

Rstudio로 사용하는 Github

<https://mrchypark.github.io/github-with-rstudio>

[pdf버전] [문의하기] [의견 및 오류 신고]

스타누르기는 콘텐츠 제작자를 춤추게 합니다.

박찬엽

2017년 11월 21일

목차

1. 강사 소개

2. 프로젝트 환경 설정

- Rstudio
- Git
- Rtools(for windows)

3. 버전 관리 시스템

- 로컬 저장소, 원격 저장소
- rstudio와 함께 사용하기

4. 원격 저장소 활용

- github에 코드 공유하기
- 다른 사람의 github 코드 활용하기

5. 용어 정리

강사소개

질문 / 상담 / 잡담 대환영!



박찬엽

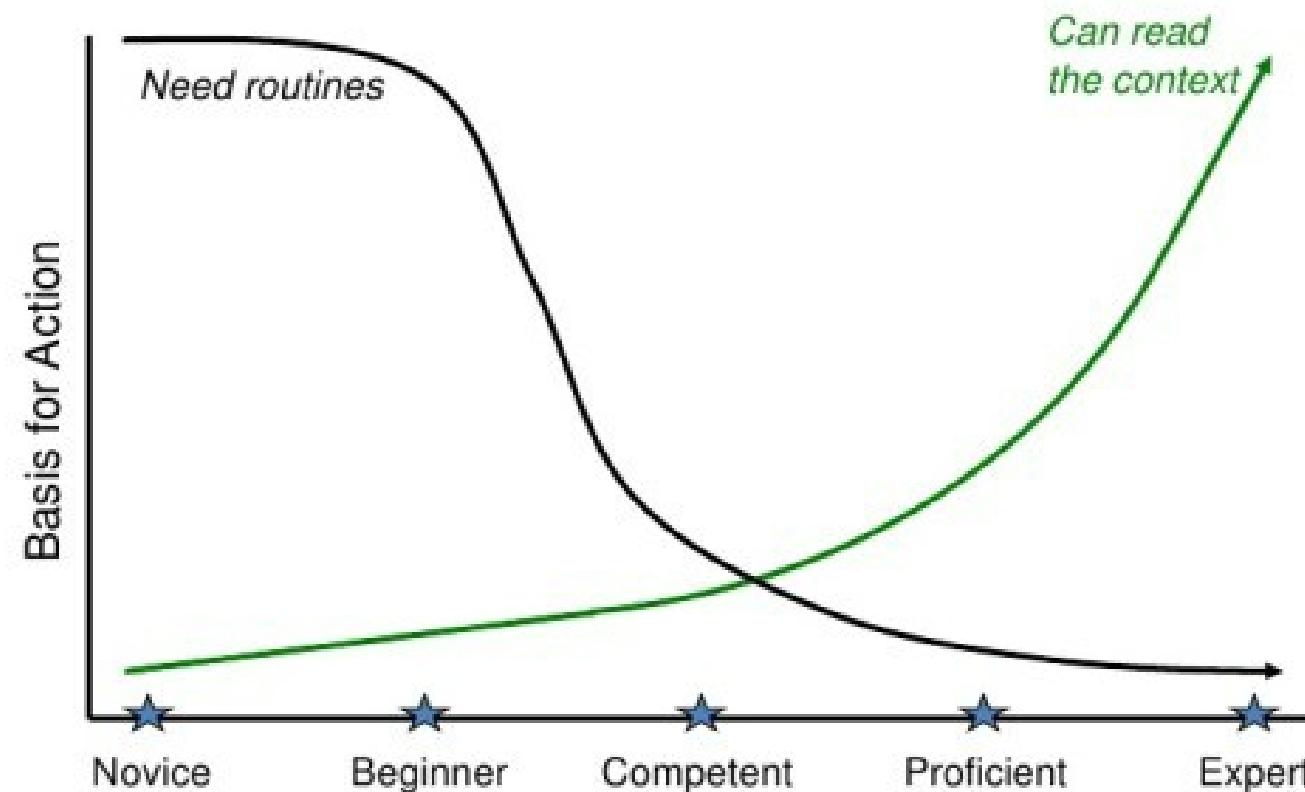
- KAKAO@알코홀릭
- FACEBOOK@mrchypark
- GITHUB@mrchypark
- PHONE+82-10-8720-6115

나는 지금 어느 정도인가?

드레이퍼스 기술 습득 모형

드레이퍼스 모델(Dreyfus Model)은 철학자 드라이퍼스 (Hubert L. Dreyfus)가 그의 저서 Mind Over Machine의 도입부에서 제안한 기술 습득의 5단계 모델¹.

The Dreyfus Model of Skill Acquisition



[1] 드레이퍼스 모델 소개글

드레이퍼스 기술 습득 모형¹

1. Novice 경험이 없기 때문에, 상황에 상관 없이 적용할 수 있는, 다시 말하면, 상황에 따라 적절하지 않을 수 있는, 조리법(recipes) 즉, context-free한 규칙이 주어졌을 때 효과적으로 과업을 수행할 수 있다.
2. Advanced-beginner 여러 가지 상황에서 의미를 가지는 요소에 대해 인지하게 되고, 이러한 요소를 context-free한 규칙에 더해서 활용하게 된다.
3. Competent 경험이 쌓이면서, 상황 하에서 어떤 요소를 중요하게 고려해야 할 지, 상황을 분석하고 규칙에 따라 행동을 선택하여 결과적으로 스스로 문제를 해결(troubleshoot)할 수 있게 된다.
4. Proficient Competent 단계에서 경험한 성공과 실패를 바탕으로, 상황에 따라 어떤 요소를 중요하게 고려해야 할 지를 결정한다. 어떤 요소들이 더 중요하고, 어떤 요소들이 무시해도 되는가는 경험이 추가됨에 따라 직관적으로 변화한다. 반면에, 실제로 어떤 행동을 해야 할 것인가에 대해서는 분석적인 사고를 필요로 한다.
5. Expert 의식적인 사고나 규칙의 필요 없이, 직관적으로 어떤 요소를 중요하게 고려해야 할지와 어떤 행동을 해야 할 것인가를 안다.

[1] 드레이퍼스 모델 소개글

분석하는데 필요한 개발 능력은 (아마도)

Advanced-beginner ~ Competent 사이

novice를 벗어나기 위해서

통계 패키지로 이해하고 있는 경우도 있지만, R은 프로그래밍 언어입니다. 다른 언어와 달리 일반 개발을 위한 용도로 시작하지 않아서 개발에 대한 요소를 전부 알아가며 공부를 할 필요는 없지만 처음 시작할 때 공통적으로 겪는 어려움이 있습니다.

novice를 벗어나기 위해서

통계 패키지로 이해하고 있는 경우도 있지만, R은 프로그래밍 언어입니다. 다른 언어와 달리 일반 개발을 위한 용도로 시작하지 않아서 개발에 대한 요소를 전부 알아가며 공부를 할 필요는 없지만 처음 시작할 때 공통적으로 겪는 어려움이 있습니다.

