쉬는 시간(10분) 매시간 50분 ~ 정각

점심시간 11:50 ~ 13:00

QR는 3번 : 9시 입실 / 13시 확인 / 17:51 이후 퇴실

환경 설정

1. Visual Studio Code 가장 먼저 설치

2. Git Bash

select components - default

choose the default editor used by git - Use Visual Studio Code as Git's default editor

adjusting the name of the initial branch in new repositories - Let Git decide

adjusting your path environment - Git from the command line and also from 3rd-party software

choosing the ssh executable - use bundle OpenSSH

choosing Https transport backend - Use the OpenSSL library

configuring the line ending conversion - checkout window-style, commit unix-style line endings

configuring the terminal emulator to use with Git Bash - Use Window default console window

choose the default behavior of git pull - default(fast-forward or merge)

choose a credential helper - Git credential Manager Core

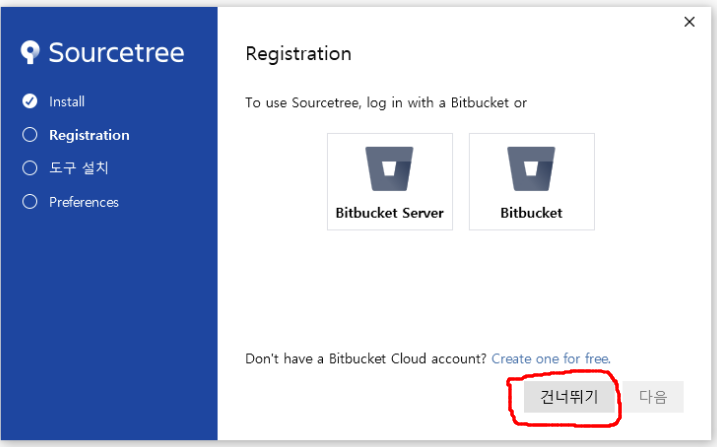
configuring extra options - Enable file system caching

configuring experimental options - 선택않함

$ git --version

git version 2.33.0.windows.1

3. Source Tree 설치



4. GitHub 계정 만들기

<https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home>

github 계정의 username, email 와 password 기억하기

채팅창에 4번까지 완료 하셨으면 “설치완료” 남겨주세요~

4.1 VSCode 에 PlugIn Git Extension Pack 설치

https://marketplace.visualstudio.com/items?itemName=donjayamanne.git-extension-pack

5. c:\MyApp 폴더 생성하기

MyApp-Octocat

MyApp-Raccoon

VSCode를 열어서 File -> Open Folder 메뉴 선택

c:\MyApp\MyApp-Octocat open folder 메뉴를 선택해서 열어 주세요

명령을 복사해서 오른 마우스 클릭 or shift + insert 키를 누르면 붙여넣기가 잘됩니다.

[ Local Repository ]

$ git init

Initialized empty Git repository in C:/MyApp/MyApp-Octocat/.git/

$ ls -al

# .git 폴더 삭제하기

$ rm -rf ./.git

$ git init

# untracked 상태 => tracked 상태로 변경이 됨

$ git add README.md

$ git commit -m "README.md 파일추가"

# tracked 상태 => untracked 상태

# 로컬의 파일은 유지하고 저장소(staging area)에서만 삭제하는 명령 ( commit을 한번도 하지 않았을 경우 )

$ git rm --cached README.md

# 로컬의 파일은 유지하고 저장소(staging area)에서만 삭제하는 명령 ( commit을 한번 했을 경우 )

$ git add index.html

$ git restore --staged index.html

user.name과 user.email 설정을 하지 않고 commit 하는 경우

Author identity unknown

\*\*\* Please tell me who you are.

Run

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

to set your account's default identity.

Omit --global to set the identity only in this repository.

# 설정된 user.name과 user.email 확인하기

$ git config user.name

$ git config user.email

$ git config --global user.name test

$ git config --global user.email test@aa.ccom

# git 사용자 설정 ( 현재 폴더에만 적용 )

git config --local user.name "John Doe"

git config --local user.email johndoe@example.com

#git 전역 사용자 설정 초기화.

git config --global --unset user.name

git config --global --unset user.email

#git 전역 사용자 설정 ( 모든 폴더에 적용 )

git config --global user.name "John Doe"

git config --global user.email johndoe@example.com

# git 전역 credential 사용자 정보 설정

$ git config --global credential.username "John Doe"

$ git config --global credential.useremail johndoe@example.com

# revert initial commit

$ git update-ref -d HEAD

# main 브랜치이름을 main으로 변경하기

$ git branch -M main

# 원격저장소 url 설정하기

$ git remote add origin https://github.com/myvega-2k/MyGitApp.git

# 원격저장소 정보 확인

$ git remote -v

$ git config remote.origin.url

# 원격저장소에 올리기(upload)

$ git push origin main

push 할때 인증하는 방법 2가지

1. username(user email) 와 password Form 인증
2. 발급받은 Token으로 인증

# 원격 저장소 token 번호를 추가한 url 를 재설정

git remote set-url origin [https://토큰번호@github.com/](https://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@github.com/mysoyul/React_Hooks_redux_router.git)myvega-2k/MyGitApp.git

GitHub에서 Token 발급 받기

Settings → [Developer settings](https://github.com/settings/apps) → Personal access tokens (classic) → Generate New Token

Origin : 프로토콜 + 도메인명(IP) + 포트번호

원격 저장소의 이름

Master(Main) : 중심이 되는 브랜치

HEAD : 현재 어떤 브랜치에 있는 지를 나타냅니다.

