 넣은 명령어들을 실행시키기 위해 사용
8. merge\_branch
   1. $(<[FILE\_NAME])  
      .token 파일에 담겨있는 토큰을 읽기 위해 사용
   2. if [ 0 = ${Var} ]  
      errorflag에 따라 merge, push를 하기위해 사용
   3. git merge origin  
      git에 merge하기 위해 사용
   4. git push origin  
      github에 push하기 위해 사용