절차적으로 생각하기

제공되는 함수를 바탕으로 순서대로 동작해야 하는 활동들을 나열하여 연결하는 것을 뜻합니다. 사람은 학교에 간다 같이 세세한 부분을 한번에 뭉개서 생각하기 때문에 세부 사항을 절차적으로 나눠서 생각하는 것이 힘든 일입니다.

novice를 벗어나기 위해서

통계 패키지로 이해하고 있는 경우도 있지만, R은 프로그래밍 언어입니다. 다른 언어와 달리 일반 개발을 위한 용도로 시작하지 않아서 개발에 대한 요소를 전부 알아가며 공부를 할 필요는 없지만 처음 시작할 때 공통적으로 겪는 어려움이 있습니다.

절차적으로 생각하기

제공되는 함수를 바탕으로 순서대로 동작해야 하는 활동들을 나열하여 연결하는 것을 뜻합니다. 사람은 학교에 간다 같이 세세한 부분을 한번에 뭉개서 생각하기 때문에 세부 사항을 절차적으로 나눠서 생각하는 것이 힘든 일입니다.

학교에 간다

샤워를 한다 > 집을 나선다 > 버스를 이용한다 > 학교에 간다

버스를 이용한다

버스카드를 확인하다 > 버스 정류장을 찾는다 > 버스 정류장으로 가다 > 이용가능한 버스를 확인하다 ...

개발 언어에 대한 이해

사람은 비슷하거나 추상화, 그룹화된 것에 대해 종합적을 판단할 수 있기 때문에 그 **즈음**을 말하는 것을 이해할 수 있습니다. 하지만 컴퓨터를 동작하게 하는 개발 언어에서는 사소한 오타나, 미묘한 문법적 차이는 **기대하는 대로 동작하지 않음**을 뜻합니다.

개발 언어에 대한 이해

사람은 비슷하거나 추상화, 그룹화된 것에 대해 종합적을 판단할 수 있기 때문에 그 **조음**을 말하는 것을 이해할 수 있습니다. 하지만 컴퓨터를 동작하게 하는 개발 언어에서는 사소한 오타나, 미묘한 문법적 차이는 **기대하는 대로 동작하지 않음**을 뜻합니다.

언어가 동작하는 방식에 대한 이해

R이 데이터를 다루는 기본적인 동작과 내용을 알고 있는 것은 매우 중요합니다. 같은 방식으로 사고하고 일을 시키지 않으면 R은 기대하는 대로 동작하지 않을 것이기 때문입니다. logical, numeric 같은 자료형서부터 function이 데이터를 다루는 방식을 파악하는 방법을 알아야 합니다.

R이 명령을 수행하기 위해서 R 객체를 메모리에 두고 사용하는 것이라든지, 함수도 R 객체의 일종이라든지 하는 것들을 알고 동작을 확인해보는 것은 매우 의미 있습니다. 하지만 동작을 확인하는 것 자체가 익숙하지 않기 때문에 학습을 위한 빠른 실패와 정보 습득의 습관을 만드는 것이 매우 중요합니다.

Advanced-beginner 를 벗어나기 위해서

잘 된 코드를 흉내내기

내 코드를 다른 사람에게 보여주기

좋은 코드가 있는 공간

think with Google™



코드를 공개하는 공간



SNS with Code

1. 다른 사람이 볼 수 있는 코드를 작성한다.

1. 주석으로 설명 작성
2. 코딩 컨벤션을 준수

2. 버전 관리 시스템을 사용한다.

1. issue로 질의응답
2. Pull Request로 소통

3. 재현 가능한 상태를 유지한다.

1. readme 작성
2. data 동봉 전략

버전 관리와 협업을 위한 환경 설정

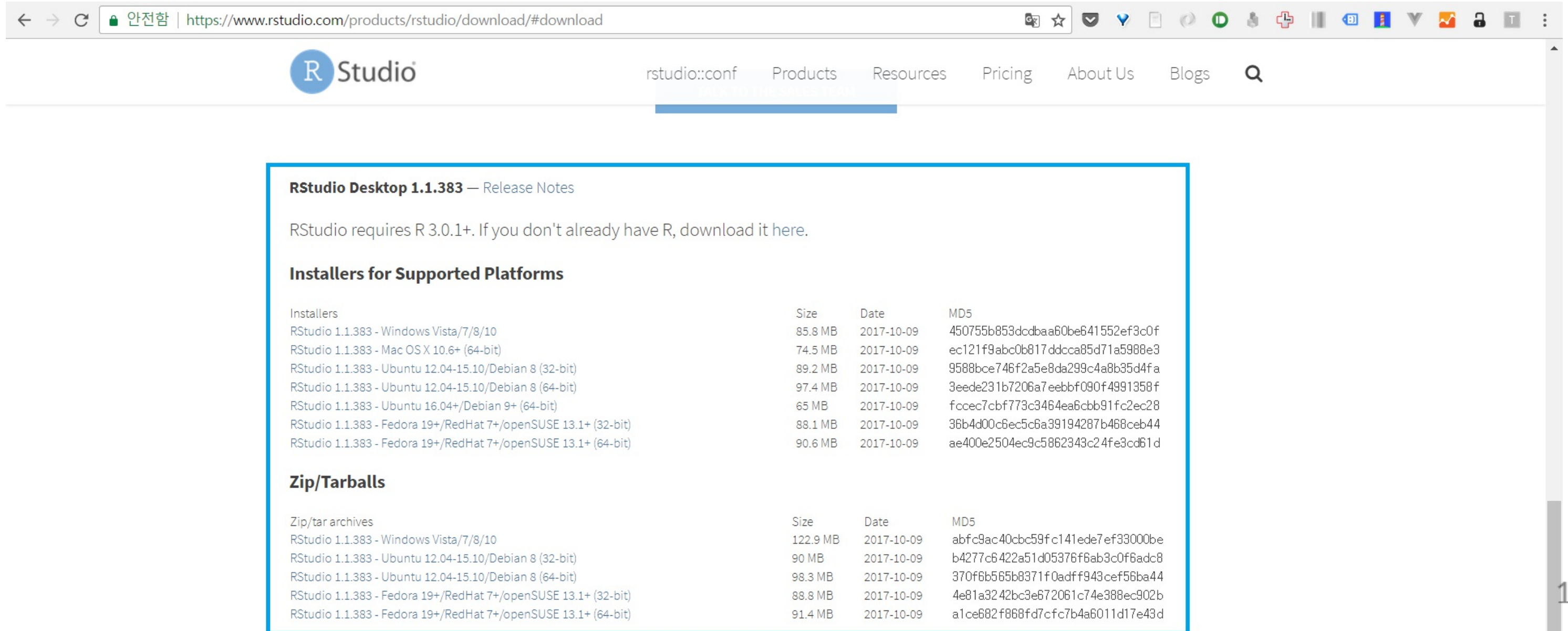
RStudio 1.1

Git

Rtools

RStudio 1.1

Rstudio가 1.1 버전부터 몇 가지 유용한 기능을 추가로 제공하기 시작했습니다. 그래서 이후 안정화 버전이 나오더라도 계속 사용할 기능들을 소개하려고 합니다. 실습을 위한 다운로드 링크[바로가기]에 들어가 각자의 OS에 맞는 버전을 다운 받아 설치해 주세요.