INDEX : staging area (git add 파일명)

----------------------------------------------------------------------------------

**Push 할 때 발생하는 오류들**

1. 방화벽(firewall) 막혀 있을 때 발생하는 에러

<https://stackoverflow.com/questions/59911649/fatal-unable-to-access-link-getaddrinfo-thread-failed-to-start>

방화벽을 해제하고 다시 push 하면 됩니다.

2. Push 할 때 아래와 같은 오류가 발생하면

hint: 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

$ git pull origin main --allow-unrelated-histories

$ git push origin main

**원격(remote )브랜치가 존재하지만 나타나지 않을 경우**

.git 폴더에 config 파일을 수정합니다.

remote 섹션에 추가한다.

fetch = +refs/heads/\*:refs/remotes/origin/\*

$ git remote update

----------------------------------------------------------------------------------

$ git log

commit 375f488222d9a78efb941901eeac717c37377178 (HEAD -> main)

$ git log

commit 375f488222d9a78efb941901eeac717c37377178 (HEAD -> main, origin/main)

# commit 이력 확인하기

$ git log

# commit 목록 축약해서 출력(커밋해쉬값 축약됨)

$ git log --oneline

# commit 목록 축약해서 출력(커밋해쉬값 전체출력)

$ git log --pretty=oneline

# 커밋 해시와 작성자 출력

$ git log --pretty="%H is by %an"

이전의 최신 커밋정보 2개 출력하기

$ git log -2

# modified 상태인 파일을 최근 커밋 버전으로 되돌리기( add 된 상태에서는 동작하지 않는다)

$ git checkout -- README.md

$ git checkout README.md

# 수정한 파일이 모두 최신 커밋 상태로 되돌려 진다.

$ git checkout .

# add와 commit을 한번에 하기 ( 단 commit 한 이후에만 사용 가능한 옵션 )

$ git commit -am 'readme.md 수정'

# diff 로 modified(변경된) 내용 확인하기

$ git diff index.html

[Clone 명령어]

git clone https://github.com/myvega-2k/MyApp.git

=> MyApp-Raccoon 아래에 원격저장소 이름과 동일한 폴더가 생성되면서 파일들이 복사가 됩니다.

git clone <https://github.com/myvega-2k/MyApp.git> .

=> 위 명령은 폴더가 비어 있는 상태이어야 가능합니다.

=> MyApp-Raccoon 아래에 폴더가 생성되지 않고 원격저장소의 파일들이 현재 폴더에 복사가 됩니다.

[ pull / fetch 명령어 비교 ]

pull => fetch + merge 두 명령을 동시에 해줌

다운(download)과 병합(merge)을 한번에 한다

$ git pull origin main

fetch => 다운(download)만 한다

: 비교(diff)를 한 후에 merge 를 따로 해주어야 한다.

$ git fetch origin main

# 변경된 내용 확인하기 (origin main가 아니라 origin/main)

$ git diff HEAD origin/main

# 변경된 파일 목록만 확인하기

$ git diff HEAD origin/main --name-only

# merge 하기 (# 최신버전(2.40)에서는 origin main 도 가능)

$ git merge origin main

Collaborator 등록하기

GitHub 원격 저장소

Settings -> Collaborators -> Manage Access 메뉴 Add People -> username이나 user email로 검색

이메일로 초대 받은 사람이 허용하면 공동 작업자가 됩니다.

# tag 생성하기 (특정 커밋에 대한 tag 생성)

62842는 커밋해쉬코드 5자리

$ git tag v0.1 62842

# tag 확인

$ git tag

# 생성된 tag를 원격에 저장하기

$ git push origin --tags

------------------

SourceTree에서 push 할 때 인증 오류가 발생하는 경우 인증토큰 발행하기

GitHub -> 계정 -> Settings -> Developer Settings

-> Personal Access Token -> Token(Classic)

생성된 Token번호를 복사

# 원격 저장소 token 번호를 추가한 url 를 재설정

git remote set-url origin [https://토큰번호@github.com/](https://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@github.com/mysoyul/React_Hooks_redux_router.git)myvega-2k/MyGitApp.git

