Git

Command

1. 브랜치 생성

```
git checkout -b <branchName>
git checkout -b <branchName> <remoteName>/<branchName>
```

2. 원격저장소로 브랜치 Push

```
git push <remoteName> <branchName>
```

3. 브랜치 이동

```
git checkout <branchName>
```

4. 브랜치 삭제

```
git checkout <branchName>
git branch -D <branchName>
#원격 브랜치 삭제 (보통은 remoteName = origin)
git push -D <remoteName> <branchName>
```

5. 브랜치 리스트

```
git branch //로컬 브런치 리스트
git branch -r //원격저장소 브런치 리스트
git branch -a //로컬+원격 브런치 리스트
```

6. 원격저장소 push 기본대상 지정

```
git push -u <remoteName> <branchName>
```

7. 원격저장소 동기화(fetch + merge)

```
#pull = fetch + merge
git pull origin <branchName>
git pull --rebase <remoteName> <branchName>

#커밋불러오기
git fetch <remoteName> <branchName>

#병합
git merge <remoteName>/<branchName> # --squash 옵션을 추가하면 하나의 커밋으로 병합됨

#master 에 branch 를 병합하고 싶다면
git checkout <branchName>
git merge <branchName> # --squash 옵션을 추가하면 하나의 커밋으로 병합됨

git merge --no-commit --squash <branchName>
# 커밋은 되지 않고 스테이징 상태로 남아있게 되므로 다른 브랜치 히스토리와 꼬일 일이 없음
```

8. commit 하기

```
git commit -m <커밋메시지>
```

9. 변경사항 임시저장

```
git stash
# 임시저장
git stash pop
# 임시 저장 버전 재적용(임시저장본은 삭제)
git stash apply
# 임시 저장 버전 재적용(임시저장본은 유지)
# 동일 작업을 브런치 별로 해야 하는 경우 stash 를 이용하여 계속 적용
git stash drop
# 임시 저장버전 삭제
git stash list
# 저장된 리스트 확인
```

10. 상태확인

```
git status
```

11. 로컬변경사항 제거

```
git checkout .
git clean -f
```

12. 계정정보 캐시

```
git config --global credential.helper "cache --timeout 2592000"
```

13. 비교

```
git diff <branchName> <otherBranch> 로컬 브런치 간 비교
git diff <branchName> <remoteName>/<branchName> 로컬과 원격브런치 비교
# --name-only 파일명만 확인
git diff <branchName> <otherBranch> 로컬 브런치 간 비교
git diff <branchName> <remoteName>/<branchName> 로컬과 원격브런치 비교
# --name-only 파일명만 확인

# git 에디터를 vscode 로 변경
git config --global core.editor "code --wait"
# config 파일 열기
git config --global -e
# config 수정
[diff]
  tool = vscode
    prompt = false
[difftool "vscode"]
  cmd = "code --wait --diff $LOCAL $REMOTE"
[difftool]
    prompt = false
# 파일비교
git difftool <branchName> <remoteName>/<branchName>
```

14. 원격저장소 연결 추가, 삭제 및 업데이트

```
git remote add <remoteName> <URL> #추가 #일반적으로는 origin
git remote remote <remoteName> #삭제
git remote -v #정보
git remote update #업데이트
```

15. commit, push 된 버전 지우기

```
git reset HEAD~ #최근 버전
git reset @ID #특정버젼
#옵션
#--hard 파일내용까지 취소
#--mixed staged상태 되돌림
#--soft add 상태로 되돌림
#--merge 바로 이전 병합 취소
```

16. 글로벌 옵션 변경

```
# pull시에 기본적으로 rebase 사용
git config --global branch.autosetuprebase always
# 이미 적용되어 있는 브런치에도 적용
git config branch.<branchName>.rebase true
```

17. 트래킹 브런치 변경

```
git branch -u <remoteName>/<branchName>
# 원격 기준 주소 변경
```

18. commit 내역 합치기

```
git rebase -i HEAD~{n}
#최근부터 n개까지의 commit 을 합친다
```

19. Push 된 폴더 지우기

```
git rm -r --cached <folderPath>
git commit -m <message>
git push
```

- 체크리스트
  - refrog
- 오류
  1. HTTP Basic: Access denied
     - 계정정보 초기화 후 다시 시도

       ```
       git config --system --unset credential.helper
       ```

     - vs 에서 발생 시 아래와 같이 진행 후 시도
       - 제어판 - 사용자 계정 - 자격 증명 관리자 - 일반 자격증명(해당 git 계정 정보 편집)
  2. fetch / merge / full 오류 - Your local changes to the following files would be overwritten by merge

- 수정된 버전이 있어서 발생했으므로 우선 로컬 수정버젼을 임시 저장 후 머지 진행 그후 다시 복원

```
git stash

git pull origin master

git stash pop
```

3. Unable to access "URL" The requested URL returned error: 403
   - 권한없으니 권한 요청해야 함
4. Updates were rejected because the tip of your current branch is behind
   - 원격저장소의 최신상태를 유지한 상태에서 push 를 진행

```
git pull --rebase
git commit -m "내용"
git push {remoteName} {branchName}
```

5. The current branch master has no upstream branch
   - 브랜치가 원격저장소에 없을경우 발생

```
git push -u origin master

git pull origin master
```

6. refusing to merge unrelated histories
   - 원격과 별개로 로컬에 브랜치를 생성하여 push 하는 경우 전혀 상관없는 프로젝트로 판단하여 오류가 발생하는 것으로써 아래 명령어는 병합이 거부된 것을 허용해주는 명령어

```
git pull origin <branchName> --allow-unrelated-histories
```