RStudio Desktop 1.1.383 — Release Notes

RStudio requires R 3.0.1+. If you don't already have R, download it [here](#).

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.1.383 - Windows Vista/7/8/10	85.8 MB	2017-10-09	450755b853dcdbaa60be641552ef3c0f
RStudio 1.1.383 - Mac OS X 10.6+ (64-bit)	74.5 MB	2017-10-09	ec121f9abc0b817ddcca85d71a5988e3
RStudio 1.1.383 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.2 MB	2017-10-09	9588bce746f2a5e8da299c4a8b35d4fa
RStudio 1.1.383 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB	2017-10-09	3eede231b7206a7eebbf090f4991358f
RStudio 1.1.383 - Ubuntu 16.04+/Debian 9+ (64-bit)	65 MB	2017-10-09	fccec7cbf773c3464ea6cbb91fc2ec28
RStudio 1.1.383 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB	2017-10-09	36b4d00c6ec5c6a39194287b468ceb44
RStudio 1.1.383 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB	2017-10-09	ae400e2504ec9c5862343c24fe3cd61d

Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 1.1.383 - Windows Vista/7/8/10	122.9 MB	2017-10-09	abfc9ac40cbc59fc141ede7ef33000be
RStudio 1.1.383 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	90 MB	2017-10-09	b4277c6422a51d05376f6ab3c0f6adc8
RStudio 1.1.383 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	98.3 MB	2017-10-09	370f6b565b8371f0adff943cef56ba44
RStudio 1.1.383 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.8 MB	2017-10-09	4e81a3242bc3e672061c74e388ec902b
RStudio 1.1.383 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	91.4 MB	2017-10-09	a1ce682f868fd7cfc7b4a6011d17e43d

windows 사용자 추가 설치

깃(git)은 [여기](#)에서 다운로드 후 설치하실 수 있습니다. git은 리눅스나 맥(mac)에서 기본 제공하기 때문에 따로 설치하실 필요가 없습니다.



git

Rtools는 원래 리눅스, 유닉스 태생의 R을 윈도우에서의 호환성을 높이기 위한 보조도구들을 모아놓은 프로그램입니다. Rtools는 [여기](#)에서 다운로드 후 설치하실 수 있습니다. 34버전을 설치해 주세요.

Building R for Windows

This document is a collection of resources for building packages for R under Microsoft Windows, or for building R itself (version 1.9.0 or later). The original collection was put together by Prof. Brian Ripley; it is currently being maintained by Duncan Murdoch.

The authoritative source of information for tools to work with the current release of R is the "R Administration and Installation" manual. In particular, please read the ["Windows Toolset" appendix](#).

Rtools Downloads

Some of the tools are incompatible with obsolete versions of R. We maintain one actively updated version of the tools, and other "frozen" snapshots of them. We recommend that users use the latest release of Rtools with the latest release of R.

The current version of this file is recorded here: [VERSION.txt](#).

Download	R compatibility	Frozen?
Rtools34.exe	R 3.3.x and later	No
Rtools33.exe	R 3.2.x to 3.3.x	Yes
Rtools32.exe	R 3.1.x to 3.2.x	Yes
Rtools31.exe	R 3.0.x to 3.1.x	Yes
Rtools30.exe	R > 2.15.1 to R 3.0.x	Yes
Rtools215.exe	R > 2.14.1 to R 2.15.1	Yes
Rtools214.exe	R 2.13.x or R 2.14.x	Yes
Rtools213.exe	R 2.13.x	Yes
Rtools212.exe	R 2.12.x	Yes
Rtools211.exe	R 2.10.x or R 2.11.x	Yes
Rtools210.exe	R 2.9.x or 2.10.x	Yes
Rtools29.exe	R 2.8.x or R 2.9.x	Yes
Rtools28.exe	R 2.7.x or R 2.8.x	Yes
Rtools27.exe	R 2.6.x or R 2.7.x	Yes
Rtools26.exe	R 2.6.x or (untested) earlier	Yes

git 확인하기

컴퓨터에 git이 있는지는 terminal에서 git을 입력해 보는 것입니다.

```
MINGW64/c/Users/mrchypark
mrchypark@DESKTOP-BK2SQBJ MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status


grow, mark and tweak your common history
```

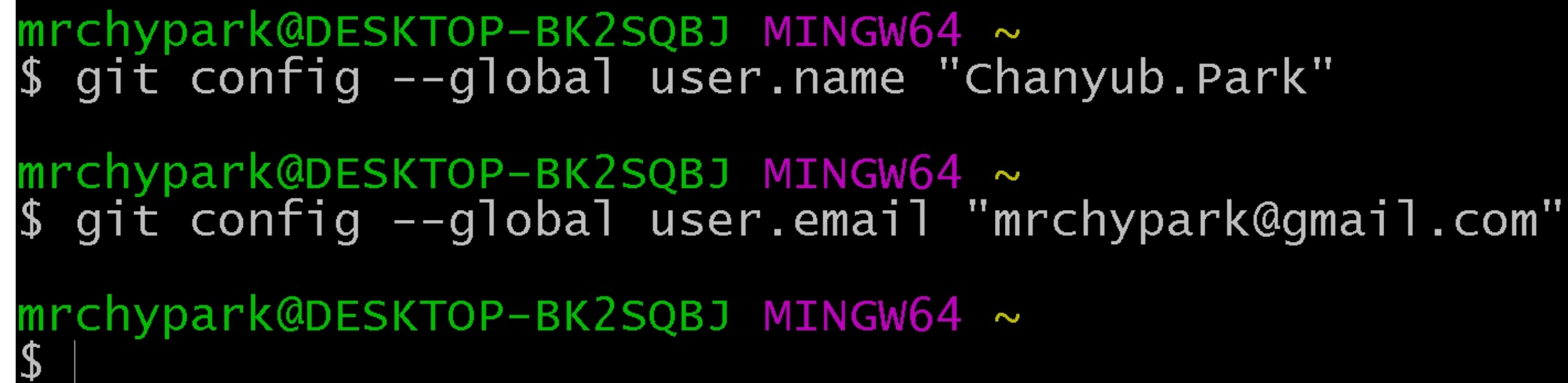
git 유저 설정하기

git은 분산 버전 관리 도구라고 표현합니다. 이 중에 분산은 협업을 위해서 중요한 특징입니다. 협업 때문에 git은 commit이라는 단위에 3가지를 필수로 기록하게 강제해놨는데, user.name, user.email, commit message 입니다. 그래서 앞에 두개는 사전에 설정해야 합니다.