------------------

Git Branch Naming Convention

https://codingsight.com/git-branching-naming-convention-best-practices/

Git Branch Type

https://nvie.com/posts/a-successful-git-branching-model/

Branch 생성(삭제) 및 이동 Merge 하기

#로컬 브랜치 확인

$ git branch -v

#원격 브랜치 확인

$ git branch -r

#로컬,원격 브랜치 모두 확인

$ git branch -a

2개의 새로운 브랜치 생성하기

dev/login 과 feat/login 브랜치를 생성하기

# main 브랜치로 부터 dev/login 브랜치를 생성하고 이동

$ git checkout -b dev/login main

# develop 브랜치 생성

$ git branch dev/login

# develop 브랜치로 이동

$ git checkout dev/login

login.html 추가

# dev/login 브랜치를 원격에 반영하기

$ git push origin dev/login

# main 브랜치에 dev/login 브랜치의 내용을 합치기(merge)

# --no-ff flag causes the merge to always create a new commit object

$ git checkout main

$ git merge --no-ff dev/login

# main 브랜치 위치에서 dev/login 브랜치를 복사해서 feat/login 브랜치를 만든다면

$ git checkout -b feat/login dev/login

# dev/login 브랜치로 이동해서

$ git checkout dev/login

# dev/login 브랜치를 feat/login 브랜치로 복사하고 이동하기

$ git checkout -b feat/login

login.js 수정

# dev/login 브랜치로 이동하기

$ git checkout dev/login

$ git branch -a

\* dev/login

feat/login

main

remotes/origin/main

# dev/login 브랜치에서 feat/login 브랜치의 내용을 merge한다

$ git merge feat/login

# dev/login 브랜치의 내용을 원격에 push

$ git push origin dev/login

# main 브랜치로 이동하기

$ git checkout main

# dev/login 브랜치를 복사해서 새로운 feat/board 브랜치 생성하고 이동하기

$ git checkout -b feat/board dev/login

# main 브랜치로 이동

$ git checkout main

# main 브랜치에 dev/login 브랜치의 내용을 merge

$ git merge dev/login

# main 브랜치의 내용을 원격에 push

$ git push origin main

# feat/login 로컬 브랜치 삭제하기

$ git branch -d feat/login

원격의 dev/login 브랜치 삭제하기

$ git push origin --delete dev/login

# dev/login 브랜치에서 login.html 수정 , feat/login 브랜치에서 login.html 수정

$ git checkout dev/login

# feat/login 브랜치의 내용을 dev/login 브랜치에 merge 할때 충돌이 발생함

$ git merge feat/login

# merging 상태를 취소하기

$ git merge --abort

Conflict(충돌) 해결하기

: Accept current change를 선택하면 dev/login branch의 내용이 남고

: Accept Incoming change를 선택하면 feat/login branch의 내용이 남는다.

: Accept Both Change - 를 선택하면 두 branch의 내용이 모두 남는다.

: Compare - 두개의 source 비교

Git의 추가 명령어들

기존 커밋을 수정하기(amend) - main/ release 브랜치에서는 사용하지 않는 것을 권장합니다.

$ git commit --amend -m 'amend 테스트'

push 하고 난 이후에 amend를 한번 더 했을 경우 에러가 발생하면

! [rejected] dev/login -> dev/login (non-fast-forward)

error: failed to push some refs to 'https://github.com/myvega-2k/MyApp.git'

$ git push origin main -f

수정한 내용을 다른 곳에 저장해 두기 (stash)

$ git stash save "리스트기능"

# stash 목록 확인

$ git stash list

# 저장한 stash를 복원하기

$ git stash apply

$ git stash apply stash@{0}

# stash 를 삭제하기

$ git stash drop (최신 stash 삭제 )

$ git stash drop stash@{0}

# stash 전체 삭제

$ git stash clear

커밋한 내용을 되돌리기 (이력을 남기면서) (revert)

$ git revert HEAD

: 가장 최근에 한 커밋을 되돌리기

: 커밋 메시지를 입력 하라는 Editor에 메시지를 입력하고 close하면

이력이 남고 source 가 이전 상태로 되돌려진다.

HEAD^, HEAD~1 (commit before HEAD)

HEAD~2 (two commits before HEAD)

커밋한 내용을 되돌리기 (이력을 남기지 않음) (reset)

# 커밋이력 없어지고, code도 원래대로 되돌려짐

$ git reset --hard HEAD~1

# 커밋이력 없어지고, code는 되돌리지 않는다

$ git reset --soft HEAD^

원격에 반영된 잘못된 커밋도 이력을 남기지 않고 코드를 원래 상태로 되돌리고 싶다면?

$ git add .

$ git commit -m '잘못된커밋'

$ git push origin dev/login

#로컬에서 커밋이력이 삭제되고, 코드를 원래 상태로

$ git reset --hard HEAD~1

# 원격에도 반영하기

$ git push origin dev/login -f

dev/login 브랜치에서

: 버그수정 으로 commit 한다.

: 일반기능 으로 commit 한다.

$ git checkout main

# main 브랜치에서 버그수정 커밋만 가져오기 (merge하고 commit한다)

$ git cherry-pick dev/login~1

# cherry pick에서 quit 하기

# git cherry-pick --quit

cherry-pick 에 대한 추가 설명

https://stackoverflow.com/questions/9339429/what-does-cherry-picking-a-commit-with-git-mean

—----------

Fork 받을 저장소

<https://github.com/myvega2k/Boxiting>