- terminal에서

```
git config --global user.name "자신의 이름- 영어로 작성"  
git config --global user.email "연락을 받을 수 있는 이메일"
```

MINGW64/c/Users/mrchypark

A screenshot of a Windows terminal window with a black background and green text. The window title bar shows 'MINGW64/c/Users/mrchypark'. The terminal displays three lines of commands and their prompts. The first line shows the prompt 'mrchypark@DESKTOP-BK2SQBJ MINGW64 ~' followed by the command '\$ git config --global user.name "Chanyub.Park"'. The second line shows the same prompt followed by '\$ git config --global user.email "mrchypark@gmail.com"'. The third line shows the same prompt followed by '\$ |'.

```
mrchypark@DESKTOP-BK2SQBJ MINGW64 ~  
$ git config --global user.name "Chanyub.Park"  
  
mrchypark@DESKTOP-BK2SQBJ MINGW64 ~  
$ git config --global user.email "mrchypark@gmail.com"  
  
mrchypark@DESKTOP-BK2SQBJ MINGW64 ~  
$ |
```

버전 관리의 개념 이해

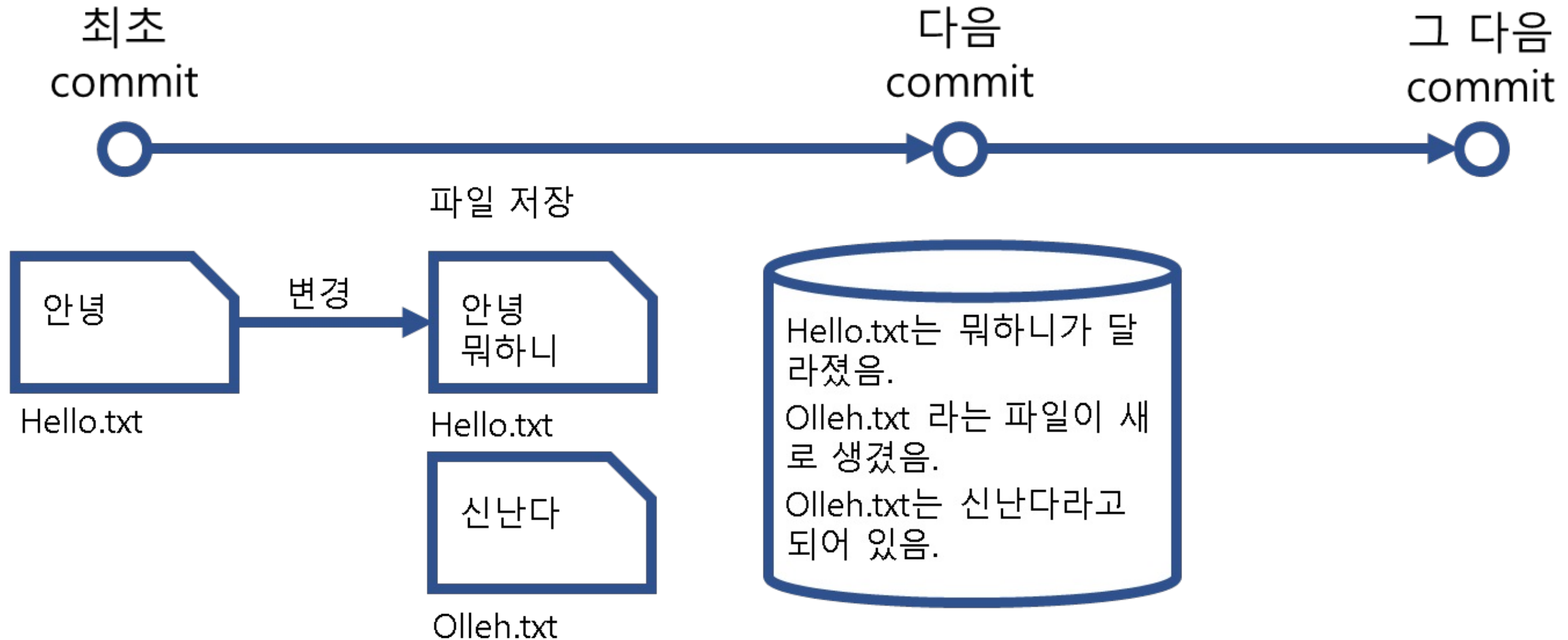
commit == ctrl+z를 저장

일하는 local, 공유하는 remote

복사본을 만드는 branch, 합치는 merge

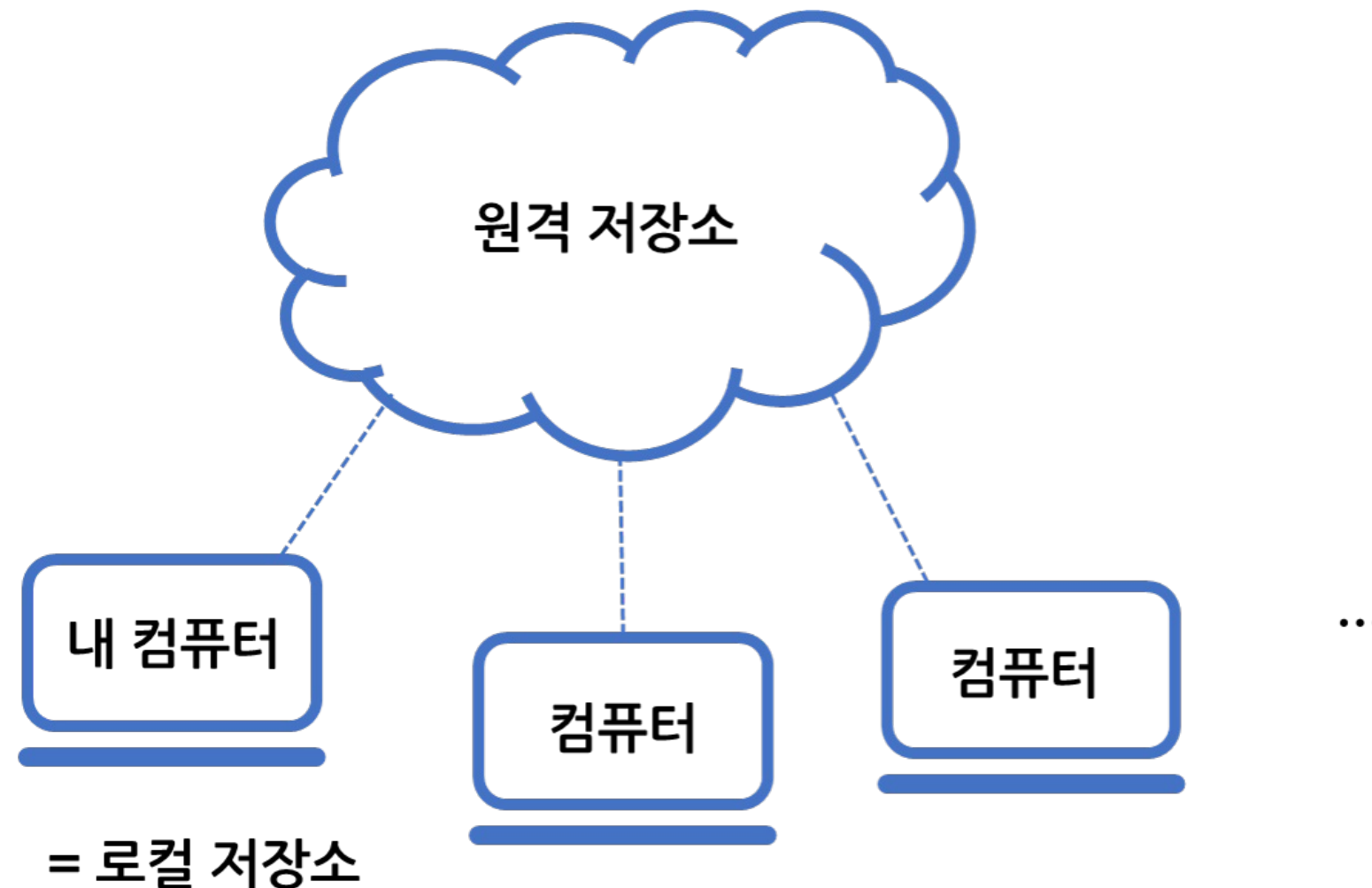
commit == ctrl+z를 저장

ctrl+z는 대표적으로 작업의 과정을 임시로 저장하고 있다가 되돌리는 기능을 의미합니다. commit은 그 되돌아 갈 수 있는 순간을 저장하는 것을 뜻합니다. 보통의 프로그램이 파일의 최종상태만 저장한다면, commit은 기존의 commit과 비교해서 **달라진 점**을 저장해서 ctrl+z를 언제든지 할 수 있게 해주는 역할을 합니다.



일하는 local, 공유하는 remote

코드의 저장 상태 백업과 공유를 위해서 git은 작업하는 나의 컴퓨터 공간인 로컬 저장소(local repository)와 원격 저장소(remote repository)라는 개념으로 구분하여 사용합니다. local을 개인이 일하고 작업하는 공간, remote는 백업하고 공유하는 공간입니다.



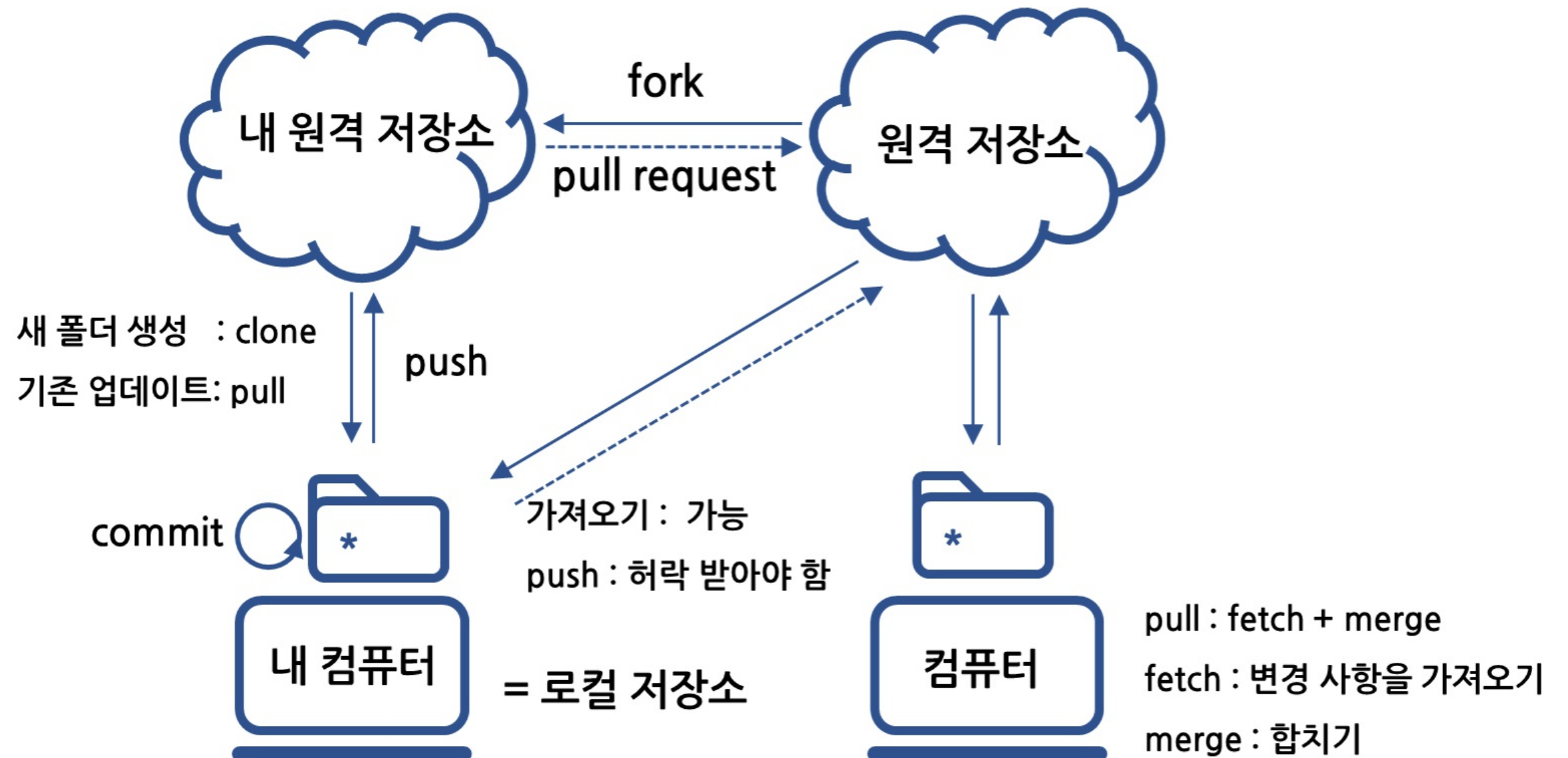
remote 공간 무료 서비스 github

Github은 원격 저장소를 회원 가입한 사람 모두에게 무료로 나눠주는 서비스입니다. 코드로 소통하는 SNS와 같다고 할 수 있습니다.



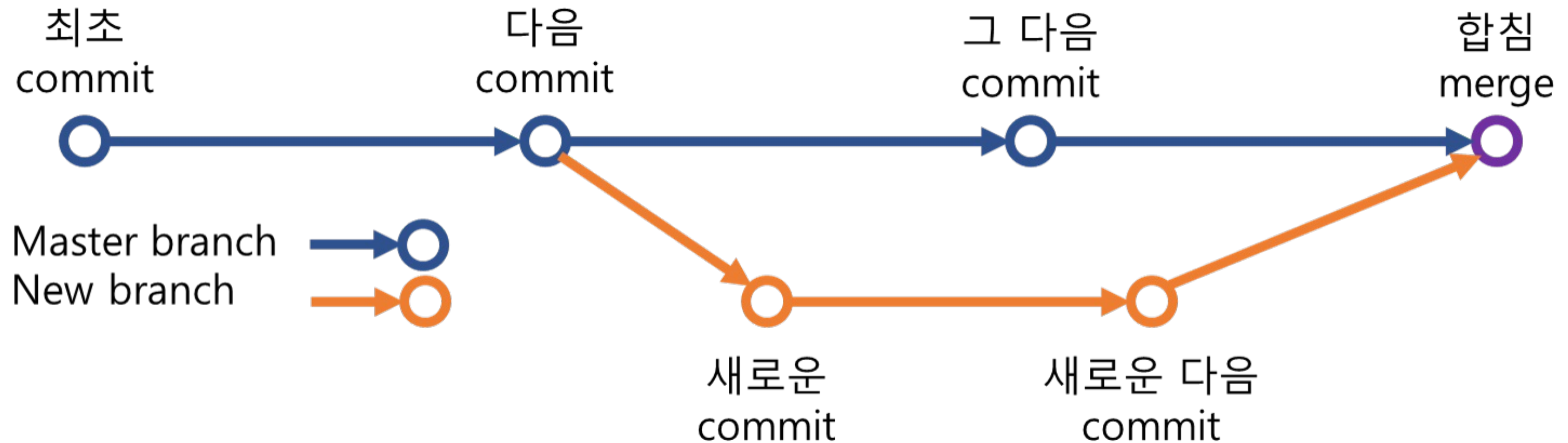
github으로 사용하는 git

github은 git의 원격 저장소를 무료로 제공하는 서비스기 때문에 원격 저장소끼리의 동작은 github의 동작이지 git의 동작이 아닙니다. github은 원격 저장소로 동작하기 때문에 git의 로컬과 원격 저장소의 상호작용에 대한 동작을 그대로 사용할 수 있습니다.



복사본을 만드는 branch, 합치는 merge

git은 달라지는 점을 저장하는 만큼 시간적인 변화 뿐만 아니라 복사본처럼 동시에 여러 버전을 관리할 수도 있습니다. 그 것을 **branch**라고 부르고 각 branch에 이름을 지정하여 옮겨다닐 있습니다. 이 복사본은 폴더 전체를 기준으로 동작합니다.



다른 사람의 코드를 사용하는 법

새로 만드는 clone

pull = local 최신화

실습

새로 만드는 clone

github이 코드로 소통하는 SNS 같은 것이라고 설명했습니다. 그렇기 때문에 공개되어있는 저장소라면 다른 사람의 저장소라도 가져오는 것은 자유롭게 할 수 있습니다. Rstudio의 project 생성을 이용해서 다른 사람의 원격 저장소를 clone해서 가져올 수 있습니다.

- 주의할 점 : 자신의 계정이 아니기 때문에 수정권한이 없어서 push하여 로컬에서 바꾼 부분을 반영할 수 없습니다. 페이스북에서 다른 사람의 프로필을 바꿀 수 없는 것과 비슷합니다.

pull = local 최신화

프로젝트를 원격 저장소를 바탕으로 만들어 clone을 하는 것은 로컬에 처음 프로젝트를 만들때 사용합니다. **pull**은 기존에 가져왔던 시점 이후에 저장소 주인이 무언가 변경점을 만들어 반영을 하면 그 변경된 내용을 로컬 저장소에 가져와서 반영하는 명령입니다.

- 주의할 점 : 모든 파일이 commit이 완료되어 git 탭에 어떤 파일리스트도 없는 깔끔한 상태여야 pull이 문제 없이 진행됩니다.

실습

1. <https://github.com/mrchypark/sejongFinData>를 브라우저로 확인합니다.
2. 주소를 복사합니다.
3. RStudio에서 File > New Project > Version Control > Git 으로 가서 Repository Url에 붙여넣기를 합니다.
4. Create Project 버튼을 눌러 프로젝트를 만듭니다.
5. 위의 주소에 있는 파일들과 같은 것이 폴더 내에 있는지 확인합니다.

나의 코드를 공유하는 법

README 파일을 작성한다.

코딩 컨벤션을 지킨다.

주석을 이해하기 쉽게 작성한다.

실습

README 란

readme 파일은 해당 저장소에 있는 코드들을 사용하는데 있어 필요한 정보들을 담고 있습니다. 대표적으로 간단한 예시, 주의해야 할 점, 설치 방법, 테스트한 환경, 버전에 따른 변경점 등을 작성합니다. README를 충실하게 작성하는 것은 다른 사람들이 잘 사용할 수 있게 유도하는 역할을 하며 Star 등을 통해서 유용한 코드라는 간접 지표를 확보할 수 있습니다.

- issue는 사람들이 사용할 때 문제가 있거나 개선점 제안 등을 할 때 사용합니다. 성실하고 빠르게 대답을 해주거나 코드에 반영하는 것은 많은 사람들이 믿고 코드를 사용할 수 있게 만들어 인기있는 코드로 만드는 요령입니다.

코딩 컨벤션이란

코딩 컨벤션은 코드를 작성할 때 사용하는 작성 규칙을 뜻합니다. 보통 띄어쓰기는 어디서 하는지, 변수명은 어떤 규칙을 통해 만드는지, 줄바꿈은 언제 하는지 같은 규칙이 있으며 R은 구글의 [스타일가이드](#)를 대표적으로 사용합니다.

코딩 컨벤션이란

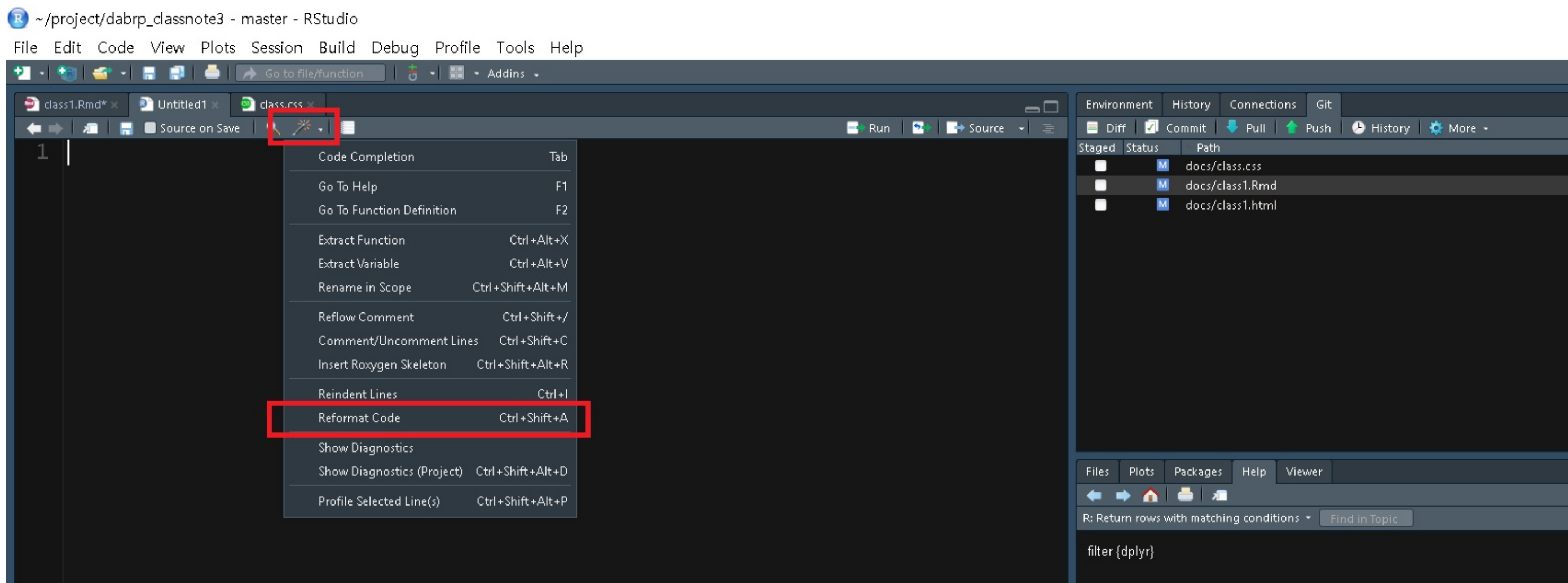
코딩 컨벤션은 코드를 작성할 때 사용하는 작성 규칙을 뜻합니다. 보통 띄어쓰기는 어디서 하는지, 변수명은 어떤 규칙을 통해 만드는지, 줄바꿈은 언제 하는지 같은 규칙이 있으며 R은 구글의 [스타일가이드](#)를 대표적으로 사용합니다.

lint

이런 규칙을 일일이 외워서 사람이 작성하는 것은 어려운 일입니다. 특히 여러 협업하는 사람들이 같은 스타일을 유지한다는 것은 쉬운일이 아닙니다. 명명 규칙같이 온전히 사람이 해야하는 일도 있지만, 띄어쓰기나 줄바꿈 규칙은 프로그램이 대신 정리해 줄 수 있습니다. **lint**는 방금의 두 규칙을 자동으로 정리해주는 것을 뜻합니다.

Rstudio의 lintr : reformat

R script 파일을 사용 중일 때 보이는 마술 지팡이 같은 버튼을 누르면 여러 코드를 작성할 때 유용한 기능들이 나옵니다. 그 중에 우리에게 필요한 것은 **reformat** 버튼입니다.



주석(comment)

가독성 좋게 작성하는 코드는 여러 가지 장점이 있습니다. 우선 다른 사람이 쉽게 이해할 수 있고, 자신도 시간이 지난뒤에 기억을 떠올리기 좋기 때문에 같은 코드를 다시 짜는 문제를 예방할 수 있습니다. 하지만 코드의 형식을 따라야 하기 때문에 아무리 가독성이 좋아도 다시 코드를 파악하는데 어려움이 있을 수 있습니다. 이 때 주석 기능을 이용해서 설명을 작성해두면 더 빠르게 코드를 파악할 수 있습니다.

```
13
14  tqk_get<-function(code, from="1900-01-01", to=Sys.Date(), tqform=T, source=c("p","d","n")){
15      # todo
16      # now just use p source only
17      print("please wait for getting data using internet.")
18      print("close and adjusted are same now.")
19      root<-"http://paxnet.moneta.co.kr/stock/analysis/pagingListAjax?method=listByDate&abbrSymbol="
20      tar<-paste0(root,code,"&currentPageNo=1")
21
```

실습

1. github에서 새로운 저장소를 만듭니다.
2. 새로 만든 저장소의 주소를 복사합니다.
3. Rstudio에서 File > New project > version control > Git 으로 이동합니다.
4. Repository Url 에 2.에서 복사한 주소를 붙여넣습니다.
5. 저장소 이름의 폴더가 생성될 경로를 설정합니다.
6. Create project 버튼을 누릅니다.
7. 데이터 파일이나 스크립트 파일을 생성하고 작성합니다.
8. reformat 버튼을 눌러 코드가 정리되는 것을 확인합니다.
9. 파일들의 변경을 commit합니다.
10. push 버튼을 누릅니다.
11. 자신의 github 페이지로 돌아가서 변경이 반영됐는지 확인합니다.

다른 사람과 코드로 협업할 때

Fork로 가져오기

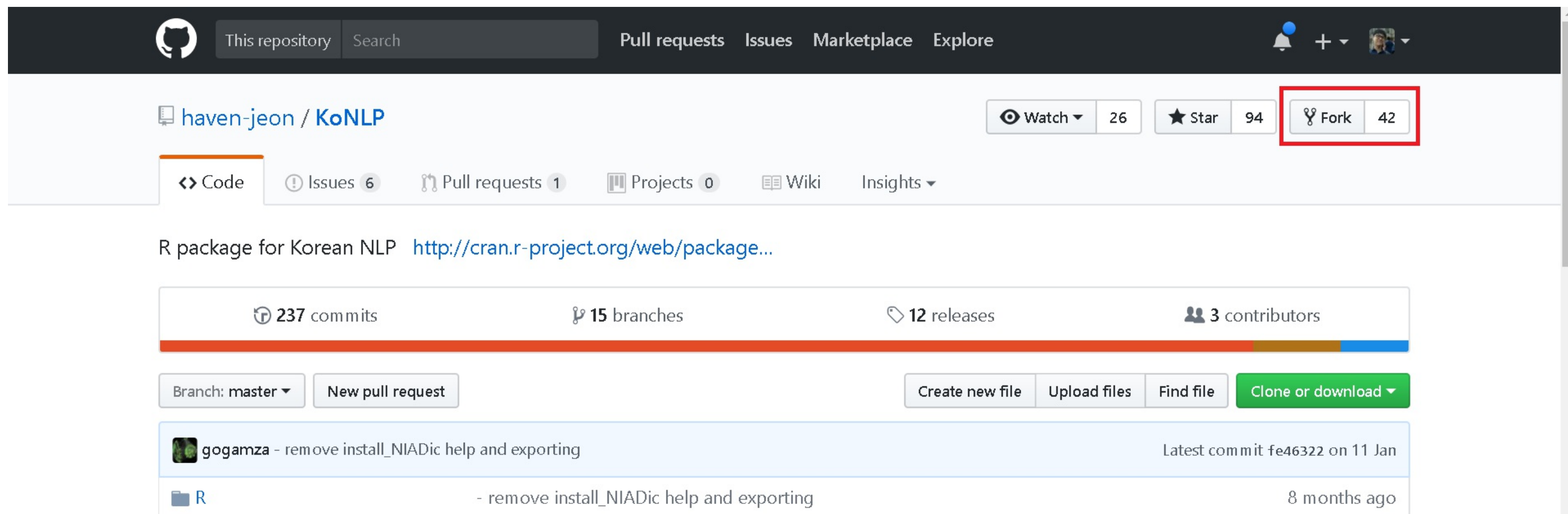
Pull Request로 가져가라고 요청하기

remote 이해하기

실습

원격 저장소 복사해오는 Fork

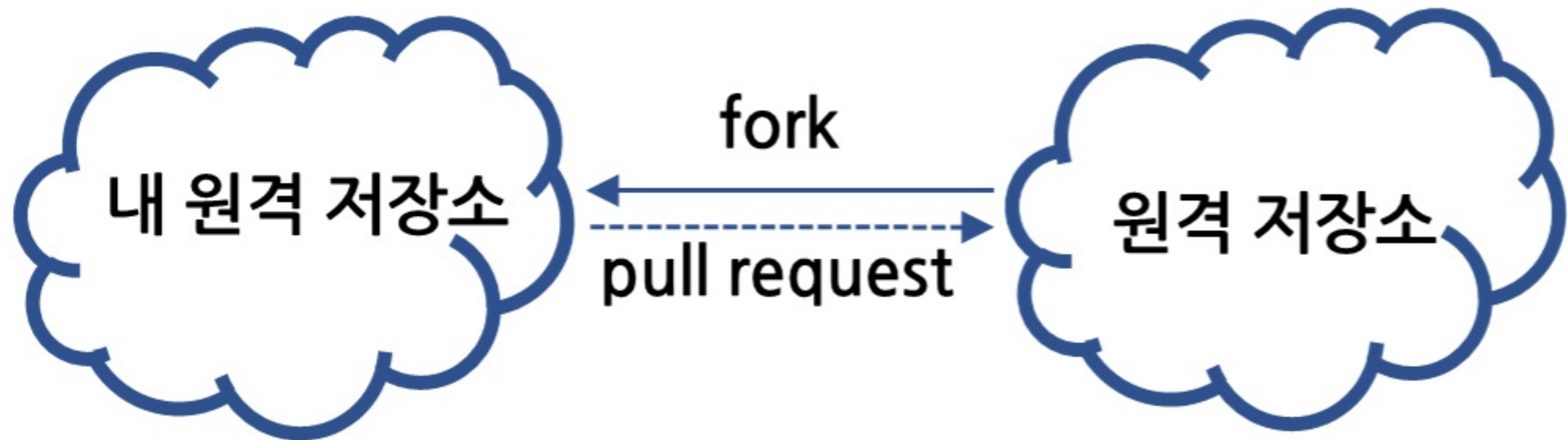
fork는 다른 사람의 원격 저장소 하나를 복사해서 자신의 원격 저장소에 저장하는 기능을 수행합니다. 다른 사람의 코드를 로컬에 가져와서 변경을 한 후 반영하는데 권한이 없었던 걸 앞에서 같이 해봤는데요. 그렇게 fork해 온 저장소는 자신의 계정내에 있기 때문에 변경 권한이 있습니다. 그래서 스스로 만든 저장소처럼 push를 진행할 수 있습니다.



The screenshot shows the GitHub repository page for `haven-jeon / KoNLP`. The repository is an R package for Korean NLP, with a link to the CRAN page: <http://cran.r-project.org/web/package...>. The repository has 237 commits, 15 branches, 12 releases, and 3 contributors. The 'Fork' button is highlighted with a red box, indicating the action to create a local copy of the repository. Below the repository information, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The latest commit is by `gogamza` with the message 'remove install_NIADic help and exporting' on 11 Jan. The repository was last updated 8 months ago.

내가 고친 부분을 사용하라고 요청하는 Pull Request

pull이 원격 저장소에서 가져오는 것이라고 했습니다. 다른 사람의 원격 저장소 입장에서는 저의 원격 저장소가 **원격 저장소** 일 수 있습니다. 그렇기 때문에 pull request라고 하는데요, 다른 사람의 원격 저장소에 제 원격 저장소의 변경 사항을 가져가라고 요청을 보내 변경 사항을 반영하는 것입니다. 용어가 어색할 수 있어서 다른 서비스에서는 merge request라고 하기도 합니다. 하지만 이미 너무 많은 개발자들이 **PR**이라고 이야기 하고 있어 Pull Request라고 하였습니다.



Fork해온 원본과 sync

Fork는 동작하는 그 순간의 내용들만 가져옵니다. 이후에 원본 저장소가 업데이트되었다고 해서 자동으로 같이 업데이트해주지 않습니다. Terminal 탭에서 git remote 명령어를 통해 원본 내용을 통합하는 방법을 알아보겠습니다.

```
# repo와 연결된 원격 repo 주소 확인
git remote -v

# sync를 맞추기 위한 원본 repo 주소를 upstream 이라는 이름으로 추가
git remote add upstream 원본 repo 주소

# pull 하기 전 commit 진행
git commit . -m "for sync"

# upstream에서 pull 진행
git pull upstream master
```

실습

1. <https://github.com/mrchypark/github-with-rstudio>를 브라우저로 확인합니다.
2. **FORK** 버튼을 눌러 자신의 계정으로 복사합니다.
3. <https://github.com/자신의계정/github-with-rstudio> 주소를 복사합니다.
4. Rstudio에서 File > New project > version control > Git 으로 이동합니다.
5. Repository Url 에 2.에서 복사한 주소를 붙여넣습니다.
6. 저장소 이름의 폴더가 생성될 경로를 설정합니다.
7. Create project 버튼을 누릅니다.
8. 새 스크립트 파일을 만들어서 `print("hello world!")` 를 입력합니다.
9. 만든 스크립트 파일을 FolderForClass1 폴더 안에 저장합니다.
10. 저장은 [자기이름영문으로].R 이름으로 합니다.
11. 파일들의 변경을 commit합니다.
12. push 버튼을 누릅니다.
13. 자신의 github 페이지로 돌아가서 변경이 반영됐는지 확인합니다.
14. github내 Pull Request 탭으로 가서 create Pull request 버튼을 누릅니다.
15. pull request가 정상적으로 요청됐는지 확인합니다.

용어 정리

git

- init : 폴더내를 git의 관리하로 만들기
- clone : 원격 저장소를 로컬에 복사해 가져오기
- commit : 이전 저장 상태와 비교해 바뀌 점을 저장하기
- commit message : 바뀐 점이 무엇인지 작성해 두는 것
- pull : 연결된 원격 저장소에서 변경된 점 가져와 합치기
- push : 로컬 저장소에서 변경된 점을 원격 저장소에 올리기
- local : 코드 수정등의 작업을 하는 공간(내 노트북)
- remote : 로컬과 대비되는 개념
- repository : 코드 저장소
- branch : 폴더 전체에 대한 복사본을 만들어 관리
- merge : 다르게 관리된 두 개의 branch를 합치는 것
- conflict : 합칠 때 자동으로 진행하지 못하는 부분이 발생한 것

용어 정리

github

- watch : 메일로 저장소의 소식을 받아 보기
- star : 즐겨찾기
- fork : 자기 계정의 원격 저장소로 복사
- pull request :
- issue : 코드에 대해 의견을 나누는 공간
- author : 코드 저장소의 주인
- contributor : 코드 작성을 도와준 사람